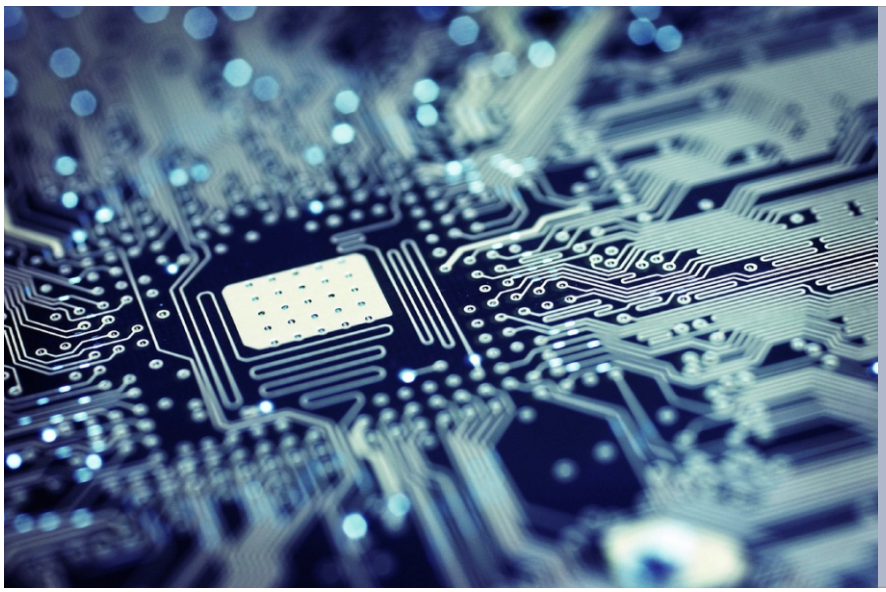
PAD 2016
Bořetice - Kraví Hora

Sborník příspěvků

Počítačové architektury & diagnostika

Česko-slovenský seminář pro studenty doktorského studia



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

14. - 16. 9. 2016

Počítačové architektury & diagnostika PAD 2016
Česko-slovenský seminář pro studenty doktorského studia

Sborník příspěvků

Editoři: Jiří Jaroš a Richard Růžička

Bořetice, Kraví Hora
14. září 2016 – 16. září 2016

ISBN 978-80-214-5376-0

Všechny příspěvky byly vytištěny podle podkladů dodaných autory příspěvků.

NÁVRH OBÁLKY
Marta Čudová

VYDALO

Vysoké učení technické v Brně,
Fakulta informačních technologií
Brno, Česká Republika

ÚVODNÍ SLOVO

Vážení školitelé, oponenti, studenti, zkrátka vážení kolegové,

setkáváme se již po čtrnácté nad sborníkem prací studentů doktorského studia, vydaným u příležitosti semináře Počítačové architektury a diagnostika, zkráceně PAD. Již sám tento fakt ukazuje, že myšlenka semináře, na němž by doktorandi spřátelených pracovišť z České a Slovenské republiky získávali nové podněty a objevovali jiné úhly pohledu na svoji výzkumnou práci, je dobrá a životaschopná. V době honby za formální kvalitou publikací v prestižních časopisech a na elitních konferencích, v době diskurzu internacionalizace vysokého školství a tlaku na kvantitu výsledků není snadné najít místo, kde bychom svobodně prezentovali a diskutovali výzkumnou práci, která je třeba teprve rozpracovaná nebo dokonce ještě ve stádiu prvních úvah a pokusů. Prestižní konference (nebo ty, které alespoň chtějí patřit mezi prestižní) takovou situaci nepřipouští, konference na opačném konci spektra, řekněme nízkoprahové, zase nedají zpětnou vazbu téměř žádnou.

Ostatně PAD přece není konference. Cílem není publikovat, získat levně „čárku.“ I když se to neustále zdůrazňuje, každý rok se najde někdo, kdo to nepochopí. To nám ale nevadí. Kdo chce, získá zde něco mnohem cennějšího než publikaci. Snahou oponentů není příspěvky filtrovat, ale dovést příspěvek a jeho autora k formě a obsahu prezentace, dovolující v následné diskusi získat co nejvíce dobrých připomínek k dosavadní práci a podnětů k práci další.

Velmi si vážím toho, že jsem se Zdeňkem Kotáskem a Lukášem Sekaninou byl u úplně prvního semináře PAD, kdy jsme ve Zvíkovském Podhradí navazovali na tradici předchozí generace školitelů a myšlenky zvěčnělého prof. Hlavičky, a je mi ctí, že jsem se v podobné pozici ocitl nyní, i když trochu náhodou, opět. V organizaci semináře se již několikrát vystřídala všechna zúčastněná pracoviště a pomyslný štafetový kolík třímá letos opět Ústav počítačových systémů Fakulty informačních technologií VUT v Brně, pracoviště vlastně jen o rok starší než série seminářů PAD. Pozvali jsme Vás tentokrát do kraje vína, do kraje odrůdy André, do té pravé jižní Moravy, kterou ve skupině pořadatelů seminářů PAD zastupujeme. Do kraje vína, slunce, pohody a snad ještě stále o něco pomalejšího života, než panuje na dnešních univerzitách a v dnešních městech. Přestože to, takto napsáno, zní jako laciné klišé, přišlo nakonec 28 příspěvků. To naznačuje, že jsme nevolili špatně a nás pořadatele to zavazuje, abychom Vám pomohli vytvořit atmosféru, kterou očekáváte.

Svobodná spolková republika Kraví Hora, která nás letos hostí, leží právě uprostřed oblasti, poeticky zvané Modré Hory a vymezené Velkými Pavlovicemi a Vrbicí. Modré snad proto, že se zde díky kromobyčejně příznivému klimatu výtečně daří i modrým hroznům a jak již bylo řečeno, zdejší šlechtitelská stanice dala světu křížením Frankovky a Svatovavřineckého i novou odrůdu André, dávající vína plné chuti a tmavě granátové barvy. Samotná republika je útvar spíše recesistický než politický a spolková je proto, že areál vinných sklepů, který ji tvoří, sestává ze tří částí. Stylový hotel, přirozeně zapadající do této malebné enklávy na jihozápadním okraji, se stává pro letošek centrem PADu a naše jednání, jak doufáme, obohatí o pozitivní a uvolněnou atmosféru. Specialitou letošního ročníku je, že kromě obligátního auta či železnice (zastávka Bořetice na trati 255 do Hodonína je jen 400 m od hotelu) lze poprvé v historii PADu užít k dopravě i letadlo. Letiště je doslova jen pár kroků. Ostatně i já jsem toto místo pro Vás objevil ze vzduchu. Věříme, že jsme učinili dobrou volbu a že letošní PAD bude pro nás všechny úspěšným jednáním a příjemným setkáním a doufáme, že nám, organizátorům-amatérům, případné nedostatky prominete.

Tož hodně úspěchů při prezentacích a na zdraví!

V Brně 14. září 2016 Richard Růžička

ŘÍDÍCÍ VÝBOR

Vladimír Drábek	FIT VUT v Brně
Elena Gramatová	FIIT STU Bratislava
Zdeněk Kotásek	FIT VUT v Brně
Robert Kvaček	ASICentrum Praha
Róbert Lórencz	FIT ČVUT v Praze
Ondřej Novák	FMIMS TU v Liberci
Stanislav Racek	FAV ZČU v Plzni
Viera Stopjaková	FEI STU Bratislava (předseda řídicího výboru)

PROGRAMOVÝ VÝBOR

Jan Dohnal	ON Semiconductor
Karel Dudaček	FAV ZČU v Plzni
Petr Fišer	FIT ČVUT v Praze
Jiří Jaroš	FIT VUT v Brně
Katarína Jelemenská	FIIT STU v Bratislavě
Jan Kořenek	FIT VUT v Brně
Tomáš Koutný	FAV ZČU v Plzni
Štefan Krištofik	FIIT STU v Bratislavě
Hana Kubátová	FIT ČVUT v Praze
Robert Lórencz	FIT ČVUT v Praze
Dominik Macko	FIIT STU v Bratislavě
Ondřej Novák	FM TU v Liberci
Antonin Pleštil	ASICentrum
Zdeněk Plíva	FM TU v Liberci
Stanislav Racek	FAV ZČU v Plzni
Richard Růžička	FIT VUT v Brně (předseda programového výboru PAD 2016)
Jan Schmidt	FIT ČVUT v Praze
Viera Stopjaková	FEI STU v Bratislavě
Josef Strnadel	FIT ČVUT v Praze
Tomáš Zahradnický	FIT ČVUT v Praze
Marcela Zachariášová	FIT VUT v Brně

ORGANIZAČNÍ VÝBOR

Richard Růžička	FIT VUT v Brně
Jiří Jaroš	FIT VUT v Brně
Marta Čudová	FIT VUT v Brně

OCENĚNÍ STUDENTI PAD 2015

První ročník (8 studentů)
Cena prof. Jana Hlavičky, DrSc: Vojtěch Mrázek (FIT VUT)
Ocenění za dobrý start do PhD studia: Juraj Šubín (FIIT STU), Matěj Bartík (FIT ČVUT)

Druhý ročník (5 studentů)
Cena prof. Jana Hlavičky, DrSc: neudělena
Ocenění za kvalitní práci: Josef Kokeš (FIT ČVUT), Martin Kováč (FEI STU)

Třetí ročník (1 student)
Cena prof. Jana Hlavičky, DrSc: neudělena

PODĚKOVÁNÍ

Organizátoři semináře PAD 2016 děkují Fakultě informačních technologií VUT v Brně. Seminář je organizován s podporou projektu FIT-S-14-2297 Architektura paralelních a vestavěných počítačových systémů.

Organizátoři dále děkují společnosti ASICentrum s.r.o a NXP s.r.o. za finanční příspěvek, který podpořil konání semináře.

SEZNAM STUDENTŮ, JEJICH VEDOUCÍCH A RECENZENTŮ

Student	Ročník	Školitel	Oponent 1	Oponent 2
Michal Riša	1	Strnadel	Jelemenská	Koutný
Jiří Čech	1	Plíva	Zahradnický	Jaroš
Ivo Háleček	1	Fišer	Jelemenská	Novák
Martin Huněk	1	Plíva	Jaroš	Stopjaková
Matej Rakús	1	Stopjaková	Dudáček	Kubátová
Filip Kodýtek	1	Lórencz	Strnadel	Kořenek
Šimon Danko	1	Stopjaková	Jaroš	Dohnal
Filip Vaverka	1	Jaroš	Racek	Lórencz
Lukáš Kohútka	1	Stopjaková	Dudáček	Ružička
Jan Nevoral	1	Růžička	Krištofík	Stopjaková
Robert Hülle	1	Schmidt	Dohnal	Krištofík
Vojtěch Miškovský	1	Kubátová	Zachariášová	Strnadel

Student	Ročník	Školitel	Oponent 1	Oponent 2
David Grochol	2	Sekanina	Fišer	Kubátová
Vojtěch Mrázek	2	Sekanina	Macko	Schmidt
Juraj Šubín	2	Gramatová	Novák	Pleštil
Martin Krčma	2	Kotásek	Fišer	Schmidt
Ondrej Perešíni	2	Krajčovič	Zachariášová	Koutný
Ondrej Kachman	2	Hluchý/Baláž	Zachariášová	Jaroš
Vojtěch Nikl	2	Jaroš	Macko	Racek
Jan Bělohoubek	2	Fišer	Strnadel	Krištofík
Matěj Bartík	2	Ubik/Kubalík	Zachariášová	Ružička

Student	Ročník	Školitel	Oponent 1	Oponent 2
Adam Crha	3	Růžička	Pleštil	Schmidt
Radek Tesař	3	Růžička	Fišer	Plíva
Ondřej Čekan	3	Kotásek	Fišer	Plíva
Jakub Podivínský	3	Kotásek	Plíva	Stopjaková
Martin Hyrš	3	Schwarz	Koutný	Lórencz

OBSAH

1. ročník	Strana
Michal Riša Scheduling and Synchronization on Multicores	10
Jiří Čech Zisk a zpracování hyperspektrálních dat	14
Ivo Háleček Logická syntéza s nativní podporou XOR hradel	18
Martin Huněk Zpracování signálů EEG na obvodech FPGA	22
Matej Rakús Analýza prúdových zrkadiel riadených substrátovou elektródou	25
Filip Kodýtek A ring oscillator based PUF proposal on FPGA	29
Šimon Danko Optimalizácia spotreby energie komunikačného modulu v implantovateľných senzorických systémoch	33
Filip Vaverka Case Study on Multi-domain Decomposition of k-Wave Simulation Framework	37
Lukáš Kohútka Hardvérová platforma pre systémy reálneho času	41
Jan Nevoral Polymorfní obvody na bázi ambipolárních tranzistorů	45
Robert Hülle Generování testu pro prostředky vestavěné diagnostiky	49
Vojtěch Miškovský Číslicový návrh spojující odolnost proti útokům a odolnost proti poruchám	53

2. ročník	Strana
David Grochol Evoluční hardware v síťových aplikacích	57
Vojtěch Mrázek Evoluční snižování příkonu: Od obvodů na úrovni tranzistorů po neuronové sítě na čipu	61
Juraj Šubín Návrh metód generovania BIST pre vnorené pamäte v systémoch na čipe	65
Martin Krčma Koncept Field Programmable Neural Networks odolný proti poruchám	69
Ondrej Perešíni Heterogénne smerovanie v kapilárnych sieťach internetu vecí s použitím kompozitnej metriky	73
Ondrej Kachman Configurable Reprogramming Scheme for Over-the-Air Updates in Networked Embedded Systems	77
Vojtěch Nikl High Performance Computing on Low Power Devices	81
Jan Bělohoubek Využití rychlého offline testu v systému se schopností maskování jedné chyby	85
Matěj Bartík Nová metoda realizace hashovacích tabulek a paměťově orientovaných struktur na FPGA	89

OBSAH

3. ročník	Strana
Adam Crha	
Polymorfní elektronika a metody syntézy	93
Ondřej Čekan	
Generování testovacích stimulů	97
Jakub Podivínský	
Funkční verifikace jako nástroj pro sledování vlivu poruch na elektro-mechanický systém	101
Martin Hyrš	
Kopulové EDA algoritmy	105

Scheduling and Synchronization on Multicores

Michal Riša
1st year, full-time study
Supervisor: Josef Strnadel

Brno University of Technology, Faculty of Information Technology
Božetěchova 2, 612 66 Brno, Czech Republic
irisa@fit.vutbr.cz

***Abstract*—The article presents a motivation, basic terms, principles and problems related to applications created on basis of multicore platforms. Then, state of the art is outlined in brief, followed by a summary of work being done in preceding MSc thesis. Afterwards, ideas behind prepared PhD thesis are outlined such as research directions, goals and roadmap, followed by details to research questions and hypothesis. At the end, methods and instruments to reach the goals are summarized and the paper is concluded.**

***Keywords*—scheduling, synchronization, multicore**

I. Introduction

For the past 50 years, Moore's law accurately predicted that the number of transistors on an integrated circuit would double every two years. To translate these transistors into equivalent levels of system performance, chip designers increased [1]:

- Clock frequencies (requiring deeper instruction pipelines),
- instruction level parallelism (requiring concurrent threads and branch prediction),
- memory performance (requiring larger caches),
- and power consumption (requiring active power management).

Each of these four areas is hitting a wall that impedes further growth [1]:

- Increased processing frequency is slowing due to diminishing improvements in clock rates and poor wire scaling as semiconductor devices shrink,
- instruction-level parallelism is limited by the inherent lack of parallelism in the applications,
- memory performance is limited by the increasing gap between processor and memory speeds,
- power consumption scales with clock frequency; so, at some point, extraordinary means are needed to cool the device.

A. Motivation

Using multiple processor cores on a single chip allows designers to meet performance goals without using the maximum operating frequency. They can select a frequency in the sweet spot of a process technology that results in lower power consumption. Overall performance is achieved with cores having simplified pipeline architectures relative to an equivalent single core solution. Multiple instances of the core in the device result in dramatic increases in the MIPS-per-watt performance [1].

The introduction of multicore processors provides a new challenge for software developers, who must now master the programming techniques necessary to fully exploit multicore processing potential. On a single-core processor, separate *tasks* share the same processor (i.e., its time). On a multicore processor, multiple tasks can run in paralel (i.e. they can be executed at the same time by the corresponding cores), resulting in more efficient execution.

B. Basic Terms, Principles and Problems

One of the first steps in mapping an application to a multicore processor is to identify the task parallelism and select a processing model that fits best.

1) Parallel Processing Models: According to [1], two dominant models exist:

- *Master/Slave*, i.e., centralized control with distributed execution. A master core is responsible for scheduling various executions that can be allocated to any available (slave-)core for processing. It also must deliver any data to a slave. Applications (e.g., multi-user data link layer of a communication protocol stack) that fit this model i) inherently consist of many small independent tasks that fit easily within the processing resources of a single core and ii) often run on a high-level OS like Linux (i.e., the master in charge of the scheduling) and potentially already have multiple execution to be done. Typically, task assignment is achieved by message-passing between the (master and slave) cores – the messages provide the control triggers to begin a task execution and pointers to the task's data.

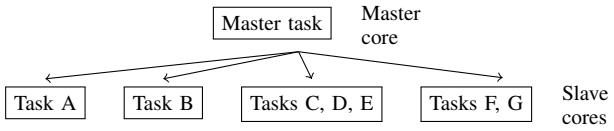


Figure 1. Master/Slave parallel processing model.

- *Data Flow*, i.e., distributed control and execution. Each core processes a block of data using various algorithms; then, the data can be passed to another core for further processing. The initial core is often connected to an input interface supplying the initial data for processing. Scheduling is triggered upon data availability. Applications (such as the physical layer of a communication

protocol stack) that fit the model often contain large and computationally complex components that are dependent on each other and may not fit on a single core. They likely run on a real-time (RT) OS where minimizing latency is critical. The challenge for applications using this model is partitioning the complex components across cores and the high data flow rate through the system. Components often need to be split and mapped to multiple cores to keep the processing pipeline flowing regularly. The high data rate requires good memory bandwidth between cores. The data movement between cores is regular and low latency hand-offs are critical. Synchronization of execution is achieved using message passing between cores. Data is passed between cores using shared memory or DMA transfers.

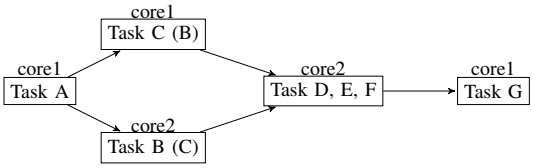


Figure 2. Data Flow parallel processing model.

2) Software Decomposition for Multicores: To design a software to be executed on a multicore platform, the following problems must be solved (typically, the solution must be searched in several cycles to find a (sub)optimal result):

- *Partitioning*, i.e., to identify a large number of small tasks (that are paralelizable, i.e. with low coupling and high cohesion) in order to yield a fine-grained decomposition; in the ideal situation, the tasks belonging to different partitions are intended to execute concurrently,
- *Dependency*, i.e., to identify (partitioned) tasks that are not independent; those must be mutually synchronized (serialized) to guarantee the dependencies, so their concurency is limited. Metrics are typically utilized to evaluate the level of cohesion in order to assist the grouping of tasks for minimizing dependency effects,
- *Combination*, i.e., to reflect information about partitions/dependencies for deciding about grouping of tasks so that they can be efficiently executed on a multicore,
- *Mapping*, i.e., to assign (potentially grouped) tasks to particular cores on basis of the selected parallel processing model (see I-B1 on p. 1). After all the tasks are mapped, the overall loading of each core can be evaluated to indicate areas for additional refactoring to balance the processing load across cores. Alike, further parameters such as message passing latency or worst-case blocking time due to the most pessimistic intra&inter-core synchronization/communication scenario(s) can be evaluated as well.

II. State-of-the-Art

A. Representatives of Existing Works

On basis of problems being solved, existing works can be divided into several groups. Basic analysis/summary of effects w.r.t. synchronization/communication on multicores (such as performance, speedup and scaling from Amdahl's law perspective) can be found in [2].

On top of that, particular works can be found, each dealing with a special area of interest w.r.t. multicores such as [3], dealing with the load balancing, or [4], trying to contribute to the synchronization problem by proposing a hardware lock implementation (so-called the lock arbiter herein) designed to reduce the lock latency while minimizing hardware overheads and maintaining high levels of fairness. In software, mechanisms such as resource access protocols [5], [6] are designed to prevent undesired effects such as priority/deadlock inversion, chained blocking and/or deadlock. Traditionally, the problem of clock synchronization must be solved too [7].

Many works deal with power management and issues [8] and solutions to the scheduling problem for multicores, being utilized to construct applications being critical somehow such as time-critical (or, real-time) applications. It should be noted there that the problem of scheduling (real-time tasks on a multicore processor) is the same as that of scheduling on a multi- processor system, i.e. an NP-hard problem – its solution can be approximated by heuristics. Basically, these heuristics can be divided into two categories, so-called *partitioning schemas* (*policies*) too [5]:

- *partitioned* being constructed to assign tasks to cores so that a task is going to be executed just by the core being assigned to the task,
- *global* allowing a task to be executed by different cores, depending on actual parameters of a system such as the computational load.

After the partitioning of all tasks is completed, tasks in each core can be scheduled using well-known mechanisms [6] or their multicore variants [5]. Due to their simplicity and efficiency, partitioned scheduling algorithms are generally preferred over global scheduling algorithms.

B. Work Done in MSc Thesis

The MSc Thesis [9] is focused on *asymmetric multiprocessing* (*AMP*) on the ARM Cortex–A9 MPcore platform. The *AMP* is an approach to computer system load distribution among heterogeneous software or hardware environments. In the thesis, there are heterogeneous software environments (see the figure below). Two equal processor cores run Linux, but when needed, the second processor core (slave) is run–time given to a bare–metal application by the first processor core (master).

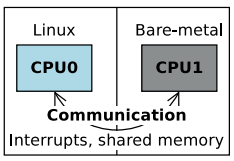


Figure 3. MSc Thesis *AMP* system.

The *AMP* was required to run on an Altera Cyclone V platform. It was discovered that an OpenAMP framework [10] that provides required functionality exists for Xilinx

Zynq platform that also contains the ARM Cortex–A9 MPcore processor. Thus the porting process from Xilinx Zynq to Altera Cyclone V had begun:

- The OpenAMP was running on PetaLinux on the Zynq and was ported to xilinx–2014.4 Linux.
- Later, the OpenAMP was ported to Altera Cyclone V hardware platform. Changes were made to OpenAMP's bare–metal libraries, system memory map was reorganized and missing loadable kernel modules for Altera's linux–socfpga were derived from Xilinx's xilinx–2014.4 Linux.

OpenAMP is now functional on the Altera Cyclone V platform although some work needs to be done in stabilizing the port.

III. PhD Thesis Details

Although many works have already dealt with the topics mentioned in II-A on p. 2, there are still many problems to solve e.g. in the area of studying inter-relationships among various parts of a multicore system. Particularly, it is necessary to study impacts of undesired events (such as a fault, error or failure, performance variations, overheating etc. of a component in the system) to parameters and behavior of the system.

For that purpose, appropriate instruments and techniques must be utilized (for more information, see III-D on p. 4, please). Those must be capable to i) describe behavior as well as attributes of a system and its components, their dynamics and ii) analyze properties of the system. Although it is planned to focus mainly to digital (discrete) systems, it would be advantageous if analog systems would be covered by the instruments too; this gives one an opportunity to model and analyze e.g. mixed/hybrid (i.e., digital/analog) systems – such as *cyber-physical* or *mixed-signal* systems – being widely utilized in practice.

A. Research Directions

1) Directions: There are many directions for the research w.r.t. multicores; we have limited their list to – modeling, analysis and/or design of – the following ones we plan to focus on:

- Cores, interconnections and topology,
- clock, memory and I/O subsystems,
- power and dependability issues,
- synchronization/communication primitives and mechanisms,
- policies for partitioning tasks, scheduling tasks and communications.

B. Research Questions and Hypothesis

On basis of our previous activity, many problems have been identified w.r.t. the multicore area. Since we intend to address some of the problems in our research, we have prepared a (preliminary, work in progress) list of *research questions* we would like to answer:

- Is it possible to build a credible and valid model of a generalized multicore platform ?
- Can the (above-mentioned) model be utilized to facilitate analysis, design and/or portability of platform-dependent routines for construction of a preemptive OS ?
- Can the (above-mentioned) model be utilized to facilitate analysis, design and/or portability of (power-, time-, safety- etc.) constrained, OS-controlled applications ?

On basis of the questions, our *research hypothesis* can be formulated as follows:

> **It is possible to build a credible and valid model of a generalized multicore platform on the basis of which analysis, design and/or portability of a software for multicores can be facilitated.**

C. Research Goals and Roadmap

1) Goals: Their list includes, but is not limited to

- scalable multicore solutions of problems related to task/ISR-level context switching, mutually exclusive access and task/kernel-level synchronization, task scheduling with potential core affinity/migration (i.e., with partitioned/global policy support),
- techniques to achieve robustness of an operating system and consistency of its data structures such as a task control block (TCB) and its adaptation to changes in a multicore platform,
- scheduling policies able to meet multiple task constraints (posed on time, power, safety etc.) under various fault/load scenarios,
- selection of comparison basis (such as benchmarks) and of proper methods and instruments to verify crucial properties (such as thread-safe or deadlock-free operation, worst-case latencies of system calls etc.) of the proposed concepts under AMP and/or SMP scenarios as well as to show their practical applicability using several case-studies of recent platforms and operating systems.

2) Roadmap: Basic blocks of the expected roadmap is depicted below. Estimation of the overall time (in months) reserved for a particular block is indicated in the circle by the block. Let it be noted there that activities w.r.t. particular blocks can overlap and be performed in parallel. The green blocks are almost completed, red ones not started yet and white ones have just started. Recent activity is highlighted using bold borders/arrows.

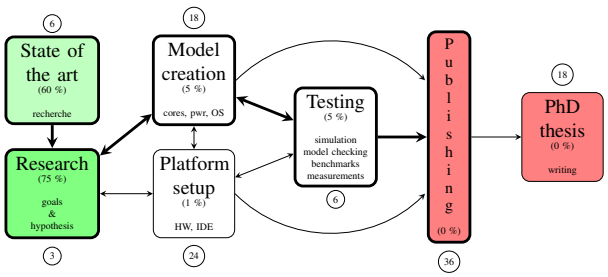


Figure 4. Research roadmap.

D. Methods and Instruments to Reach the Goals

To reach the goals outlined in III-C1, it is necessary to choose appropriate methods and instruments.

1) Modeling and Analysis Phase: Actually, we focus to the model creation and analysis phase, for which we have decided to apply the *stochastic timed automata* approach combined with *statistical model checking* (SMC) technique. Those instruments, available e.g. in the UPPAAL SMC tool [11] are able to facilitate the process of creation and analysis of models of dynamic systems e.g. by description and analysis of timing attributes and probabilistic behavior or their digital/analog/hybrid components and their parameters such as dependability or power consumption [12] [8]. The expected output of that phase is a generalized, parameterizable behavioral model of a multicore platform, consisting of key sub-models for the parts such as cores (i.e., computational elements able to execute a task), interconnections as well as interrupt, communication and memory subsystems including their parameters such as estimates of reliability, load, latencies and power consumption.

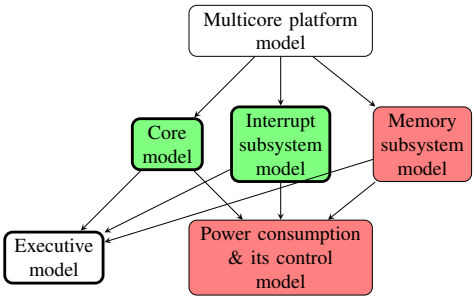


Figure 5. Generalized model of a multicore platform.

On top of the above-mentioned, it is necessary to create models of software layers such as an operating system or an application in order to study how much they are able to affect properties of a multicore system. This includes modeling and analysis of concepts such as tasks, partitioning/scheduling policies, their parameters and behavior within the context of timing, power, reliability, safety etc.

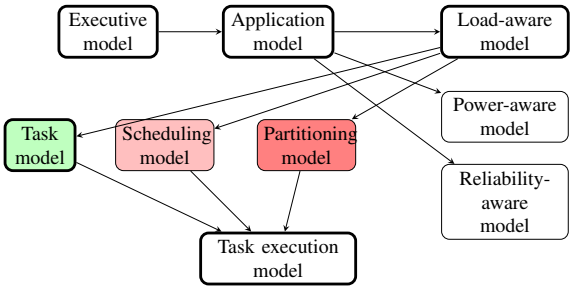


Figure 6. Top level models.

2) Model Validation Phase: To guarantee credibility and validity of models created within III-D1, the models must be made according to practical observations such as experiments over real multicore platforms. However, models for many components such as CPUs or memories and their parameters such as power consumption, load monitoring etc. which be created yet before those experiments are prepared and performed. Actually, the phase is under the preparation; later, it will overlap with III-D1. It is planned that properties of our models (such as deadlock-free operation, liveness, safety, timeliness and reliability) are going to be intensively analyzed by a technique such as SMC [11].

3) Evaluation Phase: In this phase, we plan to be inspired by existing works such as [2] [4] [13] in order to create and/or choose an appropriate comparison basis for evaluation of our approach. In this phase, we plan to focus on proving or disproving our hypothesis stated in III-B, p. 3 and start to summarize the achieved results into the PhD thesis.

Acknowledgment

This work was supported by the Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project "IT4Innovations excellence in science – LQ1602" and the inner university project No. FIT-S-14-2297 (Architecture of parallel and embedded computer systems).

The authors would like to thank Mr. Josef Strnadel for PhD study supervision, Mr. Pavol Korček and Mr. Jan Viktorin for information and experience they have provided me.

References

[1] TI, "Multicore Programming Guide," Texas Instruments, Tech. Rep., 2012, http://www.ti.com/lit/an/sprab27b/sprab27b.pdf.

[2] L. Yavits, A. Morad, and R. Ginosar, "The Effect of Communication and Synchronization on Amdahl's law in Multicore Systems," *Parallel Computing*, vol. 40, no. 1, pp. 1 – 16, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167819113001324

[3] K.-M. Cho, C.-W. Tsai, Y.-S. Chiu, and C.-S. Yang, "A High Performance Load Balance Strategy for Real-Time Multicore Systems," *TheScientificWorldJournal*, vol. 2014, p. 101529, 2014, DOI: 10.1155/2014/101529. [Online]. Available: http://europepmc.org/articles/PMC4009124

[4] R. Harding, "Synchronization on multicore architectures," Master's thesis, Carnegie Mellon University, 2010.

[5] S. Baruah, M. Bertogna, and G. Buttazzo, *Multiprocessor Scheduling for Real-Time Systems*, ser. Embedded Systems. Springer International Publishing, 2015. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-08696-5

[6] F. Cottet, J. Delacroix, C. Kaiser, and Z. Mammeri, *Scheduling in Real-Time Systems*. Hoboken NJ, United States: John Wiley & Sons, 2002.

[7] L. Lamport, "Time, Clocks, and the Ordering of Events in a Distributed System," *Commun. ACM*, vol. 21, no. 7, pp. 558–565, Jul. 1978. [Online]. Available: http://doi.acm.org/10.1145/359545.359563

[8] S. Pagani, J. J. Chen, and M. Li, "Energy Efficiency on Multi-Core Architectures with Multiple Voltage Islands," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 6, pp. 1608–1621, 2015, DOI: 10.1109/TPDS.2014.2323260.

[9] M. Riša, "Asymmetric Multiprocessing on the ARM Cortex-A9," Master's thesis, Brno University of Technology, 2015.

[10] Home page of the Open Asymmetric Multi Processing (OpenAMP) framework project. [Online]. Available: https://github.com/OpenAMP/open-amp

[11] A. David, K. Larsen, A. Legay, M. Mikučionis, and D. Poulsen, "Uppaal smc tutorial," *International Journal on Software Tools for Technology Transfer*, vol. 17, no. 4, pp. 397–415, 2015. [Online]. Available: http://dx.doi.org/10.1007/s10009-014-0361-y

[12] W. Dargie, "A Stochastic Model for Estimating the Power Consumption of a Processor," *IEEE Transactions on Computers*, vol. 64, no. 5, pp. 1311–1322, 2015, DOI: 10.1109/TC.2014.2315629.

[13] J. Mistry, M. Naylor, and J. Woodcock, "Adapting freertos for multicores: An experience report," *Softw. Pract. Exper.*, vol. 44, no. 9, pp. 1129–1154, Sep. 2014. [Online]. Available: http://dx.doi.org/10.1002/spe.2188

Zisk a zpracování hyperspektrálních dat

Jiří Čech
1. ročník, prezenční studium
Školitel: Zdeněk Plíva

Technická univerzita v Liberci,
Studentská 1402/2, 461 17 Liberec, ČR
jiri.cech@tul.cz

***Abstrakt*—Moje práce je součástí projektu vývoje hyperspektrálního detekčního systému (HDES), který si klade za cíl bezkontaktní detekci řady škodlivých plynů. HDES je složen ze speciálního objektivu a infračervené kamery vybavené výkonnou řídící jednotkou. Zaměřuji se na kompletaci a validaci dat potřebných k natrénování neuronové sítě a její implementaci do systému HDES. Aby šlo ze získaných dat rozpoznávat jednotlivé plyny je zapotřebí co nejvíce citlivý systém a proměřené charakteristiky všech jeho částí pro následnou korekci získaných dat.**

Klíčová slova—Mikrobolometr, hyperspektrální zpracování, infračervené zobrazování, FPGA zpracování

I. Úvod

Hyperspektrální kamery jsou hojně používané pro dálková měření, spektroskopii, armádní účely či různá teplotní měření. Využívají čipy s integrovaným mikrobolometrickým polem (FPA), které je schopné zachycovat určitou část spektra elektromagnetického záření a speciální optiku schopnou snímat scénu v různých úzkých částech z celého měřeného rozsahu. Nejpoužívanější oblastí elektromagnetického spektra je infračervené pásmo (IR, záření o vlnové délce od 0,7μm do 1000μm). FPA existují ve výrobních variantách CCD i CMOS a díky použití různých kombinací materiálů jsou schopné snímat části elektromagnetického spektra od blízkého IR (NIR) až po vzdálené IR (FIR).

Hyperspektrální kamera zaznamenává sadu obrázků, které se složí do tzv. Hyperspektrální kostky. Tu reprezentují dvě dimenze obrazu a třetí dimenzí je intenzita bodu obrazu v různých vlnových délkách. Používají se dva způsoby záznamu dat, buď měníme před kamerou úzkopásmový filtr, nebo snímáme pouze řádek scény a na FPA snímáme jeho rozklad do několika vlnových délek, takto po řádcích nasnímáme celou scénu. Pokud je hyperspektrální informace sbírána jen z několika bodů spektra označujeme ji jako multispektrální. Ty jsou využívané zejména v průmyslu, kde je potřeba detekovat jen jednu látku, k čemuž stačí jen vhodně zvolených pár bodů spektra. Hyperspektrální kamery snímají spektrum v desítkách, až stovkách bodů a dokáží tak rozlišit velké množství materiálů. Doba pořízení jedné hyperspektrální kostky je závislá na velikosti a rychlosti senzoru, způsobu záznamu a počtu zaznamenávaných vlnových délek, pohybuje se v řádu stovek milisekund až jednotek sekund.

Největší aktuální pokroky ve zpracování hyperspektrálních dat jsou zaměřeny na témata jako je zisk a komprese dat ze senzoru [8] (prostorové aplikace), nestejnoměrné korekce [5],[6],[7] (ve všech aplikacích) a analýza hyperspektrální kostky [4],[8] (analýza spektra). Všechna zmíněná témata mají vysoké výpočetní nároky při použití výpočtu na CPU. Řešením je používání výpočtů na výkonné grafické kartě (GPGPU) nebo na programovatelném hradlovém poli [3],[6] (FPGA). Využití FPGA má oproti ostatním výhody ve vyšší robustnosti výpočtu, nižší spotřebě a možnosti přeprogramování.

Dnešní FPGA jsou programovatelná zařízení složená z mnoha samostatných propojitelných a programovatelných bloků, pomocí kterých se realizují uživatelské návrhy (IP). Některá IP jsou v moderních FPGA přímo zaimplementované s malou nebo žádnou možností přenastavení. Jedná se o paměťové řadiče, rozhraní PCIe, Gbit Ethernet, nebo další komplexní podsystémy [9]. Největší výhodou FPGA využívající SRAM (statická RAM), je načítání konfigurace bloků a propojovacích struktur z externí paměti během spouštění systému, kterou je možné kdykoliv přepsat. Tato funkce je vhodná pro samoopravovací algoritmy (ve vesmírných aplikacích) nebo pro změnu části či celého sytému během jeho běhu, pomocí techniky dynamické (částečné) rekonfigurace.

Skupina nechlazených FPA schopných detekovat dlouhovlnné infračervené záření (LWIR 8-15 μm) [7],[10] byla využívaná výhradně v armádních aplikacích. Nyní se díky pokrokům ve výrobních procesech a snížení jejich ceny výrazně rozrůstá i do běžné komerce. Dostupná jsou rozlišení od QVGA (320×240) do SVGA (1024×768) a rychlosti se pohybují od několika snímků za sekundu až po desítky snímků za sekundu (výkonnější FPA jsou určena pouze pro armádu a to jen ve vybraných zemích). Využití mají v komerčních oblastech jako je kontrola a detekce problémů elektrických, mechanických a tepelných systémů, zjišťování vlivu elektrické zátěže, odhalování nesprávné instalace systémů, nebo průmyslová spektroskopie.

II. Motivace

Moje práce je součástí probíhajícího projektu vývoje hyperspektrální kamery firmou APPLIC ve spolupráci s Technickou Univerzitou v Liberci a Centra speciální optiky a optoelektronických systémů TOPTEC. Cílem projektu je

vytvořit levné spektroskopické ruční zařízení pro použití v bezpečnostních aplikacích (HDES) a ořezanou verzi systému pro průmyslové termometrické využití (IRCA). Hyperspektrální kamera využívá procesní platformu Zynq [9], která ovládá a zaznamenává data z mikrobolometrického LWIR senzoru ULIS PICO640E [10] také umožňuje různé další operace, které jsou důležité pro zpracování získaných dat. Surové naměřené hyperspektrální kostky je potřeba upravit korekcemi charakteristik kamery a optiky. Jedná se o korekce vadných pixelů, nelinearit, teplotní citlivosti a disperze obrazu. Upravená data se budou vyhodnocovat na FPGA v systému HDES. To bude probíhat na principu neuronových sítí [4] z důvodů očekávaných kvalitnějších a rychlejších výsledků a jednodušší implementaci.

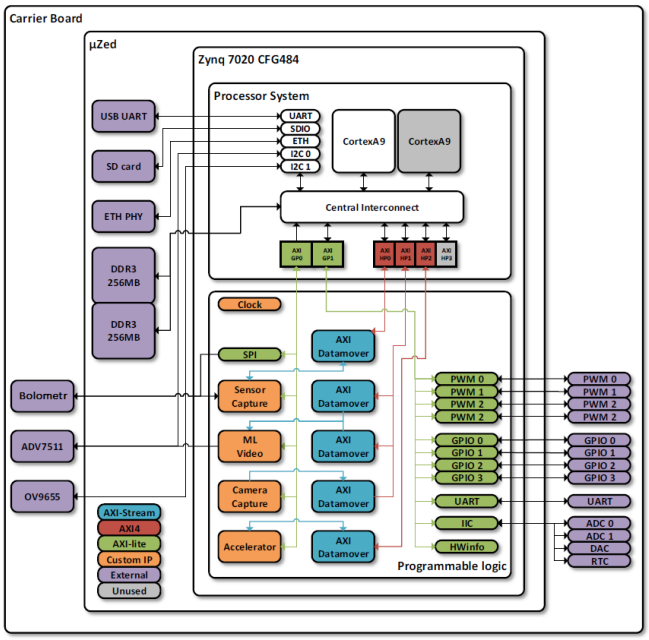
Motivací pro mou práci je nejenom dokončení projektu HDES, ale následné prozkoumání hyperspektrálních dat pořízených systémem HDES, které se liší od běžně dostupných dat pořízených satelity nebo letadly. Pomocí dostupných metod předzpracování a vyhodnocení [11] vytvořit z naměřených dat učící a testovací vzorky potřebné pro tvorbu neuronové sítě. Použít některé pokročilé metody učení neuronových sítí k jejímu natrénování („Deep learning“, „Extreme learning machine“[14]). Vytvořit vhodné předzpracování dat pro zvýšení přesnosti vyhodnocení. Na konec implementovat výsledný systém vyhodnocování do systému HDES a vytvořit vhodný systém interpretace výsledků. Za účelem zvýšení výkonu dokončit implementaci řízení celého systému pomocí operačního systému Linux. Vytvořit zobrazovací a nastavovací rozhraní pro snadnou obsluhu a možné laboratorní využití.

Protože se jedná o bezbarvé plyny, je potřeba navrhnout jak ověřit a následně i otestovat, zda metody tvorby učících vzorků dávají správné výsledky. Z možných řešení předzpracování dat prozkoumat opravu optických vad systému, výběr dat nezkreslených korekcemi, kompenzaci vlivu koncentrace dané látky na její rozpoznávání. Vyhodnocením zaznamenané hyperkostky bude procentuální obsažení látek a jejich odhadované koncentrace.

III. Popis systému HDES

Řídící jednotkou pro systém HDES (Obr. 1) je Zynq 7020, což je programovatelné zařízení složené ze dvou samostatných částí: procesního systému (PS) a programovatelné logiky (PL). PS je vícejádrový SoC (systém na čipu) obsahující dva 32bit procesory ARM Cortex A9, vestavěné řadiče (IIC, SPI, 1Gbit Ethernet, USB, atd.), systémové paměti DDR3 a IP vytvořená v PL, které vychází z rodiny Artix-7 (nízkonákladové produkty 7 série Xilinx FPGA s nízkou spotřebou).

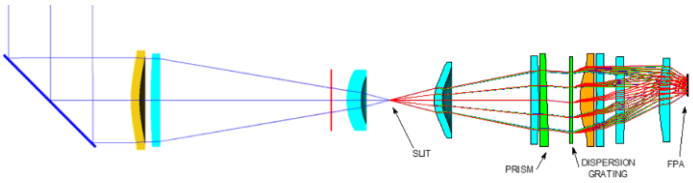
Pro ovládání FPA je použit speciální integrovaný monolitický AD/DA převodník. Ten umožňuje jeho ovládání přes SPI rozhraní a poskytuje z něj obraz přes digitální video port (DVP). Převodník jej obsluhuje analogově, vyžaduje přesné a bezšumové napájení a jeho výstupem je sériové čtení 14bit hodnot jednotlivých pixelů. Systém dále obsahuje několik uživatelských IP a další rozhraní jako je HDMI/VGA výstup, VGA kameru (viditelné spektrum) a několik PWM



Obr. 1. Blokové schéma systému HDES [13]

výstupů pro ovládání objektivu a clonky. Systémové zprávy jsou posílány po USB-UART rozhraní, ovládání a data jsou posílány přes 1Gbit Ethernet protokolem TCP/IP do uživatelského počítače. Kamera umožňuje ukládání dat na vnitřní SD kartu, ve které je uložen i její hlavní program.

Hyperspektrální objektiv kamery (Obr. 2) je navržen jako



Obr. 2. Zjednodušené schéma objektivu HDES [13]

úzká štěrbina s mřížkou, tím dosahuje vysoké propustnosti záření. Scéna se snímá po sloupcích, přičemž v každém snímku každého sloupce je zaznamenána jeho spektrální charakteristika rozprostřená na více než 280 pixelů, to umožňuje dosáhnout maximálního spektrálního rozlišení 1,6 cm^{-1}. Skutečné rozlišení je silně závislé na tloušťce štěrbiny a odstupu signálu od šumu (SNR) kamery. Optika je vybavena otočným zrcadlem, pomocí kterého může nasnímat celou scénu (tzv. „push broom“ metodou). Systém dosahuje velikosti zorného pole (FOV) v horizontálním směru 40° a ve vertikálním 10°. Dopadající záření v různých spektrech na FPA se konvertuje na digitální data následovně:

- Elektromagnetické záření dopadající na senzor mění odpor jednotlivých mikrobolometrů.
- Měnící se proud mikrobolometrickou buňkou je převáděn na napětí a zesilován transimpedančním operačním zesilovačem.
- Přečtené napětí z matice mikrobolometrů je vzorkováno a kvantováno vstupem AD převodníku.

- Data z DVP (všechny hodnoty obrazu za sebou) jsou ukládána do DDR pamětí. Během tohoto kroku je možné provést korekci nasnímaných dat.
- Poté se data odešlou do počítače, kde je možné je složit do hyperspektrální kostky a dále zpracovávat.

Pro systém HDES existují dvě verze řídícího firmware. Ten aktuálně používaný využívá knihoven dodávaných výrobcem k dané verzi desky (BSP) od Xilinx a rozšířené o knihovny pro TCP/IP komunikaci a konverzi obrázků. Ovladače pro uživatelská IP jádra jsou vyvíjeny samostatně. Druhý firmware je experimentální a je na bázi operačního systému Linux vytvořeného nástrojem „PetaLinux tool“. Je vyvíjen za účelem dosažení vyššího výkonu a lepší podpory nových rozšíření systému, ale zatím podporuje jen základní funkce kamery.

IV. Zpracování získaných dat

V důsledku odchylek při výrobním procesu má každý mikrobolometr ve FPA rozdílnou charakteristiku tepelné citlivosti a tepelného odporu [7]. Pixel je klasifikován jako vadný, pokud se jeho citlivost liší o více než 20% oproti průměrné citlivosti celého FPA. Systém HDES provádí korekci vadných pixelů tak, že místo vadného pixelu používá data pixelu předním. Tato korekce je při sériovém čtení velmi snadno implementovatelná a dobře funguje, i pokud je více vadných pixelů vedle sebe. Jiná použitelná metoda vypočítává hodnotu vadného pixelu na základě jeho okolí, ale její implementace zatím nebyla realizována.

Existuje několik přístupů ke korekci rozdílů mezi mikrobolometry a jejich nelinearity (NUC). Jednobodová NUC vyrovná posunutí pro celé FPA do jednoho pracovního bodu. Důsledkem teplotních změn FPA [1],[7] dochází k posunutí pracovního bodu, na který byla korekce počítána a tudíž i k jejímu znehodnocení. To vede k nutnosti opakování provedení korekce, které je v průběhu měření nevhodné. Dvoubodová NUC [1],[5],[6] využívá měření s černým tělesem o dvou známých teplotách, ze kterého se vypočítá posun a zesílení každého mikrobolometru. Jedná se o nejpoužívanější korekční metodu a je vhodná pro menší teplotní rozsahy (do 100 °C). Pro větší dynamické rozsahy dochází u dvoubodové korekce k velkým odchylkám.

Řešením je použití vícebodové NUC [3] vytvořené z několika lineárních korekcí napříč celým rozsahem. Zde je problémem, definování kolik lineárních korekcí je dostatečných, nepřesnosti na okrajích mezi nimi a paměťová náročnost při jejich velkém množství. Výkonná FPGA dávají možnost použití polynomiální NUC vyšších řádů, která dosahuje nízkých chyb napříč celým rozsahem. Měří se stejně jako vícebodová NUC tzn. proměřit celý rozsah senzoru v jemném kroku. Naměřená data se pro každý mikrobolometr aproximují polynomem n-tého řádu, minimalizuje se relativní odchylka od naměřených dat. Koeficienty polynomu jsou poté uloženy v paměti a nahrávají se postupně do korekčního modulu v FPGA při čtení daného pixelu. FPGA použité v systému HDES zvládne v reálném čase polynomiální NUC až pátého řádu. Naměřená data uložená v hyperkostce potřebují korekci chyb zanesených objektivem, jedná se hlavně o nerovnoměrné zesílení ve směru od optického centra rozdílnou pro každý úhel natočení zrcadla.

Výsledkem analýzy hyperspektrálních dat je rozhodnutí kde je jaký matriál. Obecně se při vyhodnocování uvažuje, že jeden pixel reprezentuje pouze jeden materiál. Informace o materiálu jsou obsaženy v jeho spektrální charakteristice, díky tomu, že každý materiál pohlcuje, vyzařuje nebo odráží záření v různých vlnových délkách. Nejvyšších hodnot dosahuje tzv. černé těleso, které pohltí veškeré dopadající záření a vyzáří maximální množství energie v závislosti na teplotě podle Planckova vyzařujícího zákona. Ten definuje průběh intenzity záření černého tělesa o dané teplotě napříč celým spektrem. Pro porovnání s ostatními materiály se definuje tzv. Emisivita jako poměr intenzity záření reálného tělesa a absolutně černého tělesa. Tato bezrozměrná veličina je závislá na vlnové délce a není nikdy vyšší než 1.

Hyperspektrální data jsou často velmi objemná, proto se řeší jejich de/komprese, popřípadě odstranění nepotřebných spektrálních pásem. Existují databáze naměřených spektrálních charakteristik jednotlivých materiálů, ale reálná data jsou zatížena vlivem pozadí, šumem a odrazy okolí. Proto se místo přímého porovnávání s laboratorními vzorky nejprve vyhledají reprezentativní průběhy [2], které jsou nejvíce rozdílné od ostatních a představují tak nejlépe některý z obsažených materiálů. Nejpoužívanější metoda uvažuje lineární závislost mezi ostatními porovnávanými a reprezentativními průběhy. Metodou nejmenších čtverců se vypočítá podobnost průběhu s každým reprezentativním průběhem. Vyhodnocovaný průběh pak patří do skupiny s reprezentativním průběhem, u kterého má nejnižší spočtenou hodnotu.

Dostupné reprezentativní průběhy se uvažují jako lineárně nezávislé, tudíž při vyhodnocování jednoho materiálu jsou výsledky zatíženy pouze šumem. Pokud jich vyhodnocujeme více, jsou problémy na okrajích a přechodech materiálů, protože dochází k jejich prolínání. Výsledky pak mají více podobných nízkých hodnot v závislosti na poměru prolínaných materiálů. Je tedy potřeba stanovit rozpoznávací mez pro každou látku, nebo uvažovat závislosti nelineární.

Hyperspektrální kostka obvykle obsahuje jen malé množství pixelů reprezentující jen jednu látku. Většinou je pixel složen z jednoho či více materiálů souhrnně označovaných jako pozadí. To se dá rozložit na lineární kombinaci hledaného materiálu a pozadí s tím, že vždy budeme mít všude nějaký šum. Pozadí můžeme definovat buďto statistickou distribucí, nebo strukturálně [12].

V. Dosažené výsledky

Byl sestaven první prototyp systému HDES, obsahující první funkční verzi objektivu od TOPTEC a infrakameru IRCA1 od firmy APPLIC, která disponuje mnoha rozšířeními, které zatím nejsou využity. K systému jsem vytvořil ovládací program, přes který lze ladit jednotlivé jeho funkce. Program umožňuje mnoho typů záznamu a zobrazení dat, které jsou pak dále využívány a zkoumány. Vyvíjený firmware systému od TUL, má odladěné všechny důležité funkce a v návaznosti na mé testování, byly některé funkce přidány nebo modifikovány.

Ze systému HDES byla úspěšně zaznamenána první data, která odhalila velmi nízkou energetickou citlivost celého systému. Pracuje se tedy nyní na jejím zvýšení, aby bylo možné naměřit lepší data potřebná pro vytvoření kvalitního identifikátoru daného souboru látek. Ze strany TOPTECu se jedná o vytvoření nového objektivu s koncentrovanějšími paprsky na úkor spektrálního rozlišení. Ze strany APPLICu probíhají pokusy na snížení šumu záznamového systému v IRCA1.

Zatím pracuji s jedním vzorkem úspěšně zaznamenané hyperspektrální kostky. Testuji na něm vyhodnocovací skript v MATLABu, jehož cílem je extrakce tréninkových dat pro učení neuronových sítí. Je založen na výpočetních metodách obsažených v dostupném balíku HyperspectralToolbox [11]. Provádí normalizaci dat, extrakci vzorových dat na základě rozdílů spektrální charakteristiky (ATGP) a následně identifikaci materiálu pomocí metody nejmenších čtverců (UCLS). Data zatím neprochází žádnou korekcí, protože vlivem vysokého šumu nepřináší žádné velké zlepšení.

Vyhodnocená data slouží pro experimentální tvorbu neuronových sítí. Jejich tvorbu provádím pomocí nástroje TORCH využívající jazyka LUA, snadno editovatelného ve studiu ZeroBrane. Vytvořil jsem skripty umožňující načtení vyhodnocených dat, učení a testování neuronové sítě nastavených velikostí. Zatím bylo dosaženo nejlepší shody vyhodnocených dat pouhých 60%. Neimplementoval jsem ještě některé metody pro zlepšení dosažených výsledků, např. pomocí zahrnutí okolních bodů, které znatelně zvyšují úspěšnost vyhodnocení. Dalším důvodem nízké schody je zmíněný vysoký šum, díky čemuž jsou rozdíly mezi jednotlivými výstupními skupinami minimální.

VI. Cíle

Cílem mé práce je vytvoření učící databáze pro tvorbu vyhodnocovacích algoritmů pomocí systému HDES. Natrénovat neuronovou síť s dostatečnými vlastnostmi pro použití a realizovat jí pomocí dostupného FPGA. Díky otevřenému přístupu k celému systému HDES lze zkoumat všechny jeho části odděleně, či je vylepšit za účelem zvýšení citlivosti systému.

Systém HDES by měl být přenosný systém pro použití v terénních podmínkách schopný identifikovat řadu škodlivých plynů. Systém musí být schopen velmi rychle data zaznamenat a poté je pomocí kvalitního identifikátoru co nejrychleji vyhodnotit, aby měly pozemní zásahové složky aktuální naměřené hodnoty k dispozici co nejdříve.

VII. Závěr

Systém HDES je založen na moderních technologiích a klade si náročné cíle, aby byl v praxi užitečný. Provádí se mnoho optických a elektrických testování všech jeho částí a pracuje se na řešení zjištěných nedostatků, nebo na jejich vylepšení.

Podílím se na dokončení systému, aby byl schopen měřit věrohodná data v dostatečné kvalitě pro experimenty s možnými identifikátory. Připravuji a zkouším využití neuronových sítí pro vyhodnocování hyperspektrálních dat, které zatím nedosahuje kvalitních výsledků. Příčinou je vysoký šum systému a očekávají se komplikace s reálným pozadím, které je na rozdíl od leteckých nebo satelitních snímků velmi různorodé. Po dokončení experimentů a natrénování použitelné neuronové sítě, implementuji vyhodnocovací systém na FPGA. Dočasně je záznam a vyhodnocení řízené přes PC, ve finální verzi HDES se počítá s řízením pomocí firmware, ale v budoucnu pro zvýšení výkonu bude firmware nahrazen operačním systémem Linux.

Poděkování

Tato práce byla podpořena Studentskou Grantovou Soutěží 2016 Technické Univerzity v Liberci. Dále byla podpořena firmou APPLIC s.r.o., Centrem speciální optiky a optoelektronických systémů TOPTEC a Ministerstvem vnitra České Republiky v rámci projektu číslo VG20132015110.

Literatura

[1] Farhat,H., Bazin,E., Haese,S., El Zein, G., „An infrared thermal image acquisition system for intra-vehicle applications“, 11th International Conference on ITS Telecommunications, ITST 2011, pp. 426-430, 2011

[2] Keshava, N. Mustard, J.F., „Spectral unmixing“, IEEE Signal Processing Magazine, vol. 19, no. 1, pp. 44-57., 2002

[3] S. Lopez, T. Vladimirova, C. Gonzalez, J. Resano, D. Mozos and A. Plaza, „The Promise of Reconfigurable Computing for Hyperspectral Imaging Onboard Systems: A Review and Trends,” in Proceedings of the IEEE, vol. 101, no. 3, pp. 698-722, March 2013.

[4] Melgani,F., Bruzzone,L., „Classification of hyperspectral remote sensing images with support vector machines“, IEEE Transactions on Geoscience and Remote Sensing, vol 42, pp. 1778-1790, 2004

[5] Mudau,A.E. and Willers,C.J. and Griffith,D., Le Roux,F.P.J., „Non-uniformity correction and bad pixel replacement on LWIR and MWIR images“, Saudi International Electronics, Communications and Photonics Conference 2011, SIECPC 2011, 2011

[6] Sosnowski,T., Bieszczad,G., Kastek,M., Madura,H., „Processing of the image from infrared focal plane array using FPGA-based system“, Proceedings of the 17th International Conference - Mixed Design of Integrated Circuits and Systems, MIXDES 2010, pp. 581-586, 2010

[7] Tissot,J.L., Trouilleau,C., Fieque,B., Crastes,A., Legras,O., „Uncooled microbolometer detector: Recent developments at ULIS”, Opto-electronics Review, vol. 14., pp. 25-32, 2006

[8] G. Yu, T. Vladimirova, and M. N. Sweeting, „Image compression systems on board satellites“, Acta Astronautica, vol. 64, no. 9–10, pp. 988–1005, May 2009.

[9] Xilinx Inc., „Zynq-7000 All Programmable SoC“, online <http://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html>, 2016

[10] ULIS-IR, „Pico640E technical datasheet“, online <http://www.ulis-ir.com>, 2016

[11] Isaacgerg, „MATLAB Hyperspectral Toolbox“, online <https://github.com/isaacgerg/matlabHyperspectralToolbox>, April 2015

[12] Manolakis, Dimitris et al., „Hyperspectral Image Processing for Automatic Target Detection Applications.”, 2003

[13] Hampl V., „Hyperspektrální kamera“, interní dokumentace, Applic s.r.o, online <http://www.applic.cz>, 2015

[14] Huang, G. „An insight into extreme learning machines: Random neurons, random features and kernels. Cognitive Computation“, 6(3), 376-390. doi:10.1007/s12559-014-9255-2 , 2014

Logická syntéza s nativní podporou XOR hradel

Ivo Háleček
1. ročník, prezenční studium
Školitel: Petr Fišer, Specialista: Jan Schmidt

Fakulta informačních technologií, ČVUT
Thákurova 9, 160 00 Praha
ivo.halecek@fit.cvut.cz

***Abstrakt*—V článku je představena možnost využití nové reprezentace logických obvodů pomocí Xor-And-Invertor grafů (XAIG) v syntézních algoritmech. XAIG jsou založeny na And-Invertor grafech, orientovaných acyklických grafech, kde uzly představují dvou-vstupá hradla AND či XOR a hrany mohou být negované. Předpokládá se, že tato reprezentace umožní algoritmům s XORy nativně pracovat a odhalit komplexnější závislosti. Pro experimentální ověření pravdivosti bude reimplementován algoritmus rewrite a bude porovnán s originálním rewritingem.**

***Klíčová slova*—Logická syntéza, XOR, AIG, XAIG, ABC, rewriting**

I. Úvod

Zlepšování logické syntézy je stále aktuální téma. Přesto, že proces logické syntézy a optimalizace se zdál být již v minulých dekádách efektivně vyřešený problém, v nedávné době se objevilo několik nových přístupů, většinou založených na nových datových strukturách používaných pro reprezentaci funkcí a sítí [22].

Původní nástroje logické syntézy vyjadřovaly funkce pomocí sum-of-products (SOP) [1], [2]. Nad touto reprezentací pracovaly dvouúrovňové, později víceúrovňové SOP minimalizační nástroje. Jako jednodušší alternativa k reprezentaci v SOP byla v mnoha algoritmech využívána reprezentace pomocí NOR hradel [3], která však přinesla pouze jednodušší implementaci algoritmů.

Velkým průlomem v reprezentaci funkcí bylo představení binárních rozhodovacích diagramů (Binary Decision Diagrams - BDD) [4], [5]. Syntézní a optimalizační algoritmy byly upraveny pro tyto struktury, což zlepšilo jejich výkon [6], [7], [8], [9].

Tyto reprezentace, ačkoliv hojně využívané, však trpěly špatnou škálovatelností. Z tohoto důvodu vznikla velmi efektivní reprezentace logiky – And-Inverter-Grafy (AIG) [10], [11], [12]. V AIG je logická síť reprezentována pomocí orientovaného grafu, kde uzly jsou 2-vstupá AND hradla a hrany mohou být negované. Mnoho algoritmů, založených na reprezentaci v AIG bylo implementováno do moderního akademického nástroje pro logickou syntézu a verifikaci, ABC [13]. Reprezentace v AIG pravděpodobně je, či brzo bude integrována i do komerčních nástrojů [14].

V tomto článku představujeme rozšíření konceptu AIG o nativní podporu hradel XOR, Xor-And-Inver grafy (XAIG). XAIG představují ortogonální přístup k Majority-Invertor-Graphs (MIGs), což je reprezentace založená na AIG, kde uzly jsou nahrazeny majoritní funkcí 3 proměnných [23].

Reprezentace pomocí XAIG je přístup ortogonální k MIG. XOR není monotónní a použití XAIG může vést k jiným oblastem efektivity implementace než MIG.

II. Popis problému

Přesto, že se několik let považovalo AIG za univerzální reprezentaci pro logické obvody, algoritmy postavené na AIG vykazují u některých obvodů hluboce neoptimální výkon, produkující obvody s mnohem větší plochou, než jaké je možné docílit. Jedna ze společných charakteristik této problémové množiny obvodů je vysoká intenzita hradel typu XOR. Důvodem neschopnosti algoritmů správně využít tato hradla může být v tom, že XOR má jiné vlastnosti než AND nebo NOR. Pokud tedy algoritmus správně neidentifikuje XOR, a pracuje s ním jako se soustavou ANDů a invertorů, může vyprodukovat řádově horší výsledky, než pokud by byl schopný nativně pracovat s XORem [15], [16].

Pro podpoření tohoto tvrzení jsme vygenerovali ze sady benchmarků [17], [18], [19], [20] verzi zkolapsovanou do dvouúrovňového popisu (SOP). Tento proces zamaskuje XORy, což, v případě nedostatečné identifikace a využití XORů optimalizačními algoritmy, může způsobit výrazné zhoršení výsledků syntézy.

V nástroji ABC provedli experimentální porovnání sady benchmarků v originální a collapsed verzi pomocí 20 iterací sekvence příkazů *dch; if; mfs*. Tato sekvence obvod optimalizuje, namapuje do LUTů a optimalizuje neurčené hodnoty. Jak je vidět v Tabulce 1, collapse u některých obvodů způsobil, že syntéza je nedokázala zoptimalizovat.

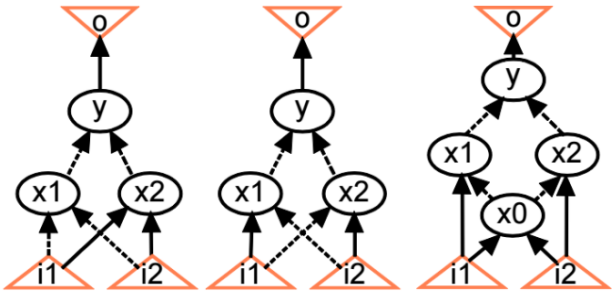
TABULKA 1: POROVNÁNÍ OPTIMALIZACE ORIGINÁLNÍCH A COLLAPSED BENCHMARKŮ

name	original LUTs	collapsed LUTs	LUT ratio
s4863	425	138111	324,97
mm9b	121	6153	50,85
c880	116	5627	48,51
c5315	475	19873	41,84
mm9a	98	2593	26,46
Altera_oc_minirisc	770	11512	14,95
c432	71	827	11,65
i10	665	6285	9,45
ITC_b05	174	1367	7,86
mm4a	40	304	7,60
Altera_oc_des_des3perf	17817	69983	3,93
Altera_oc	635	2431	3,83

III. NAVRHOVANÉ ŘEŠENÍ

Jedním ze způsobů, jak se pokusit zlepšit výsledky syntézy nad obvody s vysokou intenzitou hradel XOR je přepsat současné syntézní algoritmy tak, aby pracovaly s obecnější reprezentací obvodů a uměly XORy efektivně využít. Takovou reprezentací by mohlo být XAIG, rozšířené AIG tak, že uzly mohou mimo funkce AND představovat funkci XOR.

Jak lze vidět na Obrázku 1, XOR je v AIG reprezentován dvěma vstupy (i1,i2), dvěma či třemi vnitřními ANDy (x0, x1, x2) a výstupním ANDem (y), který může být invertovaný.



Obrázek 1: reprezentace hradla XOR v AIG

Náš nedávný výzkum ukázal, že nástroj ABC sice již obsahuje možnost reprezentovat obvody v podobné struktuře jako XAIG, And-Xor-Mux grafy. Tuto reprezentaci však využívá pouze podmnožina optimalizačních algoritmů z nového balíčku příkazů ABC9. Jelikož tyto nové algoritmy nejsou téměř dokumentované ani publikované, provedli jsme nejdříve výzkum toho, jak jsou tyto nové algoritmy schopny využít XORů v XAIG.

Nejprve jsme zjistili, které algoritmy využívají této nové reprezentace tak, že jsme po úpravě zdrojového kódu u algoritmů z balíčku ABC9 zaznamenávali volání funkce pro identifikaci XORů a jimi nahrazení ekvivalentních struktur v AIG. Zkoumali jsme pouze příkazy, u kterých jsme z nápovědy zjistili, že se týkají optimalizace (X)AIG. Výsledky jsou zobrazeny v Tabulce 2. Tyto algoritmy jsme vybrali pro porovnání jejich účinosti s modifikovanou verzí, která struktury v AIG XORy nenahrazovala. Místo vytvoření nativního XORu se vždy vytvořila funkčně ekvivalentní struktura z ANDů, jako na **Obrázek 1.**

TABULKA 2: OPTIMALIZAČNÍ ALGORITMY Z BALÍČKU ABC9 A VÝSLEDEK ZKOUMÁNÍ VYUŽITÍ XAIG (SLOUPEC XAIG)

příkaz	Oficiální popis	XAIG ?
&b	performs AIG balancing to reduce delay and area	Ano
&blut	performs AIG balancing for the given LUT size	Ano
&dc2	performs heavy rewriting of the AIG	Ne
&dch	computes structural choices using a new approach	Ne
&dsd	performs DSD-based collapsing	Ne
&dsdb	performs DSD balancing	Ne
&fadds	detects full-adder chains and puts them into white boxes	Ne
&flow	integration optimization and mapping flow	Ano
&flow2	integration optimization and mapping flow	Ano
&fraig	performs combinational SAT sweeping	Ne
&pack	performs packing for the LUT mapped network	Ne
&satclp	performs SAT based collapsing	Ano
&satfx	performs SAT based shared logic extraction	Ano
&st	performs structural hashing	Ne
&syn2	performs AIG optimization	Ano
&syn3	performs AIG optimization	Ano
&syn4	performs AIG optimization	Ano
&synch2	computes structural choices using a new approach	Ano

K porovnání jsme výsledky optimalizace namapovali na buňky LUT se 6 vstupy a sledovali jejich počet. V případě, že

by algoritmy uměly reprezentaci s XORy využít, měl by být počet LUTů po namapování menší u originálního algoritmu, proti algoritmu s výše uvedenou modifikací.

Jak je vidět v Tabulce 3, mezi výkonem algoritmu &syn4 nad AIG a XAIG je velmi silná korelace. Tabulka ukazuje pouze podmnožinu zkoumaných benchmarků a jeden vybraný optimalizační algoritmus, výsledky dalších algoritmů a kompletní sady benchmarků jsou velmi podobné; korelace je srovnatelná – dostupné algoritmy tedy s XORy pravděpodobně správně pracovat neumí.

TABULKA 3: SROVNÁNÍ ALGORITMU &SYN4 NAD AIG A XAIG

		LUTs	
name	ANDs	AIG	XAIG
s38417	8166	2247	2284
des	4123	482	561
apex4	2587	686	692
bca	2292	805	811
C6288	2290	893	703
bcb	2066	719	704
bcc	1972	677	676
apex1	1842	605	606
apex3	1561	444	452
bcd	1424	497	482
C5315	1393	250	286
frg2	1164	184	178
Correlation			**0,993**

IV. ZÁVĚR A PRÁCE DO BUDOUCNA

Jelikož výsledky experimentu přesvědčivě ukázaly na to, že optimalizační algoritmy ABC nedokáží obecnou strukturu XAIG využít, rozhodli jsme se reimplementovat algoritmus rewrite [12], který pro novou reprezentaci nebyl vůbec k dispozici.

Algoritmus rewrite (algoritmus popsaný v Zdrojovém kódu 1) identifikuje v obvodu maximální k-feasible řezy (cuts), podgrafy takové, že z jejich kořene vedou všechny cesty k primárním vstupům přes právě *k* listů tohoto podgrafu [21]. Řezy jsou konstruovány rekurzivně pro každý uzel dle algoritmu viz Zdrojový kód 2. Pro každý z nalezených řezů je simulací získáno pravdivostní ohodnocení. Pro k = 4 existuje takových ohodnocení 2^{16}, avšak permutací a negací vstupů lze všechna ohodnocení převést do 224 NPN ekvivalentních tříd. Předpočítané třídy ekvivalence pro k = 4 již jsou v ABC implementovány pro rewrite algoritmus pracující nad AIG, který jsme použili. Pro každou třídu je předpočítána optimální implementace a řez je nahrazen právě tou funkčně odpovídající jeho třídě, s příslušnou permutací a negací listů. Vnitřní uzly řezu s výstupy vedoucími mimo řez jsou před náhradou zduplikovány. Změna je přijata v případě, že se její aplikací počet uzlů grafu zmenší.

```
Rewriting (network AIG, hash table
PrecomputedStructures, bool UseZeroCost)
{
  for each node N in the AIG in topological order
  {
    for each 4-input cut C of node N computed using
cut enumeration {
      F = Boolean function of N in terms of the
leaves of C
      PossibleStructures =
HashTableLookup(PrecomputedStructures, F);
      // find the best logic structure for rewriting
      BestS = NULL; BestGainGain = -1;
      for each structure S in PossibleStructures {
        NodesSaved = DereferenceNode(AIG, N);
        NodesAdded = ReferenceNode(AIG, S);
        Gain = NodesSaved - NodesAdded;
        Dereference(AIG, S); Reference(AIG, N);
        if (Gain > 0 || (Gain = 0 && UseZeroCost)){
          if (BestS = NULL || BestGain < Gain){
            BestS = S; BestGain = Gain;
          }
        }
      }
      if (BestS = NULL){
        continue;
      }
      // use the best logic structure to update the
netlist
      NodesSaved = Dereference(AIG, N);
      NodesAdded = ReferenceNode(AIG, N);
      assert (BestGain = NodesSaved - NodesAdded);
    }
  }
}
```

Zdrojový kód 1: algoritmus rewrite [12]

```
void NetworkKFeasibleCuts (Graph g, int k){
  for each primary output node n of g {
    NodeKFeasibleCuts(n, k);
  }
}

cutset NodeKFeasibleCuts (Node n, int k){
  if (n is primary input) return {{n}};
  if (n is visited) return NodeReadCutSet(n);
  mark n as visited;
  cutset Set1 = NodeKFeasibleCuts(NodeReadChild1(n),
k);
  cutset Set2 = NodeKFeasibleCuts(NodeReadChild2(n),
k);
  cutset Result = MergeCutSets(Set1, Set2, k) U {n};
  NodeWriteCutSet(n, Result);
  return Result;
}

cutset MergeCutSets (cutset Set1, cutset Set2, int
k){
  cutset Result = {};
  for each cut Cut1 in Set1{
    for each cut Cut2 in Set2{
      if (|Cut1 U Cut2| <= k){
        Result = Result U {Cut1 U Cut2};
      }
    }
  }
  result Result;
}
```

Zdrojový kód 2: hledání k-feasible řezů [21]

Optimální reprezentace řezů v XAIG jsme předpočítali načtením pravděpodobnostní tabulky pomocí příkazu *read_truth* do ABC, optimalizovány příkazem *dch* a následně namapovány příkazem *map* do standartních buněk knihovnou obsahující invertory s cenou 1, a hradla AND a XOR s cenou 2. Namapované netlisty byly následně převedeny do XAIG. Totožná cena ANDů a XORů umožní vytvořit každý identifikovaný XOR. Rozhodnutí o tom, jak výsledné XORy implementovat tak bude necháno na mapperu.

Implementaci rewrite nad XAIG bude následovat porovnání jeho výkonu s rewritingem nad AIG. Tento experiment by měl potvrdit to, že nad XAIG lze reimplementovat algoritmy pracující nad AIG a dosáhnout tak lepšího výkonu těchto algoritmů. V opačném případě by měl ukázat, že rewriting nelze zlepšit identifikací a využitím XORů a bude třeba hledat jinou cestu.

PODĚKOVÁNÍ

- GAČR: GA16-05179S Výzkum vztahů a společných vlastností spolehlivých a bezpečných architektur založených na programovatelných obvodech (Fault-Tolerant and Attack-Resistant Architectures Based on Programmable Devices: Research of Interplay and Common Features) (03/2016 - 12/2018)
- SGS16/121/OHK3/1T/18 (Bezpečné a spolehlivé architektury vhodné pro implementaci v FPGA)
- Výpočetní prostředky byly poskytnuty CESNET LM2015042 and the CERIT Scientific Cloud LM2015085, pod programy "Projects of Large Research, Development, and Innovations Infrastructures"

REFERENCE

[1] E. McCluskey, “Minimization of boolean functions,” The Bell System Technical Journal, vol. 35, no. 6, pp. 1417–1444, Nov 1956.

[2] R. K. Brayton, A. L. Sangiovanni-Vincentelli, C. T. McMullen, and G. D. Hachtel, Logic Minimization Algorithms for VLSI Synthesis. Norwell, MA, USA: Kluwer Academic Publishers, 1984.

[3] S. Muroga, Y. Kambayashi, C. Lai, Hung, and N. Culliney, Jay, “The transduction method-design of logic networks based on permissible functions,” vol. 38, no. 10, pp. 1404–1424, 10 1989, iEEE Transactions on Computes

[4] S. B. Akers, “Binary decision diagrams,” IEEE Transactions on Computers, vol. 27, no. 6, pp. 509–516, Jun. 1978.

[5] R. E. Bryant, “Graph-based algorithms for boolean function manipulation,” IEEE Trans. Comput., vol. 35, no. 8, pp. 677–691, Aug. 1986.

[6] K. Karplus, “Using if-then-else DAGs for multi-level logic minimization,” in Proc. of Advance Research in VLSI, C. Seitz Ed. MIT Press, 1989, pp. 101–118.

[7] Y.-T. Lai, M. Pedram, and S. B. K. Vrudhula, “BDD based decomposition of logic functions with application to FPGA synthesis,” in Proceedings of the 30th International Design Automation Conference, ser. DAC’93. New York, NY, USA: ACM, 1993, pp. 642–647.

[8] C. Yang and M. Ciesielski, “BDS: a BDD-based logic optimization system,” vol. 21, no. 7, pp. 866–876, 08 2002, iEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.

[9] N. Vemuri, P. Kalla, and R. Tessier, “BDD-based logic synthesis for LUT-based FPGAs,” ACM Transactions on Design Automation of Electronic Systems, vol. 7, no. 4, pp. 501–525, 12 2001.

[10] A. Kuehlmann, V. Paruthi, F. Krohm, and M. Ganai, “Robust boolean reasoning for equivalence checking and functional property verification,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 21, no. 12, pp. 1377–1394, 12 2001.

[11] P. Bjesse and A. Borlv, “DAG-aware circuit compression for formal verification,” in IEEE/ACM International Conference on ComputerAided Design, 2004, pp. 42–49.

[12] K. Brayton, Robert, A. Mishchenko, and S. Chatterjee, “DAG-aware AIG rewriting: a fresh look at combinational logic synthesis,” in 43rd ACM/IEEE Design Automation Conference. ACM, 2006, pp. 532–535.

[13] A. Mishchenko et al., “ABC: A system for sequential synthesis and verification,” 2012. [Online]. Available: http://www.eecs.berkeley.edu/˜alanmi/abc

[14] R. Brayton and A. Mishchenko, “ABC: an academic industrial-strength verification tool,” in Proceedings of the 22Nd International Conference on Computer Aided Verification, ser. CAV’10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 24–40

[15] J. Cong and K. Minkovich, “Optimality study of logic synthesis for LUT-based FPGAs”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 2007, Vol. 26, No. 2, pp. 230–239.

[16] P. Fišer and J. Schmidt, “Small But Nasty Logic Synthesis Examples”, Proceedings of the 8th. Int. Workshop on Boolean Problems, 2008, Freiberg, pp. 183–189.

[17] F. Brglez and H. Fujiwara, “A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran,” in Proceedings of IEEE International Symposium Circuits and Systems (ISCAS 85). IEEE Press, Piscataway, N.J., 1985, pp. 677–692.

[18] F. Brglez, D. Bryan, and K. Kozminski, “Combinational profiles of sequential benchmark circuits,” in IEEE International Symposium on Circuits and Systems, 1989., May 1989, pp. 1929–1934 vol.3.

[19] S. Yang, “Logic synthesis and optimization benchmarks user guide: Version 3.0,” MCNC Technical Report, Tech. Rep., Jan. 1991.

[20] K. McElvain, “IWLS’93 Benchmark Set: Version 4.0,” Tech. Rep., May 1993.

[21] A. Mishchenko et al. Technology Mapping with Boolean Matching, Supergates and Choices, ERL Technical Report, EECS Dept., UC Berkeley, 2005.

[22] L. Amarù, P.-E. Gaillardon, G. De Micheli, “Biconditional Binary Decision Diagrams: A Novel Canonical Representation Form”, IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS), Vol. 4, No. 4, 2014, pp. 487-500.

[23] L. Amarù, P.-E. Gaillardon, G. De Micheli, “Boolean Logic Optimization in Majority-Inverter Graphs”, Design Automation Conference (DAC), San Francisco, CA, USA, 2015.

Zpracování signálů EEG na obvodech FPGA

Martin Huněk
1. ročník, prezenční studium
Školitel: Zdeněk Plíva

Technická univerzita v Liberci
Studentská 2, 461 17 Liberec 1, ČR
martin.hunek@tul.cz

***Abstrakt*—Disertační práce bude rámcově zaměřena na oblast zpracování biologických signálů, zejména EEG (elektro-encefalogramu). Hlavním cílem je prověření možností využití hlubokých neuronových sítí pro monitorování jednoduchých mozkových aktivit. V rámci tohoto výzkumu budou zkoumány možnosti filtrace šumu ze signálů naměřených pomocí neinvazivních povrchových elektrod. Jedním z klíčových úkolů je prověření možnosti minimalizace počtu elektrod při zachování rozlišovacích schopností neuronové sítě, tedy včetně simulace výpadků jednotlivých elektrod. Důraz je kladen zejména na oblasti motorického a řečového centra přičemž primárním cílem je částečná náhrada funkce mozku u spinálních poranění.**

***Klíčová slova*—EEG, zpracování signálů, FPGA, neuronové sítě, strojové učení.**

I. Úvod

V tomto příspěvku popisujeme pouze výchozí rámcové představy na možná řešení problematiky numerického zpracování dat elektro-encefalogramu (EEG) na obvodech FPGA při použití metod strojového učení. V následujícím textu jsou formulovány základní obrysy budoucí disertační práce; v současné době probíhá studium výchozích odborných předmětů v řešené oblasti. Současně probíhají i konzultace s vybranými lékařskými pracovišti o možnostech sběru dat, jejich standardním zpracování, metodice vyhodnocování a v neposlední řadě také o možnostech a zaměření další spolupráce a společného výzkumu v této oblasti. Primární cíl disertace nebude zaměřen na samotné techniky pro sběr dat případně na související problematiku vlastního snímání, tedy provedení, umístění či obecně kvalita jednotlivých používaných elektrod. V této fázi spolupráce se budeme vyhýbat klinickým testům. Rovněž nebude hlavním cílem pre-processing naměřených dat, i když právě v této oblastí je velmi dobrá možnost navázat na dosavadní výzkum v oblastí filtrování dat pomocí metody slepé separace (Blind Separation) prováděné doc. Koldovským na našem ústavu, případně stavět na využití metody Independent Component Analysis (ICA) na filtrování EEG dat publikované zde [1].

II. Současný stav poznání

V současné době probíhá většina úkonů při zpracování EEG signálů převážně ručně, s výrazně subjektivním vyhodnocením lékaře na základě jeho vlastních zkušeností a poznatků a s minimální podporou výpočetní techniky. Počítač bývá nejvíce využit jako pouhé off-line zobrazovací zařízení dříve naměřených dat s možnostmi zoomování nasnímaného signálů a také s možností ručního označování potenciálně zajímavých příznaků, nebo nestandardních projevů. Lékař tyto zkušenosti získává z části při studiu na lékařské fakultě, kdy se naučí základní rozložení mozkových funkcí v mozku, podstatu vzniku bio-elektrických signálů, možnosti jejich ovlivňování či jejich klinické manifestace. Z větší části se pak lékaři učí rozpoznávat a vyhodnocovat EEG v rámci získávání atestace pod dohledem pověřeného mentora a ve většině zdravotnických zařízení je několik málo specialistů, kteří se na toto vyhodnocování specializují, ovšem toto vyhodnocení neprobíhá on-line. Uvedený postup však přímo vybízí k využití neuronových sítí, neboť podstata trénování sítě probíhá srovnatelným způsobem a na této podobnosti lze vybudovat podpůrnou infrastrukturu pro semiautomatizované zpracování informací s klíčováním pravděpodobných projevů nestandardní činnosti sledovaných částí mozku. Pokud se takto natrénovaná síť implementuje do vhodného hardware, například FPGA obvodu, bylo by možné typické příznaky označovat on-line, nebo alespoň zvýraznit ty oblasti průběhů, na které by se měl specializta zaměřit.

Již v současné době existují vybrané spacializované aplikace v oblasti asistované diagnostiky v medicíně, v této souvislosti je nutno zmínit například diagnostika deprese která byla implementována na Tallinn university of Technology [2]. Tato metodika dokáže identifikovat s dostatečným předstihem náznaky blížícího se kolapsu na základě typických deformací EEG záznamu a tak uspíšit odborný lékařský zásah na dobu před fatálním selháním. Další oblastí již publikovaného nasazení výpočetní techniky je nasazení Fuzzy logiky pro diagnostiku epileptických záchvatů na základě typických, a algoritmizovatelných projevů v naměřených průbězích [3].

Je zřejmé, že již nyní existují vhodné matematické prostředky, které pomáhající lékaři s diagnostikou naměřených dat. Například pokud se se znalostí teorie a techniky zpracování signálů budeme zabývat procesem určování stavu vědomí, zjistíme, že určujícím parametrem je frekvence signálů. A toto zjištění automaticky vede k frekvenční analýze. Tuto pomůcku dnes používají lékaři poměrně standardně pod názvem „mapování“, toto zpracování je prováděno a je hrazeno zdravotními pojišťovnami. Zároveň však tato metodika není pro lékaře v procesu diagnózy stěžejní, místo toho se spoléhají na ”ruční”vyhodnocování průběhů křivek signálů a na po-

rovnání signálů mezi sebou.

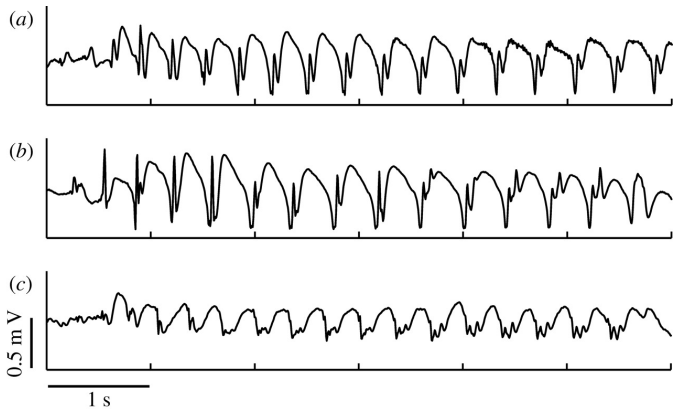
III. Rizika

Určitým omezením výzkumu při tvorbě nové diagnostické pomůcky pro oblast zdravotnictví je zejména proces schvalování zdravotnických prostředků a velký důraz na etická pravidla v oblasti zpracování osobních dat a údajů. Celá problematika zdravotnických prostředků se řídí zákonem o zdravotnických prostředcích (zákon 123/2000 Sb. [4], ve znění pozdějších předpisů). Ten stanovuje v §2 odst. 1 písmene a definici zdravotnického prostředku jako takového, který je výrobcem určen pro použití u člověka za účelem *stanovení diagnózy, prevence, monitorování, léčby nebo mírnění choroby,*. Zároveň stejný zákon v §2 odst. 2 obsahuje definici aktivního diagnostického zdravotnického prostředku: *Aktivní diagnostický zdravotnický prostředek je takový, který je použit samostatně nebo v kombinaci s dalšími prostředky k dodávání informací pro diagnostikování, monitorování, zjišťování nebo léčbu fyziologických stavů, stavu zdraví, nemocí nebo vrozených vad*.

Pokud vezmeme v úvahu i nařízení vlády 336/2004 Sb. kterým se stanoví technické požadavky na zdravotnické prostředky, dojdeme k závěru, že pokud by se mělo jednat o zdravotnický prostředek, jednalo by se o zařízení třídy IIb nebo III. S tím se pojí požadavek na klinické testy před uvedením na trh, periodické kontroly a další požadavky. V prvních fázích vývoje algoritmů a zařízení se tedy budeme vyhýbat klinickému testování, v počátcích vývoje namísto přímého použití v oblasti diagnostiky budou zpracovávána testovací anonymní data za účelem statistiky. Na základě tohoto zpracování budou navrženy jednotlivé algoritmy, bude provedeno trénování a ověřování funkčnosti cílové implementace a až po celkovém ověření budeme usilovat o případné provádění klinických testů v praktickém nasazení na ostrých zdravotnických datech, samozřejmě při splnění výše uvedených nařízení.

IV. Možné oblasti řešených problematik

I přes zmíněná rizika se v první fázi vývoje diagnostických pomůcek nabízí několik oblastí k řešení. Jednou z nich je již zmíněná včasná diagnostika příznaků epileptického záchvatu. Tu je možné z EEG průběhů poměrně jednoduše rozpoznat a to i bez hlubší znalosti problematiky. Vyznačuje se totiž jak specifickým průběhem signálů v časové oblasti (špička/vlna, obrázek 1) tak hlavně navzájem synchronními signály z více elektrod [5], které se po odfiltrování EKG běžně v mozku nevyskytují. Při vyhodnocení epilepsie se pak otevírá možnost přibližné lokalizace ložiska a vyhodnocení závažnosti záchvatu. Jedná se o poměrně dobře prozkoumanou a dokumentovanou tématiku a proto je také vhodná pro trénování neuronových sítí a vyhodnocení úspěšnosti tohoto procesu. V oblasti praktického nasazení je pak například možné vytvořit relativně jednoduché a energeticky úsporné zařízení, které v kombinaci s mobilním telefonem dokáže monitorovat blížící se záchvaty a s předstihem tak odesílat příslušná data neurologovi, ošetřujícímu personálu, případně varovnou informaci na vhodné adresy.



Obrázek 1. Průběhy epilepsie – převzato z [5] (průběh a – klasický špička/vlna, průběhy b+c – s nadpočetnou špičkou)

Druhou oblastí vývoje a možného nasazení je vytvoření jednoduché kompenzační pomůcky například pro pacienty po spinálním poranění. V tomto případě se jedná o pokračování dosavadní několikaleté spolupráce se spinální jednotkou krajské liberecké nemocnice, se kterou byla v rámci předchozího studia vyvinuta metodika návrhu implantátu pro fixaci zlomenin pánevního kruhu. V této oblasti výzkumu se jedná hlavně o úlohu přesunu procesu algoritmizovatelného vyhodnocení jednotlivých signálů z procesoru na úspornější programovatelné obvody FPGA.

Samotná architektura programovatelných obvodů FPGA by zde byla použita jako cílová aplikační platforma. V implementační fázi by FPGA mohlo snížit nároky na spotřebu elektrické energe a tím umožnit dlouhodobější bateriové napájení kompenzační pomůcky při zachování možnosti real-time zpracování. Zároveň je však finnančně dostupnější než řešení pomocí ASIC (s ohledem na nízký počet kusů a možnost variability) a může tak tvořit kompromis mezi univerzálností běžných procesorů a efektivitou ASIC. Jako vývojová platforma pro nalezení a natrénování neuronové sítě bude využita architektura grafických karet, jako je tomu v případě jiných aplikací (u nás například u zpracování řeči). Pokud ovšem bude požadavek směřovat na oblast vývoje diagnostické opory, pak by požadavek na striktní real-time zpracování odpadl, stejně jako požadavek na mobilitu. A pro tuto aplikaci by se jako vhodnější jevilo použití běžného počítače (případně i s vhodným hardwarovým aparáte) jako cílové platformy.

V. Závěr

V současné době probíhá studium základních předmětů stanovených individuálním studijním plánem a tedy studium základních témat, které budou využity pro realizaci definovaného cíle disertační práce. Současně probíhají i jednání s vybranými specializovanými pracovišti, která se budou podílet zejména na shromáždění vhodných dat, průběžné konzultace při vývoji zařízení; v neposlední řadě je třeba vyřešit i otázku výběru pracoviště pro předepsaný studijní pobyt.

PODĚKOVÁNÍ

Tento článek byl podpořen z fondu Studentské grantové soutěže (SGS 2016) na Technické univerzitě v Liberci.

VLASTNÍ PUBLIKACE

[1] E. KANIOVÁ a M. HUNĚK, "[el] – variable light", in *Workshop for Ph.D Students of Faculty of Textile Engineering and Faculty of Mechanical Engineering TUL*, Liberec: Technical university of Liberec, 2015, s. 87–91, ISBN: 978-80-7494-229-7.

[2] M. HUNĚK, "Anthropometric measurements of hollow bone structures based upon computer assisted tomography", in *DCPS - Dependable Cyber Physical Systems*, Cottbus: Brandenburg University of Technology Cottbus-Senftenberg, 2015, s. 48–49.

[3] ——, "Bezpečnost zdravotnických informací a bezpečné chování v počítačových sítích", in *Ošetřovatelství bez hranic*, Liberec: Technická univerzita v Liberci, 2015, s. 18–22, ISBN: 9788074941931.

[4] ——, "Anthropometric measurement pelvis software pack", Technická univerzita v Liberci, soft. TAČR 04011720, 2015.

LITERATURA

[1] M. Zima, P. Tichavský, K. Paul a V. Krajča, "Robust removal of short-duration artifacts in long neonatal eeg recordings using wavelet-enhanced ica and adaptive combining of tentative reconstructions", *Physiological Measurement*, sv. 33, č. 8, N39–N49, 2012. DOI: 10.1088/0967-3334/33/8/N39. WWW: http://stacks.iop.org/0967-3334/33/i=8/a=N39.

[2] M. Jenihhin, M. Gorev, V. Pesonen, D. Mihhailov, P. Ellervee, H. Hinrikus, M. Bachmann a J. Lass, "Eeg analyzer prototype based on fpga", in *Image and Signal Processing and Analysis (ISPA), 2011 7th International Symposium on*, zář. 2011, s. 101–106.

[3] R. HariKumar a M. Balasubramani, "Fpga synthesis of soft decision tree (sdt) for classification of epilepsy risk levels from fuzzy based classifier using eeg signals", *International Journal of Soft Computing and Engineering*, sv. 1, č. 4, s. 206–211, 2011.

[4] (3. ún. 2014). Zp-19 verze 3, Státní ústav pro kontrolu léčiv, WWW: http://www.sukl.cz/zdravotnicke-prostredky/zp-19-verze-3 (cit. 15. 06. 2016).

[5] F. Marten, S. Rodrigues, O. Benjamin, M. P. Richardson a J. R. Terry, "Onset of polyspike complexes in a mean-field model of human electroencephalography and its application to absence epilepsy", *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, sv. 367, č. 1891, s. 1145–1161, 2009, ISSN: 1364-503X. DOI: 10.1098/rsta.2008.0255. eprint: http://rsta.royalsocietypublishing.org/content/367/1891/1145.full.pdf. WWW: http://rsta.royalsocietypublishing.org/content/367/1891/1145.

Analýza prúdových zrkadiel riadených substrátovou elektródou

Matej Rakús
1. ročník, denná prezenčná forma štúdia
Školiteľ: Viera Stopjaková

Fakulta elektrotechniky a informatiky, Slovenská technická univerzita v Bratislave
Ilkovičova 3, 812 19 Bratislava, SR
matej.rakus@stuba.sk

***Abstrakt*—Tento príspevok sa zaoberá analýzou rôznych techník, ktoré sa využívajú pri návrhu nízko-napäťových a nízko-príkonových integrovaných obvodov (IO) a detailnejšie rozoberá návrh prúdových zrkadiel riadených substrátovou elektródou v 90 nm štandardnej CMOS technológii. Princíp tranzistorov riadených substrátovou elektródou je v ich štruktúre, kde je substrát využitý ako vstupný terminál. Táto metóda redukuje, prípadne odstraňuje nevyhnutnosť prekročenia prahového napätia na vstupe MOS tranzistora. Tranzistor riadený substrátovou elektródou je možné vytvoriť z bežného MOS tranzistora bez zmeny jeho štruktúry alebo technologického procesu. Prúdové zrkadlá zložené z takýchto tranzistorov sú schopné pracovať pri napájacích napätiach menších ako prahové napätie tranzistora, vďaka čomu je možné znížiť veľkosť napájacieho napätia pre obvod. Výsledky získané zo simulácií potvrdzujú, že táto technika je vhodnou pre návrh nízko-napäťových IO s nízkou spotrebou.**

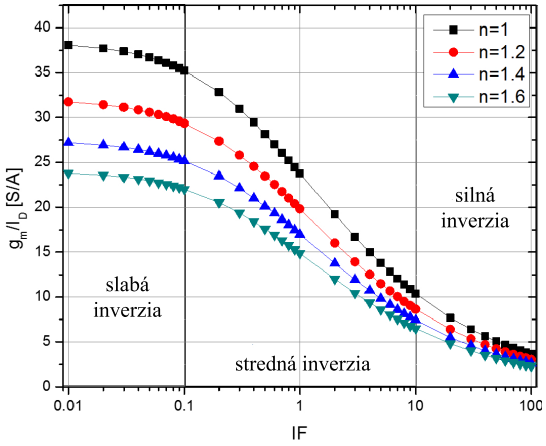
***Kľúčové slová*—návrh nízko-napäťových IO; návrh nízko-príkonových IO; bulk-driven; analógové obvody; prúdové zrkadlá**

I. Úvod

Zvýšený dopyt po všadeprítomných prenosných obvodoch a systémoch spôsobili narastajúcu potrebu rozvoja nízko-napäťových a nízko-príkonových techník pre návrh elektrických obvodov a obvodových blokov. Z tohto dôvodu je trendom zmenšovať minimálny rozmer technologického procesu, zvyšovať hustotu komponentov na čipe a znižovať napájacie napätie. Zmenšovanie veľkosti tranzistorov do submikrometrových rozmerov a neustále zmenšovanie hrúbky vrstvy izolačného oxidu na rozmer niekoľkých nanometrov, má za následok nízke prierazné napätia tranzistorov. Na zabezpečenie správnej činnosti a spoľahlivosti obvodu je teda potrebné znížiť aj napájacie napätie. Zvyšovanie hustoty komponentov na čipe vedie k zmenšovaniu celkových rozmerov čipu. Nakoľko je kremíkový substrát schopný rozptýliť iba určité množstvo tepla na jednotku plochy, príkon jednotlivých funkčných blokov IO musí byť taktiež znížený. Existuje niekoľko techník pre návrh nízko-napäťových a nízko-príkonových IO [1].

II. Techniky návrhu nízko-napäťových a nízko-príkonových IO

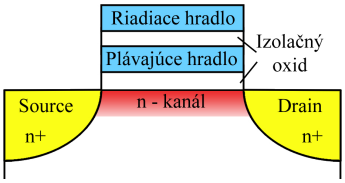
1) Tranzistory pracujúce v slabej inverzii: Táto technika využíva oblasť slabej inverzie tranzistorov, kedy parameter g_m/I_D dosahuje najvyššie hodnoty (Obr. 1) a obvod najnižšiu spotrebu. Nevýhodou tejto techniky je veľká spotreba plochy, keďže je potrebné používať tranzistory s vysokým pomerom šírky a dĺžky kanála W/L [2].



Obr. 1. Závislosť g_m/I_D od inverzného faktora (IF)

2) Tranzistory s plávajúcim hradlom: Tranzistory s plávajúcim hradlom (ďalej FGMOS *z angl.* Floating-Gate Metal-Oxid-Semicondictor) majú oproti bežným MOS tranzistorom izolované plávajúce hradlo (Obr. 2). Napätie na plávajúcom hradle V_{FG} nie je riadené priamo, ale riadiacim hradlom pomocou kapacitnej väzby. Ekvivalentné prahové napätie na riadiacom hradle môže byť redukované regulovaním množstva statického náboja Q_{FG} na plávajúcom hradle. Náboj Q_{FG} je možné regulovať aj pomocou UV žiarenia, injekcie horúcich elektrónov alebo Fowler-Nordheimhovho tunelovania. Tieto techniky môžu znížiť prahové napätie V_{TH} FGMOS tranzistora, načo sú ale potrebné relatívne komplexné programovacie obvody a pomerne vysoké hodnoty napätia, čo limituje použitie FGMOS tranzistorov v nízko-napäťových aplikáciách [3].

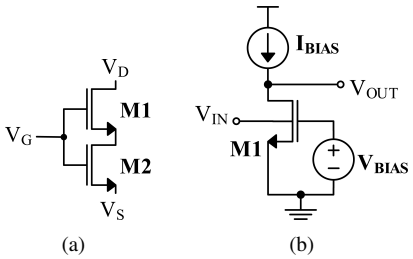
3) „Self-cascode“ topológia: Táto topológia znázornená na Obr. 3(a) má vysoký výstupný odpor s väčším výstupným rozkmitom napätia ako klasická kaskódová štruktúra



Obr. 2. Rez štruktúrou FGMOS tranzistora

[4]. Tranzistor M1 (M2) pracuje mimo saturácie (v saturácii) a pre $(W/L)_2 >> (W/L)_1$ sa obvod správa ako samostatný tranzistor pracujúci v saturácii, ale bez efektu modulácie dĺžky kanála. Výstupný odpor priamoúmerný pomeru $(W/L)_2/(W/L)_1$ a saturačné napätie $V_{DS,sat} = V_{GS} - V_{TH}$ je rovnaké ako pri samotnom MOSFET tranzistore.

4) Tranzistory riadené substrátovou elektródou: Technika využívajúca tranzistory riadené substrátovou elektródou (ďalej BD *z angl.* Bulk-driven) sa javí ako sľubná pre návrh nízkonapäťových integrovaných obvodov. MOS tranzistor je aktívny elektronický prvok so štyrmi vývodmi, z ktorých sa bežne využívajú alebo sú vyvedené iba tri: hradlo (G *z angl.* Gate), kolektor (D *z angl.* Drain) a emitor (S *z angl.* Source). Štvrtý vývod – substrátový kontakt (B *z angl.* Bulk) je zvyčajne interne prepojený s emitorom, prípadne sa pripája na jeden z napájacích potenciálov obvodu [5]. Táto elektróda však môže byť využitá aj ako signálový vstup obvodu (Obr. 3(b)), čo má za následok zníženie alebo odstránenie potreby prekonávať prahové napätie v signálovej ceste [6].



Obr. 3. Topológie: (a) „Self-cascode“; (b) „Bulk-Driven“

III. „Bulk-driven“ prúdové zrkadlá

Prúd tečúci MOS tranzistorom - kolektorový prúd I_D je riadený napätím medzi hradlom a emitorom MOS tranzistora V_{GS}. Tento prúd môže byť čiastočne ovplyvnený aj napätím medzi substrátom a emitorom MOS tranzistora V_{BS}, čo je zvyčajne považované za parazitný jav. Pripojením konštantného napätia $V_{GS} = V_{BIAS}$ a privedením vstupného signálu na substrátovú elektródu vykazuje MOS tranzistor podobné vlastnosti ako JFET tranzistor. Táto technika odstraňuje, prípadne znižuje potrebu pripojenia vyššieho napätia ako prahového napätia $V_{GS} > V_{TH}$ na riadiacu elektródu, čím sa výrazne zvyšuje rozpätie pracovných napätí a znižuje hodnota napájacieho napätia obvodu. Nevýhodou tejto metódy návrhu je vodivosť substrátu BD tranzistora g_{mb} (1), ktorá je tri až päťkrát nižšia ako prenosová vodivosť tranzistora g_m [7].

$$g_{mb} = \frac{\gamma g_m}{2\sqrt{2|\phi_F| - V_{BS}}} \tag{1}$$

Vstupná kapacita sa zmení z $C_{gs} + C_{gb}$ na $C_{b,sub} + C_{bs}$, čo môže negatívne ovplyvniť frekvenčnú charakteristiku a redukovať šírkeú pásma zosilnenia (GBW *z angl.* Gain-Bandwidth). Okrem toho, použitie BD tranzistorov zvyšuje riziko zopnutia parazitných bipolárnych tranzistorov v substráte, čo môže viesť k problémom s tzv. „latch-up“ javom.

A. Jednoduché (Widlarovo) BD prúdové zrkadlo

Ekvivalent jednoduchého prúdového zrkadla (ďalej PZ) navrhnutý pomocou BD techniky je znázornený na Obr. 4(a). Substrátové elektródy tranzistorov v oboch vetvách PZ sú navzájom prepojené a na hradlá je pripojené predpätie V_{BIAS} [8]. Predpätím je možné znížiť prahové napätie V_{TH} tranzistora podľa vzťahu (2), kde V_{T0} je prahové napätie, ak $V_{BS} = 0V$, ϕ_F je Fermiho potenciál substrátu a γ je tzv. substrátový činiteľ [9]:

$$V_{TH} = V_{T0} + \gamma(\sqrt{2|\phi_F| - V_{BS}} - \sqrt{2|\phi_F|}) \tag{2}$$

Ak je výstupné napätie tranzistora M2 menšie (väčšie) ako saturačné napätie $V_{DS2} < V_{DS,sat}$ ($V_{DS2} > V_{DS,sat}$), potom výstupný prúd je možné vyjadriť pomocou vzťahu (3) (vzťahu (4)), kde K_P je technologický činiteľ, λ je koeficient modulácie dĺžky kanála a n je strmosť PN priechodu.

$$I_D = \frac{K_P W}{nL}\Big(V_{GS} - V_{T0} - \gamma(\sqrt{2|\phi_F| - V_{BS}} - \sqrt{2|\phi_F|}) - \frac{V_{DS}}{2}\Big)V_{DS} \tag{3}$$

$$I_D = \frac{K_P W}{2nL}\Big(V_{GS} - V_{T0} - \gamma(\sqrt{2|\phi_F| - V_{BS}} - \sqrt{2|\phi_F|})\Big)^2(1 + \lambda V_{DS}) \tag{4}$$

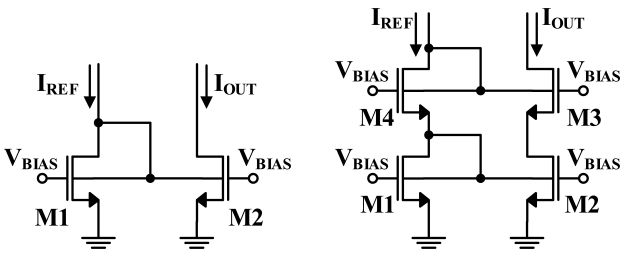
Pokiaľ je $V_{DS1} < V_{DS,sat}$, tranzistor M1 pracuje v lineárnom režime. Táto podmienka ja podporená prepojením terminálov kolektora so substrátom MOS tranzistora. Keďže substrátové elektródy oboch tranzistorov sú navzájom prepojené, potom $V_{BS1} = V_{BS2}$ a $V_{GS1} - V_{TH} = V_{GS2} - V_{TH}$. Hodnota výstupného prúdu $I_{OUT} = I_{D2}$ nie je obmedzená a teda pre výstupné napätie môžeme dosiahnuť hodnoty $V_{OUT} \geq V_{DS,sat}$ [10]. Vzťah medzi referenčným a výstupným prúdom (5) sa získa vyriešením rovnice (3) pre $V_{GS2} - V_{TH}$ a dosadením do rovnice (4).

$$I_{OUT} = \frac{K_P W_2}{2nL_2}\left(\frac{I_{REF}^2}{(\frac{K_P W_1}{nL_1})^2 V_{DS1}^2} + \frac{I_{REF}}{\frac{K_P W_2}{nL_2}} + \frac{V_{DS1}^2}{4}\right)(1 + \lambda V_{DS2}) \tag{5}$$

Výstupný odpor jednoduchého PZ $r_{rout} = r_{ds2}$ je relatívne malý v porovnaní s požadovanou (ideálnou) hodnotou. Rozkmit výstupného napätia jednoduchého PZ je limitovaný minimálnou hodnotou výstupného napätia $V_{OUT} = V_{DS,sat}$.

B. Kaskódové BD prúdové zrkadlo

Kaskódové prúdové zrkadlo je znázornené na Obr. 4(b). Táto topológia sa vyznačuje veľmi vysokou presnosťou zrkadlenia vďaka zápornej sériovej spätnej väzbe. Výstupný odpor BD kaskódového PZ sa oproti jednoduchému PZ zvýši na $r_{out} = r_{ds2} g_{mb4} r_{ds4}$. Minimálne výstupné napätie potrebné na správnu činnosť kaskódového zrkadla je $V_{OUT,min} = V_{TH} + 2V_{DS,sat}$. Minimálnu hodnotu výstupného napätia je možné zredukovať až na hodnotu $V_{OUT,min} = 2V_{DS,sat}$, pretože ako bolo spomenuté vyššie, BD technika je schopná znížiť či dokonca odstrániť závislosť tranzistorov od prahového napätia [11].

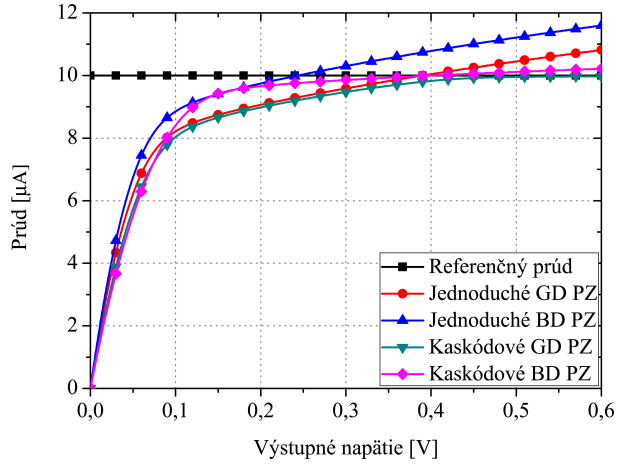


(a) Jednoduché (Widlarovo) (b) Kaskódové

Obr. 4. Základné topológie BD prúdových zrkadiel

IV. Dosiahnuté výsledky

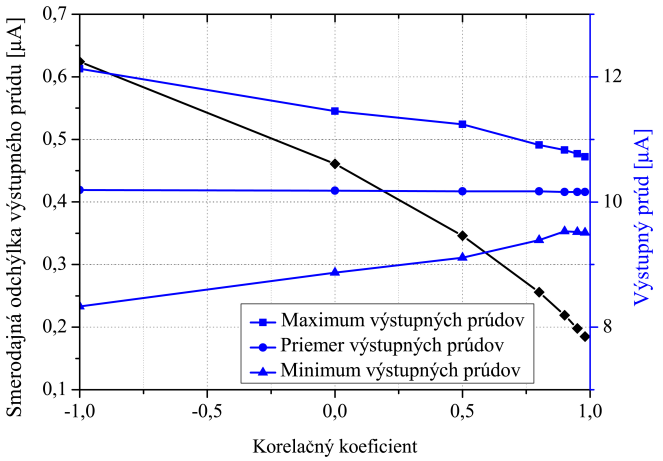
Nasledujúce výsledky boli získané simuláciami pre prúdové zrkadlá navrhnuté v 90 nm štandardnej CMOS technológii. Výsledky simulácií BD zrkadla boli porovnané s ekvivalentnými topológiami využívajúcimi bežné tranzistory riadené hradlom (ďalej GD *z angl.* Gate-Driven). Všetky tranzistory použité v daných topológiách mali rozmery $W = 5\ \mu m$ a $L = 1\ \mu m$. Jednoduché GD a BD prúdové zrkadlá majú podobnú hodnotu výstupného odporu vzťahujúceho sa na $r_{ds2} = 25\ k\Omega$, ktorý určuje sklon výstupných charakteristík na Obr. 5.



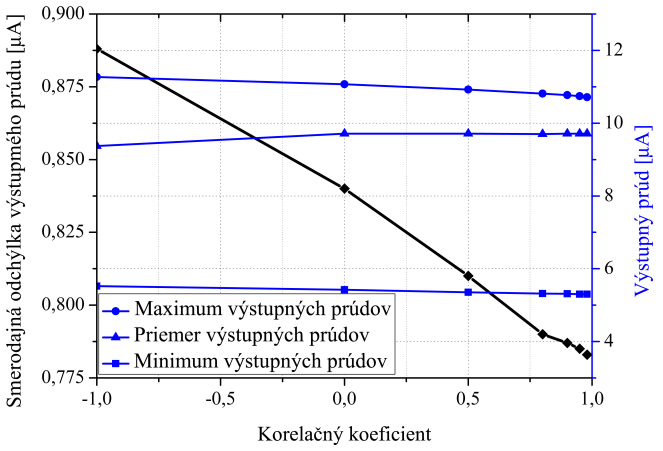
Obr. 5. Výstupné charakteristiky jednoduchého a kaskódového PZ

Minimálna hodnota výstupného napätia jednoduchého PZ dosiahnutá pre obe techniky je rovnaká $V_{OUT,min} = V_{DS2} = 78\ mV$. Kaskódové PZ má oproti jednoduchému PZ výrazne väčší výstupný odpor. Z krivky na danom grafe je možné pozorovať, že kaskódové GD PZ začína zrkadliť až po prekročení minimálneho výstupného napätia $V_{OUT,min} = 477\ mV$, čo je takmer nepoužiteľné pre nízko-napäťové aplikácie s napájaním $V_{DD} = 0,5 - 0,6\ V$. Použitie BD techniky znižuje hodnotu napätia na úroveň $V_{OUT,min} = 140\ mV$ a umožňuje tak použiť túto topológiu aj v nízko-napäťových IO. Na druhej strane, použitie BD techniky znižuje hodnotu výstupného odporu PZ z hodnoty $r_{out} = 4,36\ M\Omega$ na hodnotu $r_{out} = 0,81\ M\Omega$. Tento pokles výstupného odporu je spôsobený znížením vodivosti z g_m na g_{mb}, ktorá podstatne vplýva na výstupný odpor [12].

Na Obr. 6 sú zobrazené štatistické Monte Carlo simulácie výstupného prúdu I_{OUT} jednoduchého BD PZ pri referenčnom prúde $I_{REF} = 10\ \mu A$. Z grafu možno pozorovať, že rozptyl parametrov jednoduchého BD zrkadla je významne závislý od korelačného koeficientu (cc) oboch tranzistorov PZ.



Obr. 6. Monte Carlo simulácie výstupných prúdov jednoduchého BD PZ v závislosti od korelačného koeficientu



Obr. 7. Monte Carlo simulácie výstupných prúdov kaskódového BD PZ v závislosti od korelačného koeficientu

Podobné simulácie boli vykonané aj pre kaskódové BD PZ (Obr. 7), ktoré nie je až natoľko závislé od korelácie tranzistorov vďaka zápornej sériovej spätnej väzbe. Ako sa dalo očakávať, smerodajné odchýlky oboch štatistických simulácií majú klesajúcu tendenciu v závislosti od zlepšujúcich sa korelačných podmienok.

Tabuľka I
POROVNANIE GD A BD TOPOLÓGIÍ PZ

Technika návrhu	Topológia	$cc = 0,98$		Smerodajná odchýlka		
		r_{out} $[M\Omega]$	V_{MIN} $[mV]$	r_{out} $[k\Omega]$	V_{min} $[mV]$	I_{out} $[\mu A]$
GD	Jednod.	0,29	78,6	6,3	1,68	0,12
	Kaskod.	4,36	477	32,6	2,11	0,43
BD	Jednod.	0,26	76,8	8,9	2,37	0,18
	Kaskod.	1,48	139,8	48,9	12,57	0,78

Tabuľka I zobrazuje porovnanie základných výstupných parametrov prúdových zrkadiel pre GD a BD topológie. Získané výsledky potvrdzujú, že BD technika vylepšuje hodnotu minimálneho výstupného napätia, ale na úkor hodnoty výstupného odporu PZ.

Tabuľka II
ROZPTYL PARAMETROV V ZÁVISLOSTI OD TEPLOTY

Parameter	Jednoduché BD PZ			Kaskódové BD PZ		
T $[°C]$	-20	25	85	-20	25	85
I_{out} $[\mu A]$	9,71	10,16	10,8	5,86	9,7	10,44
r_{out} $[M\Omega]$	0,268	0,263	0,252	1,80	1,48	0,687

Tabuľka II zobrazuje rozptyl výstupných parametrov (I_{OUT} a r_{out}) BD PZ v závislosti od teploty. Zo získaných výsledkov je zrejmé, že kaskódové BD PZ je omnoho viac závislé od teploty okolia v porovnaní s jednoduchým BD PZ.

V. ZÁMER DIZERTAČNEJ PRÁCE A JEJ CIELE

Hlavným zámerom dizertačnej práce je podrobnejší rozbor techník, ktoré je možné použiť pri návrhu nízko-napäťových a nízko-príkonových IO. Našim zámerom je taktiež návrh robustných stavebných blokov analógových IO použiteľných pri návrhu zložitých nízko-napäťových integrovaných systémov. Vzhľadom na použiteľnosť jednotlivých návrhových techník vo vybraných technológiách, bude náš budúci výskum zameraný primárne na využitie tranzistorov riadených substrátovou elektródou. V budúcom výskume bude vykonaná podrobnejšia analýza zložitejších topológií nielen prúdových zrkadiel, ale aj diferenciálnych párov a iných obvodových blokov využívajúcich MOS tranzistory riadené substrátovou elektródou. Dôležitou súčasťou dizertačnej práce bude aj preskúmanie prípadného nežiaduceho vplyvu parazitných javov, ako napríklad „latch-up“ efektu na vlastnosti a použiteľnosť daných obvodových blokov. Táto analýza bude vykonaná nielen na úrovni simulácií, ale aj vo forme experimentálneho overenia vplyvu korelácie rôznych topológií obvodových blokov a štruktúr na zmenu ich charakteristík v 90 nm a 130 nm CMOS technológiách priamo na polovodičovom substráte.

VI. ZÁVER

V tomto príspevku boli uvedené najvýznamnejšie techniky využívané pri návrhu nízko-napäťových a nízko-príkonových integrovaných obvodov a ich princíp. Predložené simulácie základných topológií naznačujú použiteľnosť, výhody a nevýhody BD techniky v nízko-napäťových IO. Zo simulácií vyplýva, že táto technika môže výrazne znížiť minimálne napájacie napätie, a tak zvýšiť rozkmit pracovných napätí obvodu. Nevýhody a horšie vlastnosti niektorých parametrov BD topológií sa pokúsime zmierniť alebo odstrániť rôznymi obvodovými technikami.

V rámci mojej doterajšej práce a výskumu vznikli 3 publikácie (1 príspevok na medzinárodnom sympóziu DDECS a 1 príspevok na domácej konferencii ADEPT ako prvoautor a 1 príspevok na domácej konferencii ADEPT ako spoluautor).

POĎAKOVANIE

Táto práca bola podporená projektom APVV-15-0254. Autor zároveň ďakuje STU za finančnú podporu v rámci Grantovej schémy na podporu mladých výskumníkov.

LITERATÚRA

[1] Y. Shouli and E. Sanchez-Sinencio, “Low voltage analog circuit design techniques: A tutorial,” *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 83, no. 2, pp. 179–196, 2000.

[2] D. J. Comer and D. T. Comer, “Operation of analog mos circuits in the weak or moderate inversion region,” *IEEE Transactions on Education*, vol. 47, no. 4, pp. 430–435, Nov 2004.

[3] E. S. Sinencio, “Floating gate techniques and applications,” 2002.

[4] K. J. Baek, J. M. Gim, H. S. Kim, K. Y. Na, N. S. Kim, and Y. S. Kim, “Analogue circuit design methodology using self-cascode structures,” *Electronics Letters*, vol. 49, no. 9, pp. 591–592, April 2013.

[5] B. Razavi, *Design of Analog CMOS Integrated Circuits*, 1st ed. New York, NY, USA: McGraw-Hill, Inc., 2001.

[6] B. Blalock and P. Allen, “A low-voltage, bulk-driven mosfet current mirror for cmos technology,” in *Circuits and Systems, 1995. ISCAS ’95., 1995 IEEE International Symposium on*, vol. 3, Apr 1995, pp. 1972–1975 vol.3.

[7] S. Rajput and S. Jamuar, “Low voltage analog circuit design techniques,” *Circuits and Systems Magazine, IEEE*, vol. 2, no. 1, pp. 24–42, First 2002.

[8] F. Khateb, D. Biolek, N. Khatib, and J. Vavra, “Utilizing the bulk-driven technique in analog circuit design,” in *Design and Diagnostics of Electronic Circuits and Systems (DDECS), 2010 IEEE 13th International Symposium on*, April 2010, pp. 16–19.

[9] V. Musil, J. Brzobohatý, D. Bečváž, R. Prokop, and D. Ďuračková, *Metodika návrhu anaogových integrovaných obvodů v nových technológiích*, 1st ed. Brno: Vysoké učení technické v Brně, 2004.

[10] S. Huda, “Low voltage, low power, bulk-driven amplifier,” *Electrical Engineering Undergraduate Honors Theses*, May 2009.

[11] B. Tupti and P. Pratik, “Simulation and analysis of bulk driven circuits for low power applications,” *International Journal of Engineering and Technical Research*, vol. 2, February 2014.

[12] M. Rakús, V. Stopjaková, and D. Arbet, “Comparison of gate-driven and bulk-driven current mirror topologies,” in *Design and Diagnostics of Electronic Circuits and Systems (DDECS), 2016 IEEE 19th International Symposium*, April 2016, pp. 27–30.

A ring oscillator based PUF proposal on FPGA

Filip Kodýtek
1st class, full-time study
Supervisor: Róbert Lórencz

CTU in Prague, Faculty of Information Technology
Thákurova 9, 16000, Prague, Czech Republic
kodytfil@fit.cvut.cz

***Abstract*—This contribution deals with design of Physical Unclonable Function (PUF) on FPGA. The goal was to propose a cheap, efficient and secure device identification or even a cryptographic key generation based on PUFs. Therefore, a design of a ring oscillator (RO) based PUF producing more output bits from each RO pair is presented. The design was tested on Digilent Basys 2 FPGA boards (Xilinx Spartan3E-100 CP132) and statistically evaluated. We also discuss its properties and analyse the proposed PUF at varying temperature and voltage. Based on the results of the experiments, we propose suitable modifications of the PUF design in order to improve the quality of its output.**

***Keywords*—Hardware security, physical unclonable function, field-programmable gate array, ring oscillator**

I. Introduction

Field-programmable gate arrays (FPGAs) are used to implement digital circuits of various functionality. Just like other implementation platforms, FPGAs require security and resilience to attacks [9]. For many security protocols, a secret key needs to be stored on FPGA. However, safe storages of keys are usually complicated and expensive to achieve and the nonvolatile memory, in which the keys can be stored, tends to be vulnerable to invasive attacks, because the key is stored in a digital form.

Physical Unclonable Functions (PUFs) offer a solution to this issue. Rather than to store the secret keys in memory, they can be generated using PUFs when they are needed. PUF is a function based on physical properties, which are unique for each device. These unique physical properties can be used to distinguish various devices from each other. Therefore, PUFs can be used for identification purposes and for cryptographic key generation.

Two major groups of PUFs, which are suitable for FPGAs according to their sources of randomness are delay-based and memory-based PUFs. A very common PUF design is based on SRAM and uses it as a source of randomness, since many electronic devices have embedded SRAM [2]. This PUF is based on the content of SRAM after power-up. However some FPGAs initialize their memory after power-up, so all randomness is lost. That led to proposals of other memory-based PUFs such as Butterfly PUF [6], Latch PUF [10] and Flip-flop PUF [8].

Delay-based PUFs exploit the random variations in delays of logic gates and interconnects. One of the first delay-based PUFs is Arbiter PUF [7]. Another examples are Ring Oscillator PUF (ROPUF) [1], [11] and Glitch PUF [12].

In this work we present a ring oscillator based PUF suitable for FPGAs which showed good results in terms of good statistical properties, simplicity and efficiency. One of the advantages of the proposed PUF design is the fact that it is easy to implement, area efficient, and additionally it does not require all ROs to be mutually symmetric, in contrast with the classical approach [1] where all ROs are mutually symmetric and the PUF output is derived from the comparison of RO frequencies of various RO pairs. However, as it is shown in experimental results in Chapter IV, when the symmetric ROs are used in our design, it enhances stability of the proposed PUF design when the physical conditions are varying.

This paper is organized as follows. Section II provides a brief description of the ROPUF, that was proposed in [4]. The performance metrics that are used to evaluate the PUF are described in Section III. Section IV presents the results of experiments. The last Section V concludes the paper.

II. The proposed ROPUF

In this section we give a brief description of the proposed ROPUF architecture. More detailed description of the main concept of this ROPUF is provided in our previous work [3], [4]. In the first part of this section, we explain the main principle of this ROPUF. Then, the proposed ROPUF circuit is described and ultimately, some modifications of the ROPUF design are presented.

A. The main principle of the proposed ROPUF

The main motivation of this proposal was the simplicity of implementation and more efficient use of ROs. In the classical approach [11], the frequencies of ROs are compared and the result of this comparison produces one output bit for PUF. In order to achieve unpredictability of the PUF outputs, this approach requires all ROs to be mutually symmetric so that the differences in frequencies of ROs are influenced only by the random variations in delays of logic gates and interconnects. As also mentioned in [11], the number of pairs of ROs for this comparison is limited, so that the bits in the PUF outputs are independent.

In our ROPUF proposal, a different technique than frequency comparison is used to generate PUF output. The PUF outputs are still obtained based on the selected RO pairs,

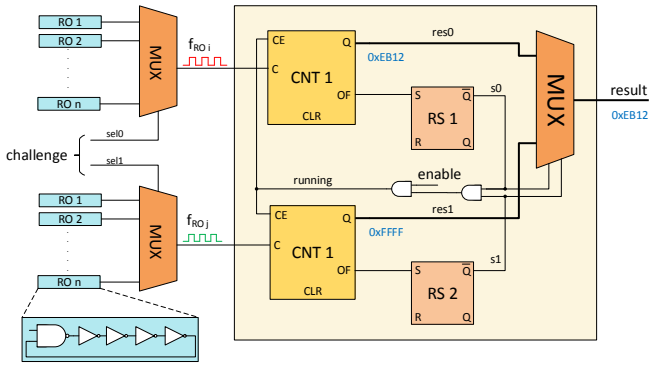


Fig. 1: The proposed ROPUF circuit.

but the problem of selecting particular RO pairs is no longer present. In addition, more bits for the PUF output are gained from each pair of ROs and this technique also does not require all ROs to be mutually symmetric. This allows us to produce longer PUF outputs using less ROs.

The basic building element of the proposed ROPUF is an ordinary five stage RO composed of 1 NAND gate and 4 inverters. Instead of measuring frequency of each RO using some reference clock, we choose one pair of ROs and count the oscillations of each RO simultaneously using two counters. As soon as one of these counters overflows, the measurement is stopped and the resulting value in the counter that did not overflow is used for further processing. This approach is shown in Fig. 1. There are two sets of ring oscillators and they are all enabled and running during the measurement. The overflow detection logic is realised by two RS Flip-flops. When implementing the logic for detecting overflow of one of the counters and stopping the other one, the routes between them may have different delays and before the second counter is stopped, it can perform some additional steps. But since these two routes are the same for all RO pairs and for all FPGAs, it will only increase the resulting value by some constant offset which is 0 or 1 in this case.

The proposed method implies that if we knew the exact frequencies of the ROs during measurement, we could determine the resulting counter value (in case of 16-bit counters) that is later processed as follows:

$$Counter\ value = \frac{f_2}{f_1} \times 2^{16}, \tag{1}$$

where f_1 is the frequency of the faster RO and f_2 is the frequency of the slower RO.

Since the obtained counter values are represented in binary code, we can use the appropriately selected part of each binary number for the PUF output. It can be assumed, that if we repeat the measurements for one RO pair, the bits that are close to the least significant bit (LSB) will vary a lot due to instability of ROs and the environmental changes. On the contrary, the bits close to the most significant bit (MSB) will be stable and the environmental changes will have almost no influence on them. The more we will be close to the MSB, the more stable the bits will be.

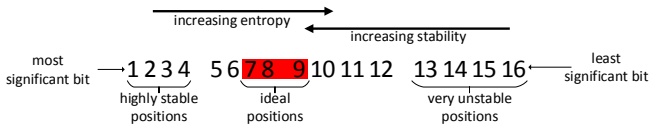


Fig. 2: The example behaviour of the bits in counter value of a 16-bit value.

Another requirement in addition to stability that needs to be met is the entropy of the selected bits. We may assume, that if we compare the measured values from two equally positioned RO pairs on two FPGAs, bits close to the MSB will not differ, while the bits that are approximately in the middle between the MSB and the LSB will be different. The bits close to the LSB will be different too, but it is caused mainly by their instability. Therefore, the ideal positions of the counter value that should be used for PUF are in the middle of the counter value. The example of described behaviour of measured counter values is shown on 16-bit counter value in Fig. 2.

B. Modifications of the proposed ROPUF

In order to eliminate some of the issues present in the original design, we proposed some modifications of the design that enhance the properties of the PUF.

The first improvement of our PUF design is the application of Gray code on the obtained counter values. One of the issues of selecting a block of bits from each counter value is that all of the selected bits may change in the next measurement since they are represented in binary code and they are a part of a counter value. So even if the final value is increased or decreased just by one, all of the bits can be influenced by this change. The first step to solving this issue involves the application of Gray code to the obtained counter values. The reason for using Gray code is the fact that two successive values differ in only one bit, so this can eliminate the partial overflow and increase the overall stability of the selected bits and even increase the number of extractable bits from each counter value. For more details see [3], [5].

The next improvement is related to the issue of the influence of various physical conditions on the behaviour of the PUF. Since the PUF design is based on ROs, the physical conditions will have a significant impact on the frequencies of ROs, however, our goal is to make the differential measurement (frequency ratio) more robust against such effects. Therefore, we present a possible solution to this problem using symmetric ROs. More detailed description of both of the modifications is provided in [3], [5].

III. Performance Metrics

To evaluate the quality of PUF, we need some metrics in order to evaluate its statistical properties. In our previous work [4], we discussed how to select good positions of the counter values for the PUF based on their statistical properties, such as stability and entropy. After selecting the suitable positions, the PUF outputs made of these positions need to be evaluated by some additional parameters to validate the selection of positions. In this section we describe the

TABLE I: The results of statistics carried out for responses composed from various bit selections for 150 RO pairs.

150 pairs of ring oscillators

positions	6–8	7–8	7–9	7–10	8–9
w [-]	3	2	3	4	2
HD_{intra} [%]	1.44	2.16	2.81	4.21	4.21
HD_{inter} [%]	43.05	48.81	49.23	49.45	49.88

evaluation method, which we used to determine the qualities of the PUF outputs. We review two common parameters that are used to evaluate the properties of PUFs, namely *Intra-Hamming distance* (HD_{intra}) and *Inter-Hamming distance* (HD_{inter}).

A. Intra-Hamming distance

To evaluate the mutual similarity of the PUF outputs, we use Intra-Hamming distance as a metric. HD_{intra} is estimated as:

$$HD_{intra} = \frac{1}{m \times k} \sum_{i=1}^{m} \sum_{j=1}^{k} HD(R_{r_i}, R_{i,j}) \times 100 \ [\%], \quad (2)$$

where m is the number of FPGAs, R_{r_i} is the reference output of the i-th PUF, which the other outputs are compared to, and k is the number of compared outputs from each PUF. As R_{r_i}, we can use either any output from the given PUF or the mean output of several outputs (this may result in lower HD_{intra}). There are several influences that affect the value of HD_{intra} such as changes in voltage or temperature which cause HD_{intra} to be of higher value.

B. Inter-Hamming distance

Another important metric that is used to evaluate the PUF quality is the uniqueness of the generated outputs among different FPGAs. We can determine the uniqueness of the generated outputs by calculating the Inter-Hamming distance, which is defined as:

$$HD_{inter} = \frac{1}{\binom{m}{2}} \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} HD(R_{r_i}, R_{r_j}) \times 100 \ [\%], \quad (3)$$

where m is the number of FPGAs and R_{r_i} is the reference output of the i-th PUF which is the mean output made of all outputs from the given PUF.

IV. Experimental results

In this section we present the results of performed measurements on Digilent Basys 2 FPGA boards (Xilinx Spartan3E-100 CP132). At first, the results for PUF with symmetric ROs that uses different selections of positions from counter values with applied Gray code are presented.

A. Evaluation of ROPUF with symmetric ROs

Table I presents the results of statistical evaluation of the PUF outputs for symmetric ROs measured 1000 times for 150 RO pairs and with Gray code applied to the selected parts of the counter values. The results indicate, that the proposed ROPUF with symmetric ROs still works correctly and enables us to reliably distinguish different FPGAs. However, the results of HD_{intra} for symmetric ROs are slightly worse than for asymmetric ROs (Table 2 in [4]).

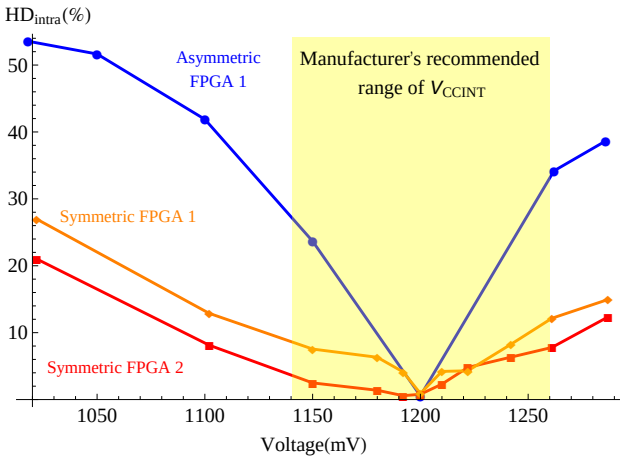


Fig. 3: Comparison of the behaviour of the proposed PUF when using mutually symmetric and asymmetric ROs for positions 7–8. The reference output for calculating HD_{intra} is the mean output from the PUF outputs measured at nominal voltage 1.2V. Yellow area represents the manufacturer's recommended range of FPGA's main power supply voltage V_{ccint}, which is from 1.14V to 1.26V.

B. Influence of supply voltage

The next measurement concerns the influence of voltage on the behaviour of the proposed ROPUF design. The measurements were performed on 2 Digilent Basys 2 FPGA boards containing Xilinx Spartan3E-100 CP132. The main power supply for the FPGA's internal logic is V_{ccint} and its nominal voltage is 1.2V. The maximum ratings for V_{ccint} are -0.5V and 1.32V, with manufacturer's recommended range from 1.14V to 1.26V. The circuit remains the same and the results presented in Fig. 3 relate to 1000 measurements for 150 RO pairs and show how the PUF outputs are different from nominal voltage, which is 1.2V. The range of tested voltages is from 1.018V to 1.286V and the selected positions of counter values for the PUF outputs are 7–8.

It can be seen in Fig. 3 that the influence of voltage is significant in case of asymmetric ROs. This is caused by the change of ratios of two frequencies of ROs in each pair. If we want the PUF outputs to remain stable with varying voltage, the ratios of the frequencies for each pair have to be the same at any voltage level. We can expect that the frequencies of ROs will change in a similar way when the ROs are mutually symmetric. Therefore, we placed the RO gates so that all ROs are mutually symmetric.

Fig. 3 presents the comparison of the behaviour of the proposed ROPUF when using mutually symmetric and asymmetric ROs for positions 7–8. The results for HD_{intra} are not ideal, but they demonstrate the improvement when using symmetric ROs compared to asymmetric ROs and show the

TABLE II: Evaluation of HD_{intra} for 150 asymmetric/symmetric RO pairs and selected positions 7–8 and different temperatures.

Asymmetric ROs					
FPGA 1		FPGA 2		FPGA 3	
Temperature [°C]	HD_{intra} [%]	Temperature [°C]	HD_{intra} [%]	Temperature [°C]	HD_{intra} [%]
36.7 → 41.2	2.67	38.4 → 42.3	2.67	37.7 → 41.8	1.0
36.7 → 51.8	7.67	38.4 → 50.1	6.67	37.7 → 50.9	5.0
36.7 → 60.4	9.33	38.4 → 60.3	9.33	37.7 → 61.3	7.0
36.7 → 71.1	11.33	38.4 → 69.9	12.67	37.7 → 70.1	12.0
Symmetric ROs					
FPGA 1		FPGA 2		FPGA 3	
Temperature [°C]	HD_{intra} [%]	Temperature [°C]	HD_{intra} [%]	Temperature [°C]	HD_{intra} [%]
33.0 → 42.4	2.67	34.4 → 40.9	1.67	34.5 → 41.1	3.67
33.0 → 50.5	3.67	34.4 → 50.5	3.0	34.5 → 51.4	6.0
33.0 → 60.6	3.67	34.4 → 60.8	4.67	34.5 → 60.6	7.0
33.0 → 71.0	4.67	34.4 → 70.2	5.33	34.5 → 70.4	7.33

way for further investigation of the influence of the placement of ROs on the stability of the PUF outputs.

C. Influence of temperature

This subsection examines the influence of change of temperature on the proposed ROPUF. The statistical properties of PUF using both symmetric and asymmetric ROs will be compared. For the purpose of our experiment, we performed measurements at different temperatures. For these measurements, FPGA was preheated to a preset temperature (e.g. 40 °C) with ROs enabled. Each of the measurements was carried out when the temperature measured on the package of the FPGA stabilized at the given value. We used 3 Digilent Basys 2 FPGA boards for this experiment.

Table II displays the values of HD_{intra} for 3 FPGAs for asymmetric and symmetric ROs. The column temperature presents the temperatures, at which the PUF outputs are compared. The values of HD_{intra} for symmetric and asymmetric ROs are almost equivalent for small differences in temperature, but for larger changes of temperature, there is a visible improvement of the PUF behaviour, when symmetric ROs are used.

V. Conclusion

We proposed a RO based PUF, which is able to provide more output bits from each pair of ROs and is also not dependent on the symmetry of ROs, implying that it is easy to implement. However, as it was shown in Section IV, the proposed PUF exhibits better behaviour at varying physical conditions in terms of stability when symmetric ROs are used.

In our future work, we would like to build a statistical model of ring oscillators and eventually model the whole ROPUF. This would be useful to evaluate the PUF and also a true random number generator, which can be based on the same circuit (also future work). Then we would like to examine the influence of aging on this ROPUF and further investigate the influence of supply voltage and temperature together with the placement of ROs. Our goal is to modify the PUF design, so that it will be resilient to environmental changes as much as possible. This is one of the conditions in order to generate cryptographic keys of sufficient length using the proposed ROPUF.

References

[1] Bossuet, L., Ngo, X. T., Cherif, Z., Fischer, V. A PUF based on a transient effect ring oscillator and insensitive to locking phenomenon. In *IEEE Transactions on Emerging Topics in Computing*, pages 30–36, March 2014.

[2] Holcomb, D. E., Burleson, W. P., Fu, K. Power-Up SRAM State as an Identifying Fingerprint and Source of True Random Numbers. *IEEE Transactions on Computers 58, 9*, pages 1198-1210, 2009.

[3] Kodýtek, F. *Behaviour Analysis and Improvement of the Proposed PUF on FPGA*. Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2016.

[4] Kodýtek, F., Lórencz, R. A design of ring oscillator based PUF on FPGA. In *IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems - DDECS 2015*. Belgrade, RS, April 2015.

[5] Kodýtek, F., Lórencz, R., Buček, J. Improved ring oscillator PUF on FPGA and its properties. In *Microprocessors and Microsystems*. 2016.

[6] Kumar, S., Guajardo, J., Maes, R., Schrijen, G.-J., Tuyls, P. Extended abstract: The Butterfly PUF Protecting IP on Every FPGA. In *IEEE International Symposium on Hardware-Oriented Security and Trust - HOST 2008*, pages 67–70. IEEE, Washington, DC, USA, 2008.

[7] Lee, J. W., Lim, D., Gassend, B., Suh, G.E., van Dijk, M., Devadas, S. A technique to build a secret key in integrated circuits for identification and authentication applications. In *Symposium on VLSI Circuits - VLSIC 2004*, pages 176–179, June 2004.

[8] Maes, R., Tuyls, P., Verbauwhede, I. Intrinsic PUFs from Flip-flops on Reconfigurable Devices. In *Benelux Workshop on Information and System Security - WISSec 2008*. Eindhoven, NL, 2008.

[9] Majzoobi, M., Koushanfar, F., Devadas, S. FPGA PUF using Programmable Delay Lines. In *Information Forensics and Security*. December 2010.

[10] Su, Y., Holleman, J., Otis, B. A 1.6pJ/bit 96% Stable Chip-ID Generating Circuit using Process Variations. In *IEEE International Solid-State Circuits Conference - ISSCC 2007*, pages 406–611. IEEE, February 2007.

[11] Suh, G. E., Devadas, S. Physical Unclonable Functions for Device Authentication and Secret Key Generation. In *Design Automation Conference - DAC 2007*, pages 9–14. ACM, New York, NY, USA, 2007.

[12] Suzuki, D., Shimizu, K. The Glitch PUF: A New Delay-PUF Architecture Exploiting Glitch Shapes. In Workshop on Cryptographic Hardware and Embedded Systems - CHES 2010, pages 366–382. Springer, 2010.

Optimalizácia spotreby energie komunikačného modulu v implantovateľných senzorických systémoch

Šimon Danko
1. ročník, denná prezenčná forma štúdia
Školiteľ: Viera Stopjaková

Fakulta elektrotechniky a informatiky, Slovenská technická univerzita v Bratislave
Ilkovičova 3, 812 19 Bratislava
simon.danko@stuba.sk

***Abstrakt*—Tento príspevok sa zaoberá analýzou dostupných riešení autonómneho napájania implantovateľných systémov s RF komunikačným modulom. Tiež obsahuje opis ultra-wideband komunikácie, ktorá bola vybraná pre predstavovaný koncept aktívnych bio-senzorických implantátov, vzhľadom na jej nízku energetickú a hardvérovú náročnosť. V závere sú opísané možné prínosy práce ako aj jej ďalšie smerovanie.**

Kľúčové slová — nízko-príkonové obvody; implnatovateľné senzory; bezdrôtová komunikácia; Body Area Network; Ultra-wide Band; monitorovanie zrdavotného stavu

I. Úvod

Senzorické elektronické zariadenia implantované do ľudského tkaniva musia spĺňať špeciálne požiadavky napr. biokompatibilita použitých materiálov, veľmi nízka spotreba energie, bezpečnosť prenášaných dát, či priestorové nároky systému zaručujúce jeho miniaturizáciu. So súčasným hardvérom a technológiou batérií môžu tieto zariadenia operovať autonómne po dobu niekoľkých rokov iba s veľmi malou pracovnou striedou alebo s veľkou batériou. Pod pojmom striedа v tomto prípade rozumieme pomer medzi časom kedy je implantát aktívny a spotrebúva energiu a dĺžkou časového úseku, za aký meriame, t.j. periódou. Nízka striedа znamená, že zariadenie je aktívne počas krátkeho časového úseku (meranie, odosielanie údajov, atď.) a zvyšný čas je v režime spánku kedy je jeho spotreba minimálna, ideálne nulová. Tento cyklus sa pravidelne opakuje s istou periódou. Z pohľadu spotreby energie je kritickou časťou systému bezdrôtový komunikačný modul a energia potrebná na prenos dát. Pre zvýšenie životnosti implantátov je možné využiť vstavané zberače energie, ktoré sú schopné získať potrebnú energiu priamo z ľudského tela. Preto existuje rozsiahly výskum na poli zberačov energie a nízko-príkonových obvodov. Takéto zariadenia, schopné získavať energiu zo svojho okolia sa nazývajú aktívne bio-senzorické implantáty (ABSI).

Moderné implantovateľné zariadenia sa spoliehajú na nabíjanie internej batérie a prenos údajov magnetickou indukciou za použitia dvoch cievok tvoriacich vzduchový transformátor. V mnohých prípadoch dokonca batéria nie je nabíjateľná a každé tri až päť rokov je nutné ju vymeniť počas invazívneho operačného zákroku. [1] Takýto prístup k prenosu energie a dát však vyžaduje, aby budiacia a prijímacia cievka boli v bezprostrednej blízkosti (napr. v kochleárnych implantátoch), čo môže obmedzovať pacienta pri pohybe, či brániť vo výkone iného zákroku. Preto sa rádiová komunikácia (RF) javí ako vhodnejšia alternatíva pre prenos informácií z a do implantátu.

II. Získavanie energie

Ako ich názov napovedá, zberače energie sú schopné získať energiu z okolia (príp. ľudského tela) a premeniť ju na využiteľnú elektrickú energiu. Existuje viacero prístupov k získavaniu energie. V tabuľke 1 sú uvedené niektoré príklady aj s účinnosťou prevodu energie a ich výkonovou hustotou [2-5].

Tabuľka 1: Zdroje elektrickej energie z okolia a ich nominálna účinnosť premeny energie.

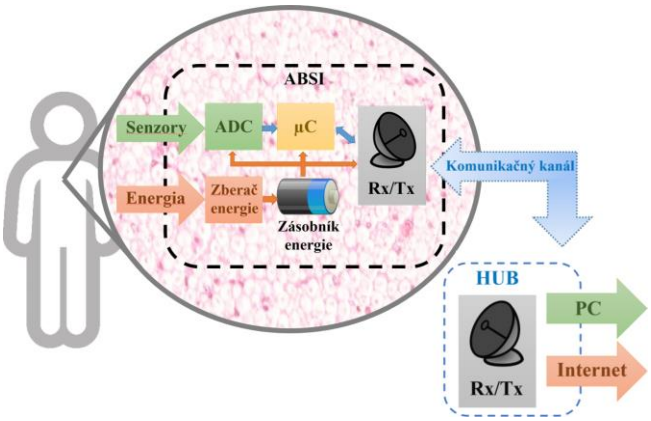
Zdroj energie	Zberač	Účinnosť [%]	Energetická hustota [$\mu W/cm^2$]
Svetlo	solárny článok	10 - 20	$0{,}15 – 15.10^3$
Pohyb / vibrácie	piezoelektrický generátor	20 - 50	200
	elektrostatický generátor		50 - 100
	elektromagnetický		1
Tepelný tok	Peltierov článok	1 - 5	15
Elektro-magnetické žiarenie	GSM (900 MHz)	50	0,1
	Wifi (2.4 GHz)		0,001

Z tabuľky 1 je vidieť, že výkon, ktorý dokážu zberače dodať je rádovo v mikrowatoch a tomu treba prispôsobiť celý koncept. Taktiež si môžeme všimnúť, že piezoelektrický generátor poskytuje najviac výkonu vzhľadom na rozmery zberača. Tento fakt potvrdzuje aj prezentácia flexibilného tenkovrstvového piezoelektrického PIMNT článku, ktorý podľa [1] dokázal vygenerovať z pohybu laboratórnej myši 0,7 *mW* a úspešne tak napájať implantovaný DBS. Deklarované parametre tohto článku sú 11 *V* naprázdno a 0,57 *mA* nakrátko.

III. KONCEPT ABSI

A. *Dostupné riešenia*

Súčasné riešenia dlhodobého monitorovania zdravotného stavu pacienta sú založené na pravidelných prehliadkach a vyšetreniach. Zariadenie na kontinuálne sledovanie stavu (napr. srdcovej činnosti po infarkte) sa nazýva holter. Je to externé zariadenie pripojené na elektródy prilepené na hrudi pacienta. Holter monitoruje stav pacienta 24 až 72 hodín a následne sa jeho stav vyhodnocuje. Okrem káblov na pripojenie elektród zvyšuje nepohodlie pacienta aj fakt, že počas doby kedy je pacient monitorovaný sa nemôže kúpať ani sprchovať. ABSI ponúkajú možnosť sledovať stav pacienta kontinuálne a dlhodobo, bez potreby vonkajšieho zariadenia. Implantát v tele neobmedzuje pacienta a môže odosielať dáta o zdravotnom stave pacienta pravidelne cez internet prostredníctvom zabezpečeného protokolu, alebo iba monitorovať stav a informovať o prípadnom zhoršení. Na obr. 1 je znázornený koncept riešenia pre ABSI.



Obr. 1 Koncept navrhovaného riešenia ABSI.

B. *Telová sieť - Body Area Network*

Podľa štandardu IEEE 802.15 (802.15.4a, 802.15.6) je telová sieť (ďalej len BAN) definovaná ako komunikačný štandard optimalizovaný pre nízko-výkonové zariadenia operujúce v okolí, v tesnej blízkosti alebo vo vnútri ľudského tela slúžiaci na rôzne účely, ako napr. zábava alebo medicína. BAN umožňuje aj komunikáciu viacerých nezávislých implantátov medzi sebou.

Tabuľka 2: Vybrané informácie zo štandardu IEEE 802.15.6

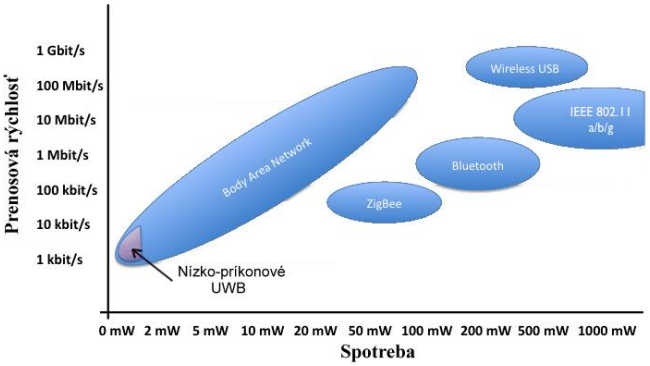
Vybrané údaje a vlastnosti z štandardu IEEE 802.15.4a	
Vzdialenosť	2 *m* - štandardne 5 *m* - v špeciálnych prípadoch
Hustota sietí	2 – 4 m^{-2}
Veľkosť siete	Max. 100 zariadení
Spotreba energie uzla	1 *mW/Mbps*
Globálne pásmo bez potreby licencie	
Efektívny režim spánku	
Peer-to-Peer alebo Multi-point	

ABSI odosiela údaje do centrálneho bodu (tzv. HUB), ktorý následne môže dáta vyhodnotiť alebo ich odoslať na ďalšie spracovanie. Počas doby kedy je zariadenie mimo siete môže zaznamenané dáta ukladať vo vnútornej pamäti a po opätovnom pripojení ich odoslať. Ďalšou alternatívou je smartfón, ktorý môže plniť úlohu HUB-u.

C. *Širokopásmová komunikácia*

Aj keď UWB (z angl. Ultra-wideband) je technológia známa vyše 40 rokov jej potenciál pre komunikáciu sa stretol zo záujmom až v posledných rokoch. UWB je technológia prenosu veľkého objemu digitálnych dát v širokom spektre (>500 *MHz*) frekvencií s veľmi malým výkonom a na krátku vzdialenosť. Takýto signál má schopnosť prenikať cez prekážky, ako napríklad steny a dvere, ktoré inak odrážajú signály v užšom pásme a s väčšou energiou. Je porovnateľná s Bluetooth technológiou, ktorá sa stala štandardom pre pripojenie mobilných zariadení a počítačov. Na rozdiel od komunikácie v rozptýlenom spektre, UWB neinterferuje s konvenčnou úzkopásmovou komunikáciu na rovnakých frekvenciách vďaka jej nízkej spektrálnej hustote energie (< - 40 *dBm/MHz*). Presnosť, nízky výkon a krátky čas prenosu činí UWB vhodným typom komunikácie v prostrediach citlivých na elektromagnetické žiarenie, ako napr.: nemocnice a iné medicínske zariadenia. Širšie spektrum signálu ho činí odolnejším voči rušeniu a viaccestnému šíreniu čím sa zvyšuje pravdepodobnosť správneho prenosu. UWB je možné použiť v pozičných systémoch reálneho času [6].

Z obr. 2 je vidieť, že UWB technológia poskytuje veľkú flexibilitu ako po stránke prenosovej rýchlosť, tak aj energie potrebnej na prenos dát. Aj preto bola zvolená pre zariadenie napájané energetickými zberačmi [7].

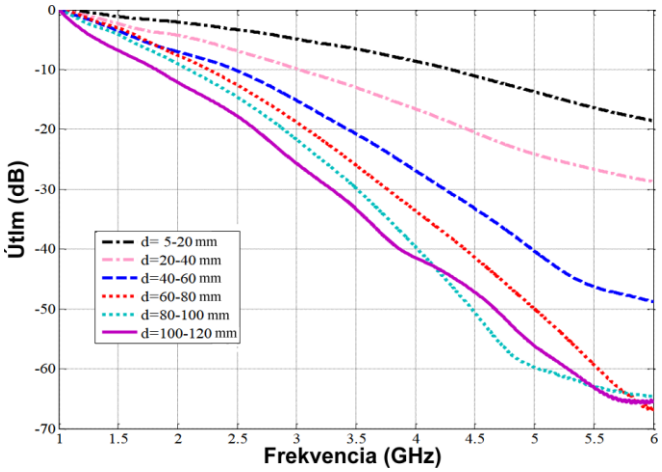


Obr. 2: Porovnanie UWB a iných technológií z pohľadu prenosovej rýchlosti a spotreby energie.

D. *Komunikačný kanál*

Rádiové vlny musia byť schopné preniknúť z tela do vonkajšieho prostredia a naopak. Prostredie cez, ktoré vlnenie prechádza od vysielača k prijímaču sa nazýva komunikačný kanál. Živé tkanivo má značný útlm na požadovaných

frekvenciách. Na obr. 3 je znázornená závislosť útlmu signálu od hĺbky implantátu v hrudi človeka. [8] Je zrejmé, že s rastúcou hĺbkou a frekvenciou stúpa útlm v tkanive. Tento fakt je potrebné brať do úvahy pri návrhu komunikačného modulu z pohľadu výkonu.



Obr. 3: Závislosť útlmu tkaniva od frekvencie pre rôzne hĺbky implantátu v hrudi človeka smerom k povrchu [8].

IV. Komunikačný modul

Komunikačný modul bude pozostávať z prijímača na frekvencii 2,4 GHz a IR-UWB vysielača. Tým sa vyhneme značnej zložitosti UWB prijímača v ABSI a docieli sa tak zníženie plochy čipu a vlastnej spotreby prijímača. Na druhej strane, HUB bude pozostávať z úzkopásmového vysielača 2,4 GHz a IR-UWB prijímača. Pre túto nesymetrickosť liniek pri implantovateľných senzorických zariadeniach je predpokladom, že implantát odosiela väčšie množstvo dát ako prijíma. Samotný modul bude pozostávať z dvoch základných častí, digitálny back-end a analógový front-end. Vzhľadom na celkový koncept ABSI a výslednú aplikáciu sme zvolili anténu integrovanú na čipe.

V. Súčasné riešenia

Ako jedno zo súčasných riešení v tejto oblasti uvádzame miniatúrny jednokomorový pacemaker Micra od firmy Meditronic Inc. Ide o 25,9 mm dlhý implantát o hmotnosti 1,75 g obsahujúci celý hardvér a batériu. Prístroj má deklarovanú životnosť 10 rokov. Pritom kapacita batérie je 120 mAh pri nominálnom napätí 3,2 V [9]. Na obr. 4 je znázornený práve spomínaný pacemaker.

Obr. 4: Jednokomorový kardiostimulátor Meditronic Inc. Micra

Okrem udržiavania frekvencie srdca, Micra ukladá denné merania po dobu 15 dní a ďalších 80 záznamov o týždennom maxime a minime frekvencie činnosti srdca. Po prekročení tejto doby sa najstarší záznam vymaže nahradí novým [9].

VI. Zámer dizertačnej práce a jej ciele

Ako už bolo spomenuté, hlavným zámerom budúceho výskumu bude návrh a optimalizácia spotreby komunikačného modulu v pásmach UWB a 2,4 GHz ISM. Ten má v súčasnosti najvyššiu spotrebu zo všetkých častí ABSI. Výber vhodného kľúčovania má dopad nie len na samotnú energiu potrebnú pre prenos informácií, ale taktiež aj priepustnosť kanála a maximálnu hĺbku umiestnenia implantátu. Spôsob riadenia napájania jednotlivých blokov je taktiež dôležitým aspektom a bude kladený dôraz na čo najmenšiu striedu pri zachovaní dostatočne rýchleho vzorkovania signálov. Ďalej budú preskúmané metódy návrhu nízko-príkonových obvodov pre RF aplikácie. Táto práca počíta s veľmi malým množstvom dostupnej energie získavanej využitím vstavaných zberačov energie, rádovo 1 až 10 μW. Pri všetkých týchto aspektoch sa bude okrem spotreby hľadieť aj na minimalizáciu plochy čipu.

Tento výskum má ďalej umožniť kompletnú integráciu celého ABSI počnúc ADC prevodníkom, mikroprocesorovou jednotkou, zberačom energie a končiac komunikačným modulom aj s integrovanou anténou.

VII. Záver

V tejto práci sme predstavili koncept energeticky autonómneho bio-senzorického implantátu, ktorého úlohou je monitorovanie vitálnych funkcií a odosielanie dát na spracovanie. Zníženie spotreby a využitie energetických zberačov umožní ABSI pracovať kontinuálne a dlhodobo bez potreby relatívne rozmernej batérie, či nutnosti ju dobíjať, prípadne vymeniť počas operačného zákroku.

Monitorovanie a pravidelné odosielanie informácií o zdravotnom stave pacienta a ich vyhodnotenie aj z pohodlia domova zvýši pacientov komfort. Zber údajov v reálnom čase môže okamžite poukázať na prípadné zhoršenie zdravotného stavu.

Hlavným smerovaním ďalšej práce bude návrh komunikačného modulu a následná optimalizácia jeho spotreby. Samotný prenos údajov bude založený na RF komunikácii v UWB pásme postavenom na topológii siete typu BAN. Integrácia všetkých potrebných komponentov aj s anténou na čip zmenší rozmery ABSI a zabezpečí energetickú autonómnosť celého systému.

POĎAKOVANIE

Táto práca bola podporená projektom APVV-15-0245.

REFERENCIE

[1] Geon-Tae Hwang, et al. Self-powered Deep Brain Stimulation via a Flexible PIMNT Energy Harvester. Energy Environ. Sci., 2015.

[2] R. Vullers, R. Schaijk, H. Visser, J. Penders, and C. Hoof. Energy Harvesting for Autonomous Wireless Sensor Networks. Solid-State Circuits Magazine, IEEE, 2(2):29–38, 2010.

[3] P. Spies, M. Pollak, and G. Rohmer. Power management for energy harvesting applications. 2007.

[4] R. Vullers, R. van Schaijk, I. Doms, C. V. Hoof, and R. Mertens. Micropower energy harvesting. Solid-State Electronics , 53(7):684 – 693, 2009. Papers Selected from the 38th European Solid-State Device Research Conference (ESSDERC 2008).

[5] S. Evanczuk. Low-voltage DC-DC Converters Build Efficient Power Management into Energy Harvesting Designs. Energy Harvesting Solution, 2011.

[6] S. Mohammad-Sajad Sadough, A tutorial on ultra wideband modulation and detection schemes, 2009

[7] IEEE 802.15 working group for WPAN & Task group 6 for BANs. http://www.ieee802.org/15/pub/TG6.html

[8] A. Khaleghi, R. Chávez-Santiago, and I. Balasingham, An improved ultra wideband channel model including frequency-dependent attenuation for in-body communications, 34th Annual International Conference of the IEEE EMBS

[9] Meditronic Inc., špecifikácia produktu Micra. http://www.medtronic.com/content/dam/medtronic-com/products/cardiac-rhythm/pacemakers/micra-pacing-system/documents/2016-04-micra-specification-sh

Case Study on Multi-domain Decomposition of k-Wave Simulation Framework

Filip Vaverka
1st year, full-time study
Supervisor: Jiri Jaros

Brno University of Technology, Faculty of Information Technology
Božetěchova 2, 612 66 Brno, Czech Republic
ivaverka@fit.vutbr.cz

***Abstract*—This article describes both the advantages and challenges of using GPU equipped clusters to implement efficient distributed algorithms. Our main focus is the decomposition of numerical simulations based on (pseudo-)spectral methods. We use a pseudo-spectral simulation of the ultrasound wave propagation in homogenous medium (k-Wave) as a showcase of the distributed algorithm. We show we are able to achieve a three fold speedup of the simulation while lowering the cost at the same time by employing a Local Fourier Basis decomposition.**

***Keywords*—Pseudo-spectral Simulation, Local Fourier Basis, GPGPU, CUDA, GPU Cluster**

I. Introduction

This article describes our plan to approach the design of distributed algorithms utilizing accelerator (such as GPU, MIC, APU or even FPGA) equipped clusters. We can see that one of main roadblocks in pursuit of higher performance of our super-computer systems is the efficiency of compute nodes and inter-node communication bandwidth. The first of those problems is usually tackled in two ways: we employ more efficient processing elements (to this date GPUs can achieve up to 8x FLOPs/W more than general purpose CPUs) and we design efficient algorithms on these processing elements.

The second problem (which is actually more painful) is the inter-node communication, which is usually several times slower than the intra-node communication. Even worse, the inter-node communication is evolving rather slowly and the discrepancy between the performance of the node and the interconnect bandwidth is growing. On the top of that, there is a plenty of tools for optimizing computation itself (compilers, profilers, etc.), yet there is significantly fewer tools to optimize communications and data locality. In distributed memory environments, we have to manually optimize the communication or employ shared memory emulation [4], which is usually unaware of the algorithm running on top of it.

The following sections briefly touch on current and near future architectures of accelerator equipped clusters (mainly GPU accelerated). After that, we show a case study of the multi-domain decomposition of the k-Wave simulation framework, and finally, roughly outline possible directions of following research.

II. Modern Accelerated Cluster Architectures

The basic building block for the current cluster architectures are multi-core CPU based nodes. Each of these nodes then has a few hundreds GBs of a coherent memory (usually NUMA architecture) and a fast network adapter. We can call such a cluster "flat" as it has mostly (ignoring CPU caches) single, flat and coherent memory space.

A simple way to build an accelerated cluster is to add an accelerator (GPU, MIC, FPGA, ...) to each node (usually connected through the PCI-E bus). As both PCI-E and the main memory are rather slow, accelerators usually have their own on-board, high bandwidth, memory with a limited size. Figure 1 shows the architecture of such a cluster (using dual socket Intel Xeon E7 CPU and R9 Fury X GPU). The on-board memory is usually not coherent with the main memory, therefore, the node memory space becomes hierarchical (with manual management). This hierarchical structure brings new challenges to the efficient use of such architectures.

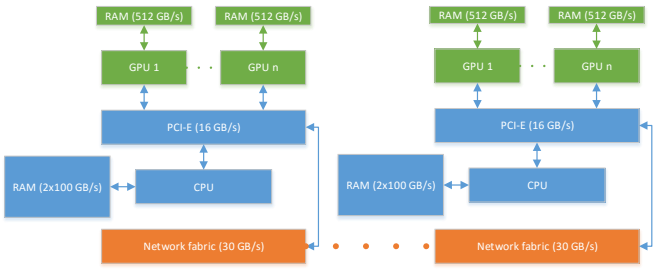


Fig. 1: GPU Accelerated Cluster

In future, we expect an increase in the bandwidth of the intra-node interconnect (NVlink and AMD GMI) and maybe even fully coherent memory, yet these improvements will probably lag behind the increase of the compute power of accelerators. There are also efforts to achieve tighter integration of accelerators with CPUs (NVlink + Power8 [9] and APUs [8]). All this may result in effortless memory coherency of the whole accelerated node, yet performance-wise we expect the hierarchical structure to remain present (in similar way as in current NUMA architectures).

III. Pseudo-spectral Simulation Approach

To show challenges associated with using such clusters we chose a GPU based pseudo-spectral ultrasound wave propagation solver as a case study of an algorithm with non-trivial decomposition.

Computationally, the most intensive part of the solver is the evaluation of the spatial derivatives which is usually calculated, by either the Finite Difference Methods (FDM [10]) or (in our case) pseudo-spectral methods [11]. The pseudo-spectral solver was chosen (a) because it's a widely spread solution and (b) its decomposition is non-trivial (compared to FDM). Big advantage of spectral methods is high accuracy (due to their ability to achieve exponential convergence).

The most often used pseudo-spectral methods are based on the Fourier transform and the derivative can be computed in Fourier basis space (i.e. $f'(x) = \mathbb{F}^{-1}\{ik\mathbb{F}\{f(x)\}\}$), where k is a matrix of wave numbers and i is the imaginary unit). The primary advantage of this approach is that we know derivatives of the basis functions exactly and therefore we are getting an advantage in precision. According to [1], our simulation needs up to an order of magnitude fewer grid points (in 1D case) to achieve accuracy on par with FDM. The secondary advantage is that the Fourier transform can be computed in $O(nlog(n))$ time by the Fast Fourier Transform.

A. Local Fourier Basis

One of a limited number of approaches to alleviate the all-to-all communication burden caused by the k-dimensional FFT computation is the use of the Local Fourier Basis (LFB) decomposition of the simulation domain (so called Multi-domain Decomposition). This method is based on subdivision of the original domain into a number of independent sub-domains. If neighboring sub-domains have a certain overlap and each of them form an LFB (i.e. local functions are smooth and periodic) then the derivatives computed locally approximate the global ones. Figure 2 shows such a decomposition of the simulation domain in 3D.

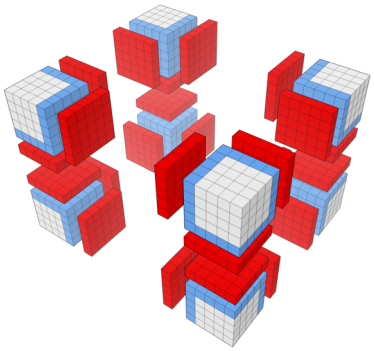


Fig. 2: Multi-domain decomposition

In order to form an LFB after splitting the original domain, we adopted a method described in [2]. Smoothing is done by multiplying a function with a so called bell function (Fig. 3) defined by equations 1 and 2.

$$\theta(x) = \sqrt{\frac{\pi}{2\epsilon}\left[1 + \mathrm{erf}\left(\frac{x\sqrt{\epsilon}}{\sqrt{2}}\right)\right]} \tag{1}$$

$$B(x) = \begin{cases} \theta(x) & \text{if } x \in \langle a_i - \epsilon, a_i) \\ 1 & \text{if } x \in \langle a_i, a_{i+1}) \\ \theta(\epsilon - x) & \text{if } x \in \langle a_{i+1}, a_{i+1} + \epsilon) \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Where ϵ is the depth of the overlap and $\langle a_i, a_{i+1}\rangle$ is the local core interval of the function.

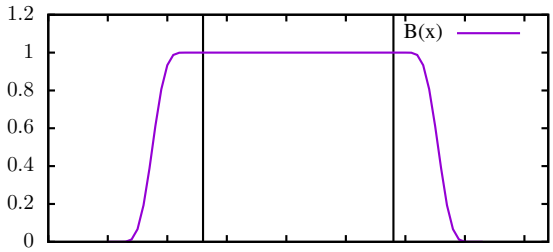
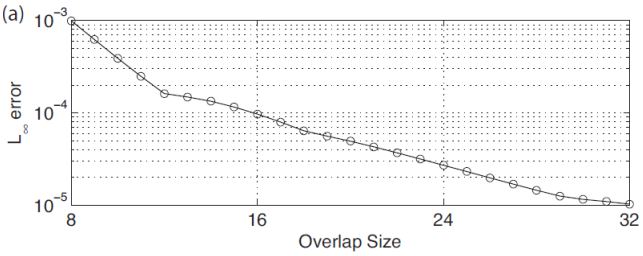
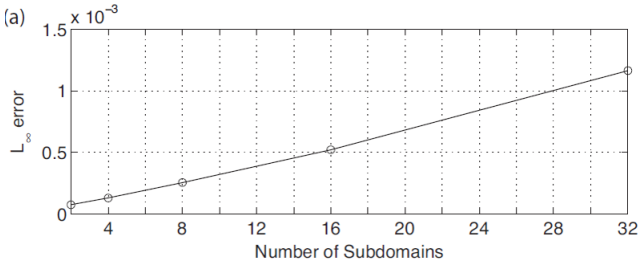


Fig. 3: Bell function

The accuracy of the approximation is given by the number of sub-domains (in a given direction of FFT) and the depth of overlaps. This gives us the opportunity to trade between the overhead (both computation and communication) and the accuracy. The impact of the number of sub-domains and the depth of overlaps on the accuracy shows Fig. 4. Its worth to note that, we can afford error up to 10^{-3}, because that's the accuracy level of the PML which is necessary to avoid periodic behavior of the spectral method.



(a) Depth of Overlaps



(b) Number of sub-domains

Fig. 4: Decomposition error

B. K-Wave Local Decomposition

The proposed k-Wave simulation decomposition was implemented using OpenMPI for inter-process communication, CUDA to access GPUs and parallel HDF5 for I/O. We used cuFFT to compute local gradients on each GPU. The whole implementation can be divided into three parts: Simulation core, communication framework and I/O subsystem.

The simulation core is mostly identical with the global single GPU implementation and consists of a few simple CUDA kernels and cuFFT calls. We have extended the core by CUDA kernels for domain overlap extraction/injection.

The communication framework is responsible for domain decomposition and overlap exchange between neighboring sub-domains. The decomposition is planed ahead and each process loads its part of the domain independently using distributed I/O (results are collected back in the similar way). Overlap exchanges are realized using asynchronous OpenMPI communications and are partially overlapped. Communication overlapping is realized in a way where we extract halos of multiple matrices and start all communications. Here, we only need to wait for all halo zones of the first matrix before computing its gradient.

C. Performance and Scaling

The implementation of the local decomposition was tested on the Emerald cluster which consists of nodes equipped with three or eight NVIDIA C2070 (GF100 - Fermi). Nodes are interconnected by a QDR Infiniband in the half-duplex mode. Both GPUs themselves and the interconnection are therefore rather outdated and we should be (and in fact we are) able to achieve much better results on clusters like Anselm (with very limited number of GPUs). Our experiments were conducted running 100 steps of the simulation (which should be enough to hide any warm-up latencies) with overlaps of 16 grid points. Figure 5 shows the scaling and overhead of our solution on a range of configurations with one to 128 GPUs and domains sizes of $256 \times 256 \times 256$ to $1024 \times 1024 \times 2048$ (2^{24} to 2^{31} points) using decomposition in all three dimensions.

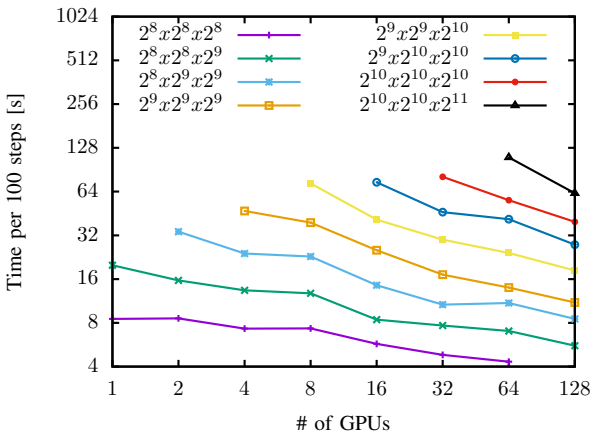


Fig. 5: Scaling on Emerald Cluster

We are able to achieve reasonable strong scaling. The overall efficiency of proposed code increases with the size of the simulation domain. This behavior is expected and can be explained as a result of the decrease in both computational and communication overhead (which grows only with surface area of the sub-domains).

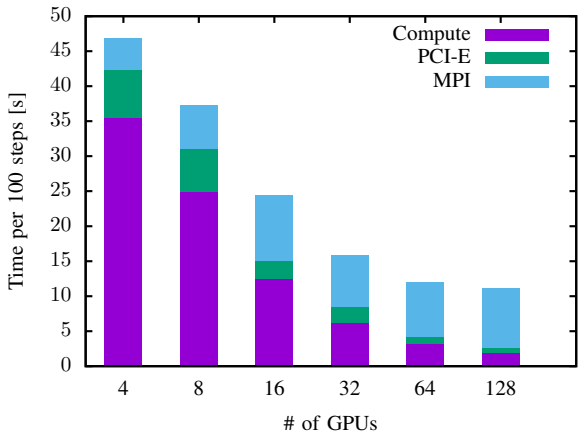


Fig. 6: Overhead on Emerald Cluster

Figure 6 shows the breakdown of the simulation run time for domain size of $512 \times 512 \times 512$ using a 3D decomposition and 16 grid points of overlap. It can be seen that the MPI communication overhead increases until the 3D decomposition is reached ($2 \times 2 \times 4$ GPUs). We can see that the pure computation time doesn't decrease exactly linearly, due to the local domain extensions overhead. The extension overhead reaches exactly 50% at 128 GPUs, which is one of the limits of algorithms strong scaling. Other more prominent limit of the strong scaling is the communication overhead which reaches over 50% of total simulation time at about 16-32 GPUs. Overall the overhead is still significantly lower that that of the global decomposition of FFT [5].

IV. Future Research

The proposed simulation algorithm is apparently a member of the large class of algorithms whose performance tend to be memory and inter-connect bandwidth (and/or latency) bound. Such behavior may not be intrinsic to the algorithm itself, but rather result of its design and implementation. It may be the case that the algorithm behaves like memory/communication bound one due to its inefficient use of the clusters memory hierarchy or poor mapping to the distributed architecture.

In the future we therefore want to focus our effort on the design of efficient decomposition methods (such as proposed local decomposition), communication, memory access patterns and data locality optimizations. We plan to tackle these areas in multiple phases, first we are going to analyze some other simulation algorithms and try to find their common patterns. Secondly we will try to find common solutions of these problems and finally propose ways to automatize these solutions.

At this point we are leaning towards functional representation of the algorithms, which should give us some advantages for their automatic analysis. There is being done some progress at mapping functional paradigm onto massively parallel hardware [6] and we would like to extend such (or similar) approach to distributed architectures with multiple memory address spaces.

There is also a new development in the area of memory access pattern analysis (for non-uniform memory architectures), which significantly reduces the effort necessary to use multiple GPUs or port GPU kernels to new architectures. In the recent work [7] these qualities are achieved by introducing new representation of the GPU kernels. It should be possible to take similar approach and create such representation of the algorithm from its original functional form.

V. Conclusion

This article outlines some of the current problems in the high performance computing on accelerated clusters with non-uniform memory and multiple memory address spaces. We discussed our work in this area on the pseudo-spectral simulation where we achieved significant improvements in the performance. We also briefly outlined possible directions of our future research in the design of efficient distributed algorithms.

Acknowledgment

This work was supported by the FIT-14-2297 Architectures of Parallel and Embedded Computer Systems project.

References

[1] B. E. Treeby and B. T. Cox, "k-Wave: MATLAB toolbox for the simulation and reconstruction of photoacoustic wave fields," *Journal of biomedical optics*, vol. 15, no. 2, pp. 021 314–021 314, 2010.

[2] M. Israeli, L. Vozovoi, and A. Averbuch, "Spectral multidomain technique with local Fourier basis," *Journal of Scientific Computing*, vol. 8, no. 2, pp. 135–149, 1993.

[3] J. Jaros, A. P. Rendell, and B. E. Treeby, "Full-wave nonlinear ultrasound simulation on distributed clusters with applications in high-intensity focused ultrasound," *International Journal of High Performance Computing Applications*, p. 1094342015581024, 2015.

[4] J. Nieplocha, R. J. Harrison, and R. J. Littlefield, "Global arrays: A nonuniform memory access programming model for high-performance computers," *The Journal of Supercomputing*, vol. 10, no. 2, pp. 169–189, 1996. [Online]. Available: http://link.springer.com/article/10.1007/BF00130708;http://www.emsl.pnl.gov/docs/global/papers/tjs.pdf

[5] A. Gholami, J. Hill, D. Malhotra, and G. Biros, "AccFFT: A library for distributed-memory FFT on CPU and GPU architectures." *CoRR*, vol. abs/1506.07933, 2015. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1506.html#GholamiHMB15

[6] M. M. Chakravarty, G. Keller, S. Lee, T. L. McDonell, and V. Grover, "Accelerating Haskell array codes with multicore GPUs," in *Proceedings of the sixth workshop on Declarative aspects of multicore programming*. ACM, 2011, pp. 3–14. [Online]. Available: http://dl.acm.org/citation.cfm?id=1926358;http://129.94.242.51/~keller/Papers/acc-cuda.pdf

[7] T. Ben-Nun, E. Levy, A. Barak, and E. Rubin, "Memory access patterns: the missing piece of the multi-GPU puzzle." in *SC*, J. Kern and J. S. Vetter, Eds. ACM, 2015, pp. 19:1–19:12. [Online]. Available: http://dblp.uni-trier.de/db/conf/sc/sc2015.html#Ben-NunLBR15

[8] "The AMD Fast Forward Project," Feb. 2014. [Online]. Available: https://asc.llnl.gov/fastforward/AMD-FF.pdf

[9] "Summit and Sierra Supercomputers: An Inside Look at the U.S. Department of Energy's New Pre-Exascale Systems," Nov. 2014. [Online]. Available: http://www.teratec.eu/actu/calcul/Nvidia_Coral_White_Paper_Final_3_1.pdf

[10] K. Okita, K. Ono, S. Takagi, and Y. Matsumoto, "Development of high intensity focused ultrasound simulator for large-scale computing," *International Journal for Numerical Methods in Fluids*, vol. 65, no. 1-3, pp. 43–66, 2011. [Online]. Available: http://dx.doi.org/10.1002/fld.2470

[11] Jan S. Hesthaven, Sigal Gottlieb, David Gottlieb, *Spectral methods for time-dependent problems*. Cambridge : Cambridge University Press, 2007.

Hardvérová platforma pre systémy reálneho času

Lukáš Kohútka
1. ročník, denné štúdium
Školiteľ: Viera Stopjaková

Fakulta elektrotechniky a informatiky, Slovenská technická univerzita v Bratislave
Ilkovičova 3, 812 19 Bratislava
lukas.kohutka@stuba.sk

***Abstrakt*—Tento príspevok prezentuje doterajšie výsledky výskumu a koncepčné návrhy v rámci oblasti systémov reálneho času za účelom zlepšenia ich výkonu, deterministickosti, veľkosti a robustnosti. Ako riešenie je navrhovaná nová platforma pre systémy reálneho času, ktorá pozostáva z architektúry na systémovej úrovni a jednotlivých komponentov na úrovni medziregistrových prenosov. Základnými princípmi použitými v rámci tohto výskumu sú: hardvérová akcelerácia, paralelizácia na úrovni počtu jadier procesora, prúdové spracovanie inštrukcií, modulárnosť systému a dynamická rekonfigurácia realizovaná hradlovými poliami.**

Kľúčové slová—systém reálneho času; hardvérová akcelerácia; koprocesor; architektúra; determizmus; FPGA; rekonfigurácia

I. Úvod a Motivácia

S rastúcou hustotou integrácie integrovaných obvodov (IO) a systémov na čipe (SoC) sa postupne rozširujú možnosti, ako využiť pribúdajúce množstvo tranzistorov pre rozličné aplikácie, vrátane aplikácií určených pre systémy reálneho času. No napriek zvyšujúcej sa hustote integrácie, rast pracovnej frekvencie číslicových IO sa za posledných pár rokov výrazne spomalil, až takmer zastavil. Aby sa aj naďalej stúpal výpočtový výkon počítačových systémov, začal sa kompenzovať tento nepriaznivý jav používaním viacjadrových procesorov, ktoré vykonávajú viacero inštancií programov paralelne. Prístup použitia viacjadrových procesorov je vhodným riešením najmä pre bežné systémy a dobre paralelizovateľné aplikácie, pretože s rastúcim počtom jadier sa znižuje priemerný čas potrebný na dokončenie paralelizovateľných častí programu. Avšak pre systémy reálneho času predstavuje tento prístup menší prínos [1].

Alternatívnym prístupom ako využiť zvyšujúce sa množstvo tranzistorov na čipe je hardvérová akcelerácia. Tá predstavuje hardvérovú realizáciu rozličných výpočtových algoritmov, ktoré sa realizujú zvyčajne softvérovo. Hardvérová realizácia môže znížiť časovú asymptotickú zložitosť vybraného algoritmu. Napríklad ak najlepšia známa softvérová realizácia daného algoritmu má lineárnu časovú zložitosť, tak hardvérová realizácia môže dosiahnuť až konštantnú časovú zložitosť. Konštantná časová zložitosť znamená, že bez ohľadu na množstvo dát v systéme, daná operácia alebo algoritmus trvá vždy rovnako dlhý, čiže konštantný časový úsek. Práve konštantná časová zložitosť je veľmi dôležitá pre systémy reálneho času, pretože prispieva k tomu, aby bol celý systém reálneho času viac deterministický. Deterministickosť je pre systémy reálneho času mimoriadne dôležitá vlastnosť [1].

Charakteristickou črtou systémov reálneho času je to, že obsahujú aspoň jednu úlohu, ktorú možno označiť za úlohu reálneho času. Úloha reálneho času je inštancia programu alebo jej časť (t.j. proces alebo vlákno), pričom takáto úloha musí byť dokončená najneskôr do stanoveného času. V opačnom prípade môže byť výsledok tejto úlohy považovaný za nepresný alebo dokonca úplne nepoužiteľný. Úlohy reálneho času je potrebné naplánovať v systéme takým spôsobom, aby bolo zabezpečené a zaručené ich splnenie v správnom čase. Čím je celý systém deterministickejší, tým ľahšie je možné tento cieľ dosiahnuť. Vysoká miera deterministickosti navyše umožňuje vytvárať komplexnejšie systémy reálneho času s väčším množstvom úloh [1].

Systémy reálneho času sa využívajú v mnohých oblastiach priemyslu ako napríklad letecký, vesmírny, automobilový, železničný, výrobný, energetický, chemický a ďalšie. Celkovo sú teda systémy reálneho času dosť rozšírené a preto má zmysel sa zaoberať tým, ako ich vylepšiť. V mnohých prípadoch sa využívaju systémy reálneho času na kritické aplikácie, kedy je nutné sa zaoberať aj spoľahlivosťou systému a jeho kritických úloh. Ak sa totiž nejaká kritická úloha nevykoná včas, môže to mať fatálne následky. Hardvérová akcelerácia prináša pre systémy reálneho času nielen zrýchlenie systému, ale zároveň umožňuje zvýšiť mieru deterministickosti, spoľahlivosti a v neposlednom rade aj robustnosti celého systému, čiže možnosť disponovať väčším množstvom úloh a s tým súviacej širšej funkcionality systému.

II. Cieľ Práce

Cieľom dizertačnej práce, o ktorej je tento príspevok, je navrhnúť a vytvoriť novú hardvérovu platformu pre systémy reálneho času. Takáto platforma by mala byť univerzálna, efektívna, modulárna, konfigurovateľná a prispôsobiteľná na základe potrieb konkrétnych systémov reálneho času. Celá platforma bude realizovaná a otestovaná na jednom FPGA čipe.

Práca sa bude zaoberať nielen architektúrou platformy na systémovej úrovni, ale aj jednotlivými komponentami platformy na úrovni medziregistrových prenosov. To umožňuje optimalizáciu platformy na viacerých úrovniach abstrakcie – ako komponentov, tak aj ich vzájomného prepojenia. Najdôležitejším parametrom optimalizácie je miera deterministickosti, ktorá je v ideálnom prípade

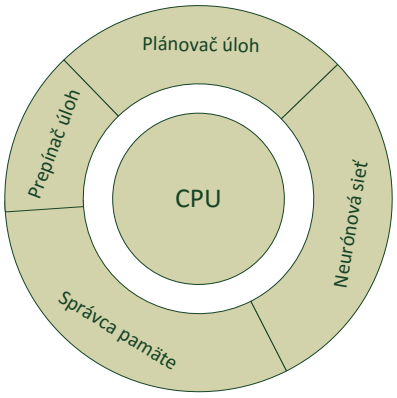
dosiahnutá konštantnou časovou zložitosťou operácií, ktoré má platforma poskytovať. Ďalšími parametrami optimalizácie sú štandardné parametre, na ktoré sa prihliada pri vývoji číslicových obvodov. Medzi tie patrí napríklad pracovná frekvencia systému, počet hodinových cyklov potrebný na vykonanie operácie, plocha čipu (pre FPGA množstvo spotrebovaných logických elementov, registrov a pamäte) a spotreba energie.

III. Navrhnutá Platforma

Nová architektúra platformy pre systémy reálneho času je založená na kombinovaní jedného alebo viacerých jadier procesora s hardvérovými akcelerátormi vo forme koprocesorov. Jediný povinný komponent tejto architektúry je CPU, ktorý je implicitne jednojadrový. Ostatné komponenty sú nepovinné a konfiguráciou na úrovni architektúry sa určuje najmä to, ktoré z nepovinných komponentov má systém obsahovať. Toto závisí od konkrétnej aplikácie, na ktorú sa má systém použiť. Platforma celkovo disponuje týmito komponentmi:

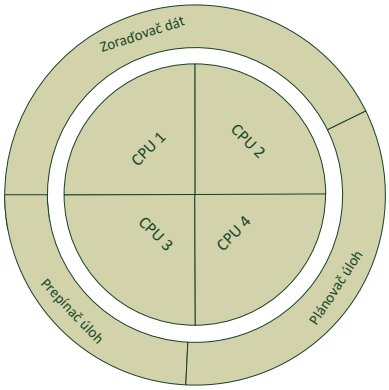
- CPU
- Prepínač úloh
- Plánovač úloh
- Správca dynamickej pamäte
- Zoraďovač aplikačných dát
- Neurónová sieť

Na obrázku č. 1 je zobrazený príklad možnej konfigurácie architektúry HW platformy, v ktorej CPU má k dispozícii 4 koprocesory: plánovač úloh, prepínač úloh, správca pamäte a neurónovú sieť. Rozdielne veľkosti jednotlivých komponentov vyjadrujú relatívnu spotrebu plochy na čipe v porovnaní s ostatnými komponentmi. V tomto prípade je CPU jednojadrové.



Obrázok č. 1 – Príklad konfigurácie platformy 1

Na obrázku č. 2 je zobrazený druhý príklad konfigurácie. V tomto prípade je použitá nielen iná kombinácia komponentov, ale aj iná konfigurácia samotných komponentov, čo vplýva okrem iného aj na spotrebu plochy na čipe a teda na zmenený pomer veľkostí jednotlivých komponentov. Keďže tento príklad obsahuje 4-jadrový procesor, zmenil sa tiež pomer spotreby plochy medzi procesorom a koprocesormi.



Obrázok č. 2 – Príklad konfigurácie platrofmy 2

Takáto architekúra je navyše veľmi vhodná aj pre dynamickú rekonfiguráciu. Tá umožňuje meniť konfiguráciu systému aj počas jeho činnosti. Prvé jadro CPU je označené za hlavné jadro, ktoré je realizované v statickej časti logiky. Ostatné jadra CPU a jednotlivé koprocesory sú realizované dynamickou časťou logiky, ktorá tým pádom tvorí výraznú väčšinu celej logiky. Vďaka tomu je možné, aby si jednotlivé dynamické komponenty navzájom prepožičiavali logické elementy a registre na základe ich potrieb, ktoré závisia od aktuálnej konfigurácie ako celej architektúry, tak aj jednotlivých komponentov. Dôsledkom dynamickej rekonfigurácie pre túto architektúru je výrazná úspora spotreby plochy na čipe.

IV. Komponenty Platformy

A. CPU

V rámci dizertačnej práce bude navrhnutý procesor s vlastnou inštrukčnou sadou tak, aby bolo možné čo najefektívnejšie a najjednoduchšie prepojiť takýto procesor s ostatnými komponentmi. Zároveň bude kladený dôraz na deterministickosť riešenia, napríklad pri aplikovaní techniky prúdoveho spracovania inštrukcií známeho ako pipelining, aby bol procesor čo najvhodnejší pre systémy reálneho času.

Konfigurovateľnosť CPU spočíva najmä v určení počtu jadier CPU, ale môže ísť napríklad aj o počet stupňov v prúdovom spracovaní inštrukcií, množstvo registrov a bitová šírka dátových typov.

B. Prepínač úloh

Prepínanie úloh je možné realizovať hardvérovo s konštantnou časovou zložitosťou vďaka existujúcej technike zvanej „tieňové registre“. Nevýhodou tejto techniky je zvýšená plocha na čipe, najmä kvôli registrom.

Konfigurovateľnosť prepínača úloh spočíva najmä v stanovení maximálneho množstva úloh v systéme a v špecifikácií množstva a bitovej šírky registrov procesora. Parametre tohto komponentu teda závisia od parametrov CPU a plánovača úloh.

C. Plánovač úloh

Doterajší vývoj v rámci práce bol zameraný práve na preemptívny plánovač úloh s konštantnou časovou zložitosťou nezávislou od množstva úloh v systéme. Podarilo sa vyvinúť viacero rôznych verzií plánovača úloh, ako pre jednojadrový procesor, tak aj pre viacjadrový. Takýto plánovač je založený na algoritmoch EDF (Earliest Deadline First) a FIFO, rozšírených o možnosť odstrániť ľubovoľnú úlohu na základe jej identifikátora. Plánovanie úloh sa teda v tomto prípade realizuje nielen na základe doby odozvy, do ktorej je potrebné dokončiť úlohy reálneho času, ale aj na základe priority. Vďaka tomu je možné pomerne jednoducho kombinovať v tom istom systéme úlohy reálneho času s bežnými úlohami, ktoré nemajú stanovenú dobu odozvy. Celkovo možno skonštatovať, že tieto nové verzie plánovača úloh sú flexibilnejšie a časovo efektívnejšie v porovnaní s doterajšími existujúcimi riešeniami [2-6].

Pre dvojjadrové procesory boli taktiež navrhnuté dve techniky riešenia konfliktov, pričom pojem konflikt je vnímaný ako situácia, kedy viacero procesorových jadier chce použiť koprocesor v tom istom čase. Prvá metóda nazývaná *semaforová technika*, vyberá v prípade konfliktu víťazné jadro procesora, ktoré môže ihneď použiť daný koprocesor, zatiaľ čo ostatné jadra procesora sú zatiaľ pozastavené. Táto technika je pomerne jednoduchá a efektívna z hľadiska plochy na čipe. Druhá technika nazvaná ako *simultánne spracovanie* je založená na vnútornom rozšírení logiky v koprocesore takým spôsobom, aby bol koprocesor schopný prijímať a vykonávať inštrukcie od viacerých jadier procesora súčasne. Tým sa zamedzí vzniku konfliktov a tým pádom ani nedochádza k pozastavovaniu alebo oneskoreniu v procesore. Nevýhodou tohto prístupu je iba zložitejší návrh koprocesora a s tým spojená väčšia plocha čipu potrebná na realizáciu.

V prípade, že procesor obsahuje viac ako 2 jadrá, je možný ešte tretí prístup, ktorý kombinuje predchádzajúce dva prístupy. Tento prístup prináša určitý kompromis medzi výhodami a nevýhodami predchádzajúcich prístupov.

Medzi konfigurovateľné parametre plánovača úloh patria:

- Maximálny počet úloh
- Pracovná frekvencia
- Presnosť a rozsah času
- Počet priorít
- Spôsob počítania doby odozvy (časová pečiatka alebo zostávajúci čas)
- Počet stupňov prúdového spracovania v plánovači
- Rozhodovacie algoritmy (EDF alebo Least Slack Time, FIFO alebo Round Robin)

Navrhnuté verzie tohto koprocesora sú už zároveň implementované, verifikované metodológiou Universal Verification Methodology zosyntetizované pre FPGA Altera Cyclone II, otestované funkčným BIST testom, a v neposlednom rade publikované vo viacerých príspevkoch na rôznych konferenciách [7-12].

V budúcnosti by bolo možné plánovač úloh ešte ďalej rozšíriť o funkcionalitu synchronizácie úloh a o podporu periodických, prípadne aj sporadických úloh.

Správca dynamickej pamäte

Pod pojmom dynamická správa pamäte sa myslí alokácia a dealokácia blokov pamäte rozličnej možnej veľkosti, podobne ako je tomu v prípade funkcií jazyka C: *malloc* a *free*.

Aj keď už existuje aj realizácia dynamickej správy pamäte pre systémy reálneho času, táto realizácia je žiaľ iba softvérová a pomerne heuristická, a teda málo univerzálna. Naviac štatistické výsledky ukazujú, že využitie pamäte je len na úrovni 75% a menej, čo je teda pomerne neefektívne.

Riešením by mohol byť koprocesor, ktorý by realizoval existujúci algoritmus známy ako *worst fit*. Tento algoritmus bol zvolený z toho dôvodu, že jediná zložitejšia operácia, ktorú obsahuje, je zoraďovanie blokov pamäte podľa jej veľkosti. Keďže operáciu zoraďovania sa podarilo implementovať v konštantnom čase už pre plánovač úloh, je zrejmé, že rovnako aj tento algoritmus je možné realizovať tak, aby mal konštantnú časovú zložitosť, a teda aby bol maximálne deterministický. Nevýhodou tohto algoritmu je potenciálna náchylnosť na externú fragmentáciu, ktorá znižuje využitie pamäte. Na druhej strane je všeobecne známe, že v prípadoch, keď aplikácia používa podobne veľké bloky pamäte, dosahuje tento algoritmus najlepšiu mieru jej využitia.

Z hľadiska konfigurovateľnosti by mal správca pamäte používať tieto parametre:

- Veľkosť pamäte vyhradenej pre dynamickú správu
- Granularita – veľkosť najmenšieho bloku pamäte a jeho možných násobkov
- Maximálny počet voľných blokov pamäte

D. Zoraďovač aplikačných dát

Keďže predchádzajúce dva koprocesory už disponovali hardvérovou realizáciou zoraďovania určitých dát (zoraďovanie úloh v prípade plánovača úloh a zoraďovanie voľných blokov pamäte v prípade správcu dynamickej pamäte), prirodzene vznikol nápad na použitie obdobnej architektúry zoraďovania pre aplikačné dáta, ktoré používajú jednotlivé úlohy. Obdobným príkladom sú softvérové zoraďovacie dátové štruktúry a algoritmy ako *quicksort, heapsort* a vyvážené binárne stromy.

Avšak nestačí iba prevziať zoraďovanie z predchádzajúcich dvoch koprocesorov. V tomto prípade treba ešte brať do úvahy fakt, že počet a veľkosť požadovaných zoraďovacích štruktúr sa môže rýchlo meniť v čase, v závislosti od aplikácie a jednotlivých práve existujúcich úloh. Preto v tomto prípade nie je možné sa spoliehať iba na dynamickú rekonfiguráciu, ktorou disponuje FPGA, keďže takáto rekonfigurácia je príliš pomalá. V prípade, že nejaká úloha potrebuje vytvoriť novú zoraďovaciu štruktúru pre svoje dáta, by čas strávený dynamickou rekonfiguráciou mohol výrazne zredukovať čas ušetrený hardvérovou akceleráciou. Z tohto dôvodu bola navrhnutá nová architektúra zostavená z určitého množstva menších zoraďovacích štruktúr, ktoré je možné spájať do väčších štruktúr zreťazením. Vďaka tomu je takýto koprocesor prispôsobiteľný potrebám jednotlivých úloh.

Konfigurovateľné parametre zoraďovača aplikačných úloh:

- Počet zoraďovacích blokov
- Veľkosť zoraďovacieho bloku
- Bitová šírka kľúča
- Bitová šírka hodnoty
- Smer zoradenia (minimum alebo maximum)
- Počet stupňov prúdového spracovania pre každý blok

E. Neurónová sieť

Niektoré systémy reálneho času sú zároveň vhodným kandidátom pre inteligentné spracovanie informácií a umelú inteligenciu. Príkladom takejto aplikácie je počítačové videnie, konkrétne automatické vyhodnocovanie obrazu, automatické brzdenie a automatický parkovací systém v moderných automobiloch.

V prípade potreby inteligentného spracovania informácií v systémoch reálneho času vo všeobecnosti dominuje realizácia pomocou umelých neurónových sietí, pretože v porovnaní s ostatnými metódami je po natrénovaní takejto siete rýchlosť výpočtu vysoká. V prípade neurňových sietí existuje pomerne intenzívny výskum a taktiež už existuje pomerne veľa riešení aj na úrovni hardvéru. Ich nevýhodou je však pomerne vysoká spotreba plochy na čipe a relatívne zložitý proces trénovania siete.

V rámci tejto oblasti vznikol nápad ohľadne novej architektúry doprednej prípadne aj rekurentnej neurónovej siete, ktorá by bola založená na princípe sériovej komunikácie medzi neurónmi a počítadlami, ktoré by vyjadrovali vnútorné stavy neurónov. Týmto spôsobom by sa fungovanie výslednej neurónovej siete nielen viac podobalo na fyziológiu biologických neurónových sietí, ale taktiež by sa tým mohla výrazne zredukovať plocha čipu HW realizácie, pretože operácia váhovania v neuróne by nebola realizovaná násobičkami, ale iba logickým násobením. Zatiaľ sa jedná iba o koncepčný návrh na vysokej úrovni abstrakcie a preto bude predmetom ďalšieho výskumu v budúcnosti.

Medzi konfiguračné parametre bude patriť minimálne:

- Počet vrstiev neurónovej siete
- Počet neurónov v každej vrstve
- Topológia siete
- Rozsah a presnosť hodnôt v neuróne
- Zoznam spätných väzieb (pre rekurentú sieť)
- Typ neurónovej siete
- Typ trénovania
- Počiatočné váhy

V. Záver

Bola navrhnutá nová architektúra ako platforma pre systémy reálneho času, ktorá bola opísaná na vyššej úrovni abstrakcie – systémovej úrovni. Základnou vlastnosťou a výhodou tejto architektúry je univerzálnosť, modulárnosť, konfigurovateľnosť a prispôsobiteľnosť na základe potrieb konkrétnych aplikácií. Navyše je takáto architekrúra veľmi vhodná pre technológiu FPGA, pretože veľká časť platformy môže patriť do dynamickej časti logiky, vďaka čomu je možné dosiahnuť relatívne veľkú úsporu plochy na čipe.

V rámci tohto príspevku boli opísané aj jednotlivé komponenty navrhovanej architektúry, medzi ktoré patrí: CPU, plánovač úloh, prepínač úloh, správca pamäte, zoraďovač dát a neurónová sieť. Niektoré z týchto komponentov boli taktiež stručne opísané s použitím nových prístupov ich realizácie. Na nižšej úrovni abstrakcie, na úrovni medziregistrových prenosov, bol navrhnutý plánovač úloh ako jeden z komponentov tvoriacich navrhovanú platformu. Tento plánovač úloh úloh bol navrhnutý vo viacerých verziách, ktoré boli odsimulované, implementované a otestované, a dosiahnuté výsledky boli publikované v rámci viacerých konferencií.

V úvode príspevku bola opísaná motivácia a možné prínosy v rámci tejto témy. Najväčším prínosom je zvýšenie výkonnosti výpočtov a miery determinizmu v rámci systémov reálneho času, čo následne pozitívne vplýva na možnosť realizácie väčších a komplexnejších systémov reálneho času.

Poďakovanie

Táto práca bola podporená projektom APVV-15-0245.

Referencie

[1] G.C. Buttazzo, “Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications,” 2011.

[2] Y. Tang, and N.W. Bergmann, “A Hardware Scheduler Based on Task Queues for FPGA-Based Embedded Real-Time Systems,” IEEE Transactions on Computers, 2015.

[3] J. Starner, J. Adomat, J. Furunas, and L. Lindh, “Real-Time Scheduling Co-Processor in Hardware for Single and Multiprocessor Systems,” Proceedings of the EUROMICRO Conference, 1996.

[4] C. Ferreira, and A.S.R. Oliveira, “Hardware Co-Processor for the OReK Real-Time Executive,” 2010.

[5] S.E. Ong, and S.C. Lee, “SEOS: Hardware Implementation of Real-Time Operating System for Adaptability,” Computing and Networking (CANDAR), 2013 First International Symposium, 2013.

[6] K. Kim, D. Kim, and Ch. Park, “Real-Time Scheduling in Heterogeneous Dual-core Architectures,” Proceedings of the 12th International Conference on Parallel and Distributed Systems, 2006

[7] L. Kohutka, “Hardware Task Scheduling in Real-Time Systems,” Proceedings of IIT.SRC, 2015.

[8] L. Kohutka, M. Vojtko, and T. Krajcovic, “Hardware Accelerated Scheduling in Real-Time Systems,” Engineering of Computer Based Systems Eastern European Regional Conference, 2015.

[9] L. Kohutka, V. Stopjakova, “Hardware-Accelerated Task Scheduling in Real-Time Systems: Deadline Based Coprocessor for Dual-Core CPUs,” International Symposium on Design and Diagnostics of Electronic Circuits and Systems, 2016.

[10] L. Kohutka, V. Stopjakova, “Hardware Accelerated Task Scheduling in Real-Time Systems,” Conference on Advances in Electronic and Photonic Technologies, 2016.

[11] L. Kohutka, V. Stopjakova, “Task Scheduler for Dual-Core Real-Time Systems,” International Conference on Mixed Design of Integrated Circuits and Systems, 2016.

[12] L. Kohutka, V. Stopjakova, “Improved Task Scheduler for Dual-Core Real-Time Systems,” Euromicro Conference on Digital System Design, 2016.

Polymorfní obvody na bázi ambipolárních tranzistorů

Jan Nevoral
1. ročník, prezenční studium
Školitel: Richard Růžička

Fakulta informačních technologií, Vysoké učení technické v Brně
Božetěchova 2, 612 66 Brno, Česká Republika
inevoral@fit.vutbr.cz

***Abstrakt*—Cílem této práce je představit novou metodu pro evoluční návrh polymorfních obvodů na úrovni tranzistorů. K dosažení rychlého a přitom dostatečně přesného ohodnocení obvodu je využit diskrétní simulátor. Navržený přístup byl testován na množině polymorfních hradel řízených přehozením napájecích přívodů. Ukázalo se, že je metoda schopná navrhovat funkční řešení. Bylo vytvořeno několik polymorfních obvodů na bázi ambipolárních tranzistorů, které jsou sestaveny z menšího počtu tranzistorů než stejná dosud publikovaná hradla. Díky navržené metodě byla také objevena nová třída polymorfních hradel – hradla s MOSFET tranzistory přepínaná změnou polarity napájení. Ty mají pravděpodobně nejlepší parametry z dosud známých polymorfních hradel postavených z konvenčních tranzistorů.**

Klíčová slova—polymorfní elektronika, ambipolární tranzistor, číslicový obvod, logické hradlo, evoluční návrh, CGP.

I. Úvod

V současné době je většina výpočetních strojů založena na prvcích na bázi anorganických polovodičových materiálů, jako je křemík. Nejčastěji se jedná o tranzistory použité v roli spínačů, z nichž jsou sestavena logická hradla, která realizují základní booleovské funkce. Z hradel jsou nakonec pomocí syntézních prostředků skládány složitější obvody, které již mohou realizovat žádaný algoritmus. Pro návrh takovéto elektroniky, označované jako konvenční, jsou již automatizované postupy známé.

V dnešní době existují však již i jiná, alternativní řešení, která mohou v použitém systému přinášet jisté výhody. Za tyto nekonvenční technologie lze považovat například obvody založené na organických polovodičích, polovodičových prvcích na bázi grafenu nebo křemíkových nanodrátků, které vykazují různé chování v závislosti na stavu okolního prostředí. Toho je možné využít v polymorfní elektronice, která v závislosti na stavu okolního prostředí realizuje různé funkce. Smyslem použití polymorfní elektroniky může být úspora prostředků použitých při realizaci stejné funkčnosti konvenční elektronikou nebo jiná přidaná hodnota oproti klasickému řešení.

Návrh polymorfních hradel není pro návrháře jednoduchá úloha především proto, že je při návrhu nutné myslet na všechny funkce, které má obvod schopný vykonávat. Proto je také většina existujících polymorfních hradel výsledkem evolučního návrhu. Do této chvíle však nebyla publikována žádná evoluční metoda pro návrh polymorfních obvodů na bázi ambipolárních tranzistorů, které mohou být základním stavebním prvkem polymorfních obvodů.

Tento článek shrnuje základní principy polymorfní elektroniky, vlastnosti ambipolárních tranzistorů a představuje implementovaný systém pro evoluční návrh polymorfních hradel, pomocí něhož bylo vytvořeno několik hradel s menším počtem tranzistorů, než využívala stejná dosud publikovaná hradla.

II. Polymorfní elektronika

Polymorfní elektronika [1] je oblast elektroniky, která se zabývá číslicovými elektronickými obvody, jež dokážou plnit více než jednu funkci, zatímco jejich zapojení zůstává stále stejné. Aktuálně prováděná funkce závisí na stavu okolního prostředí. Všechny funkce jsou do obvodu zabudovány úmyslně již při jeho návrhu, nejedná se tedy například o poruchový stav sytému způsobený překročením provozních parametrů obvodu.

Polymorfní obvod je prostorově (tedy co do počtu tranzistorů nebo plochy čipu) skromnější než obvod navržený konvenčními postupy, jenž se skládá z několika monofunkčních obvodů, které se podle stavu prostředí přepínají. Detekce stavu prostředí je přirozenou součástí obvodu, na úrovni tranzistorů je provázána s jeho dalšími funkcemi. Přepínání mezi funkcemi se tedy děje u polymorfního obvodu přirozeně a okamžitě, není třeba čekat například na dokončení rekonfigurace.

Stav okolního prostředí lze často popsat hodnotou nějaké fyzikální veličiny, pro konkrétní stav prostředí lze jednoznačně určit, jakou funkci bude obvod plnit. Mezi stavy okolního prostředí můžeme zařadit jeho teplotu, velikost napájecího napětí systému, jeho polaritu nebo například napětí na speciálním vodiči.

Polymorfní obvod se stejně jako klasický obvod typicky skládá z menších částí, polymorfních hradel. Několik takových hradel již bylo v minulosti publikováno [1]:

- **Hradla řízená teplotou:** and/or, nand/nor
- **Hradla řízená napájecím napětím:** and/or, nand/nor
- **Hradla řízená speciálním signálem:** and/or, nand/nor, nand/xor, and/or/xor

Velkým problémem polymorfních hradel řízených velikostí napájecího napětí je jejich spotřeba, která je při určitých kombinacích vstupních hodnot příliš veliká. Stejně tak jsou výstupní úrovně některých hradel příliš vzdáleny těm ideálním.

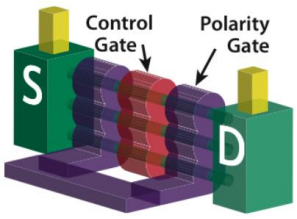
Polymorfní hradla se dvěma funkcemi, která jsou řízena napětím na speciálním signálu, většinou využívají pro tyto funkce napájecí napětí obvodu a napětí 0 V. Lze je tedy považovat za kombinační obvody s jedním vstupem navíc a navrhovat je klasickými metodami.

V rámci disertační práce se zabývám především návrhem polymorfních obvodů řízených polaritou napájecího napětí (přepínáním napájecích přívodů), které jsou sestaveny z MOSFET nebo čtyřvývodových ambipolárních tranzistorů. Tyto obvody mají nízkou spotřebu, neboť jsou na rozdíl od některých z výše zmíněných hradel navrženy tak, aby nikdy nedocházelo ke zkratu mezi napájecím napětím a zemí. Stejně tak poskytují navržené obvody kvalitní výstupní úrovně. Návrhem polymorfních hradel se zabývá na FIT VUT i Ing. Radek Tesař. Ten však využívá ambipolárního chování tranzistorů se třemi elektrodami [2].

III. Ambipolární tranzistory

Vývoj tranzistorů na bázi křemíku postupně naráží na technologické limity. V poslední letech se však začínají objevovat polovodičové materiály, které by mohly křemík nahradit. Některé z těchto materiálů vykazují ambipolární chování.

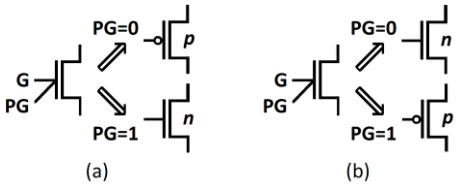
Ambipolární tranzistory [3] jsou FET tranzistory, které jsou nejčastěji konstruovány se čtyřmi vývody. K těm klasickým (*source, drain, control gate*) přibyl čtvrtý, označovaný jako *polarity gate*. Čtyřvývodové ambipolární tranzistory již byly popsány v několika technologiích: křemíkových nanodrátcích (SiNWs), grafenu a karbonových nanotrubičkách (CNTFET). Tranzistor vyrobený z takovéhoto materiálu se pak může za určitých podmínek chovat jako tranzistor typu P, za jiných jako tranzistor typu N. Na Obrázku 1 je zobrazen tranzistor založený na křemíkových nanodrátcích, který takovéto chování vykazuje.



Obrázek 1: Struktura ambipolárního SiNW tranzistoru [3].

Chování ambipolárních tranzistorů se liší podle použitých technologií [3] [4]. Mohou být rozděleny do dvou kategorií:

- *Typ 1*: Při logické jedničce na *polarity gate* vykazují chování tranzistorů typu n-MOS, při logické nule tranzistorů typu p-MOS – viz Obrázek 2a.
- *Typ 2*: Při logické jedničce na *polarity gate* se chovají jako tranzistory typu p-MOS, při logické nule jako tranzistory typu n-MOS – viz Obrázek 2b.



Obrázek 2: Chování ambipolárního tranzistoru typu 2 [4].

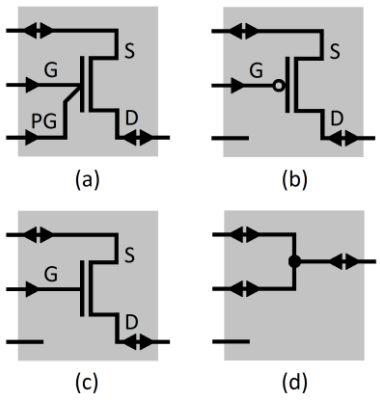
IV. Návrh polymorfních hradel

Návrh polymorfních číslicových obvodů [1] je definován jako hledání grafu G, který reprezentuje zapojení obvodu tak, aby obvod dokázal realizovat všechny zamýšlené funkce. Při změně funkce obvodu se tedy musí změnit pouze funkce jeho jednotlivých komponent, zapojení (graf G) zůstává stále stejné.

Návrh číslicové polymorfní elektroniky probíhá v současnosti pouze na úrovni hradel. Samotná polymorfní hradla se pak navrhují na úrovni tranzistorů. K návrhu polymorfních hradel se používají nejčastěji dva způsoby:

- **Ad hoc** – Menší a jednodušší polymorfní obvody je někdy možné syntetizovat bez použití nějakých technik, pouze s využitím zkušeností návrháře.
- **Kartézské genetické programování** – Kandidátní řešení jsou reprezentována pomocí orientovaných grafů. Tyto grafy (s pevně daným počtem uzlů) jsou zakódovány do stejně dlouhých posloupností čísel a hledání hradel probíhá pomocí záměny čísel na jednotlivých pozicích posloupnosti. Vzhledem k velikosti stavového prostoru se používá zejména evoluční návrh.

Autor tohoto článku vytvořil systém založený na kartézském genetickém programování (CGP), který umožňuje evoluční návrh polymorfních hradel a jednoduchých polymorfních obvodů využívajících čtyřvývodové ambipolární tranzistory. Byla zvolena reprezentace, kde každý uzel může plnit funkci propojky, n-MOS, p-MOS nebo ambipolárního tranzistoru. Některé vstupy uzlů jsou nevyužité – viz Obrázek 3.



Obrázek 3: Funkce uzlů v CGP navrženého systému: (a) ambipolární transistor, (b) p-MOS transistor, (c) n-MOS transistor, (d) propojka.

Graf je rozdělen do mřížky s *m* sloupci a *n* řádky uzlů, každý uzel má pomyslné tři vstupy a jeden výstup. Vstupy

mohou být zapojeny na výstupy uzlů z přecházejících sloupců nebo vstupy celého obvodu, mezi něž spadá i napájecí napětí obvodu. Výstup (popř. výstupy) celého obvodu může být připojen na libovolný vstup celého obvodu nebo na výstup některého z uzlů. Takovouto reprezentací je možné zapsat libovolný obvod složený ze zmíněných tranzistorů.

Evoluční návrh pracuje na principu generování a testování mnoha kandidátních řešení, proto má výkonnost simulátoru významný vliv na škálovatelnost celého evolučního přístupu. Kandidátní řešení je možné evaluovat použitím přesného SPICE simulátoru [5], rekonfigurovatelného analogového obvodu [6], Žaloudek navrhl přístup založený na jednoduchém simulátoru [7]. Žádný ze zmíněných postupů však není ideální – přesná simulace pomocí SPICE je časově náročná, u ostatních návrhů simulace se projevovala odchylka od reality. Z tohoto důvodu byl po vzoru [8] implementován vlastní diskrétní simulátor, který vychází z vícehodnotové simulace a snaží se co nejvíce reflektovat chování tranzistorů včetně degradace signálů. V případě rozpoznání zkratu dokáže vyhodnocení obvodu předčasně ukončit.

Implementovaný systém je schopný navrhovat polymorfní obvody a hradla řízená napájecím napětím i napětím na speciálním signálu. Při návrhu je možné zvolit požadovaný typ ambipolárních tranzistorů – typ 1 i typ 2. Systém pomocí evoluce hledá obvod, který pro dané kombinace vstupních vektorů generuje požadované výstupní hodnoty. Ty jsou vždy nedegradované. Pokud je obvod nalezen, pokračuje se s evolucí dále s cílem nalézt řešení s minimálním počtem tranzistorů. Po předem daném počtu generací je evoluce vždy ukončena.

Nalezené obvody s minimálním počtem tranzistorů často obsahují tranzistory, které spínají některé vstupní signály přímo na výstupní, proto umožňuje systém také hledání obvodů s vysokou vstupní a nízkou výstupní impedancí. Dalším rozšířením je například omezení zapojení *polarity gate* ambipolárních tranzistorů na vodič řídící polymorfní funkci obvodu nebo jeho negaci, čímž by se mohl zjednodušit návrh čipu s ambipolárními tranzistory. U obvodů řízených polaritou napájecího napětí jsou to právě oba vývody napájecího napětí.

V. Dosažené výsledky

Systém popsaný v předchozí kapitole byl využit k návrhu polymorfních hradel. Byla generována hradla řízená speciálním signálem i hradla řízená polaritou napájecího napětí. Z důvodů popsaných v kapitole 2 byl však důraz kladen především na druhý způsob ovládání polymorfních hradel.

A. *Polymorfní hradla řízená polaritou napájecího napětí složená z ambipolárních a MOSFET tranzistorů*

Aby bylo zjištěno, která hradla řízená polaritou napájecího napětí lze vůbec pomocí ambipolárních tranzistorů vytvořit, byla zahájena evoluce vybraných hradel bez jakýchkoliv omezení. K tomuto účelu bylo vybráno celkem 36 polymorfních hradel vytvořených kombinacemi identity (id), negace (not) a základních dvouvstupných hradel (and, nand, or, nor, xor, xnor). Id/id značí polymorfní multiplexor, not/not polymorfní invertor. Hradla and/and, nand/nand apod. jsou invariantní vůči záměně napájecích přívodů obvodu.

Jak je možné vidět v Tabulce 1, pro libovolnou vstupní a výstupní impedanci a ambipolární tranzistory typu 1 bez omezení na zapojení vývodu polarity gate byla nalezena všechna polymorfní hradla řízená změnou polarity napájecího napětí, včetně multiplexoru.

Tabulka 1: Minimální počet tranzistorů (MOSFET a ambipolárních typu 1) nutných pro konstrukci polymorfních hradel řízených polaritou napájecího napětí

	Id	**Not**	**And**	**Nand**	**Or**	**Nor**	**Xor**	**Xnor**
Id	4	4	7	6	7	6	8	8
Not	-	2	6	6	6	6	7	7
And	-	-	5	7	5	7	7	7
Nand	-	-	-	4	7	4	7	7
Or	-	-	-	-	9	7	7	7
Nor	-	-	-	-	-	4	7	7
Xor	-	-	-	-	-	-	6	4
Xnor	-	-	-	-	-	-	-	6

B. *Polymorfní hradla řízená polaritou napájecího napětí složená z ambipolárních tranzistorů*

Při návrhu obvodů je vhodné, aby byly všechny tranzistory stejného typu a nekombinovaly se různé technologie, neboť kombinace technologií komplikuje výrobu obvodu. Z 36 hradel navrhovaných v předchozí podkapitole byly evolucí nalezeny čtyři, které využívají pouze ambipolární tranzistory: nand/nor, and/or, xor/xnor a invertor. Minimální počty ambipolárních tranzistorů nutných pro jejich realizaci zobrazuje tabulka 2.

Tabulka 2: Minimální počet ambipolárních tranzistorů nutných pro konstrukci polymorfních hradel řízených polaritou napájecího napětí

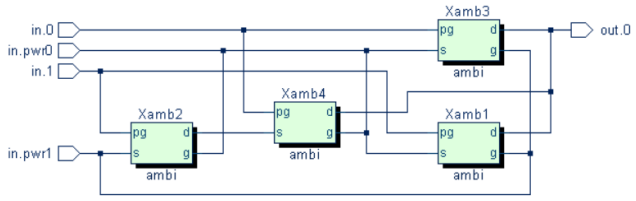
	Typ ambipolárních tranzistorů	
	Typ 1	**Typ 2**
Not/not	2	2
Nand/nor	4	4
And/or	5	3
Xor/xnor	4	5

Polymorfní hradlo nand/nor se 4 ambipolárními tranzistory již představili Yang a Mohanram [9]. Stejně tak popsali hradlo xor/xnor se 4 tranzistory. Toto hradlo však předpokládá existenci negací obou vstupních signálů. Pro realizaci by ho tedy bylo nutné doplnit dvěma invertory, čímž by počet tranzistorů vzrostl na 8. Evolučním návrhem byla nalezena hradla xor/xnor se **4 až 8 tranzistory** v závislosti na použitém typu ambipolárních tranzistorů, nárocích na vstupní a výstupní impedanci a rozpoznání degradovaných úrovní na control a polarity gate tranzistorů.

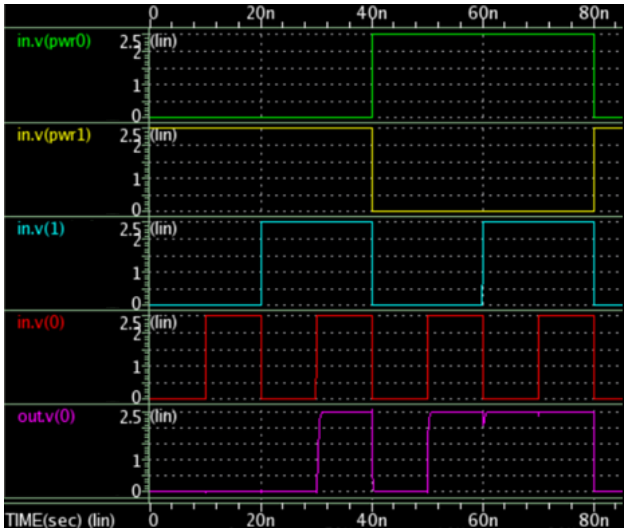
Polymorfní hradlo and/or nebylo dosud nikde prezentováno, jednoduše by ho však bylo možné realizovat 6 tranzistory pomocí hradla nand/nor a invertoru. Evolučním návrhem vznikla hradla and/or se **3 až 6 tranzistory**. K realizaci hradla and/or s vysokou vstupní impedancí stačí například pouze čtyři tranzistory typu 2 – viz Obrázek 4.

Evolvovaná hradla byla ověřena simulačním nástrojem HSPICE pomocí modelů chování obou typů ambipolárních

tranzistorů. Výsledek simulace obvodu z Obrázku 4 je zobrazen na Obrázku 5.



Obrázek 4: Polymorfní hradlo and/or s vysokou vstupní a nízkou impedancí řízené polaritou napájecího napětí



Obrázek 5: Výsledek simulace hradla and/or z Obrázku 4.

VI. Závěr

Byla vytvořena metoda, která umožňuje evoluční návrh polymorfních hradel přímo na úrovni tranzistorů. Pomocí této metody bylo navrženo několik hradel s menším počtem tranzistorů, než využívají dosud publikovaná. V tuto chvíli je v přípravě článek o vytvořeném systému pro návrh polymorfních hradel, stejně tak článek s evolvovanými hradly and/or a xor/xnor. V průběhu práce se zrodila myšlenka na implementaci polymorfní elektroniky řízené změnou polarity napájecího napětí čistě pomocí MOSFET tranzistorů, jejichž chování je dobře známé, existují přesné simulační modely a v dnešní době není problém čipy s těmito tranzistory jednoduše vyrobit. Evoluční návrh v implementovaném systému ukázal, že mnoho takovýchto hradel vytvořit lze, čímž se otevírá další možný směr polymorfní elektroniky.

Cíle disertační práce

Experimenty popsané v tomto článku ukázaly zajímavou možnost ovládat polymorfní hradla složená z MOSFET tranzistorů pomocí změny polarity napájecího napětí. Tímto směrem by se tedy mohl vydat další výzkum, který by prozkoumal možnosti, jenž tento směr přináší.

Do této chvíle se polymorfní hradla s běžnými MOSFET tranzistory realizovala jako analogové obvody, byla snaha měnit funkci například zvýšením nebo snížením napájecího napětí. S myšlenkou polymorfních hradel ovládaných změnou polarity napájecího napětí, která jsou složena čistě z MOSFET tranzistorů, dosud nikdo nepřišel, nebyla tedy zkoumána. V řadě aplikací je přehození polarity obvodu nepraktické a mohlo by ohrozit ostatní části systému, nicméně v některých třídách aplikací jako watermarking, PUF nebo diagnostika přehození polarity vadit nemusí.

Střídání polarity napájecího napětí (záměna napájecích přívodů) navíc na rozdíl od "analogových" polymorfních hradel vykazuje digitální chování, čímž by se nemusely projevit nectnosti původních analogových polymorfních hradel s MOSFET tranzistory. Došlo by k úspoře místa na čipu (rozdíly velikostí tranzistorů u analogových obvodů byly až desetinásobné), ke snížení spotřeby (žádné stavy systému by cíleně neobsahovaly zkrat) a výstupní napěťové úrovně hradel by odpovídaly výstupním úrovním klasických obvodů z MOSFET tranzistorů. V současné době jsou navíc oproti čtyřvývodovým ambipolárním tranzistorům dostupné simulační modely těchto tranzistorů, což by umožnilo přesnou simulaci navržených obvodů.

Poděkování

Tato práce vznikla za podpory národního COST projektu LD14055 Nekonvenční návrhové techniky pro číslicové obvody s vlastní rekonfigurací: od materiálů k implementaci.

Reference

[1] R. Růžička, "Polymorfní elektronika," habilitační práce, FIT VUT v Brně, 2011.

[2] R. Tesař, "Komponenty pro polymorfní číslicové obvody na bázi ambipolárních tranzistorů," Sborník příspěvků PAD2014, Malá Skála: Liberec University of Technology, 2014, pp. 25-32.

[3] O. Turkyilmaz, F. Clermidy, L. G. Amaru, P.-E. Gaillardon, and G. De Micheli, "Self-Checking Ripple-Carry Adder with Ambipolar Silicon NanoWire FET," 2013 IEEE International Symposium on Circuits and Systems (ISCAS2013), pp.2127-2130, May 2013.

[4] M. H. Ben-Jamaa, K. Mohanram, and G. De Micheli, "An Efficient Gate Library for Ambipolar CNTFET Logic," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 30, no. 2, pp. 242-255, Jan. 2011.

[5] J. Walker, J. Hilder, and A. Tyrrell, "Towards evolving industry-feasible intrinsic variability tolerant CMOS designs," IEEE Congress on Evolutionary Computation, 2009, pp. 1591–1598.

[6] M. Trefzer, "Evolution of Transistor Circuits," dizertační práce, Ruprecht-Karls-Universität Heidelberg, 2006.

[7] L. Žaloudek L., and L. Sekanina, "Transistor-Level Evolution of Digital Circuits Using a Special Circuit Simulator," Evolvable Systems: From Biology to Hardware, LNCS, vol. 5216, Springer Verlag, 2008, pp. 320–331.

[8] V. Mrázek, and Z. Vašíček, "Evolutionary Design of Transistor Level Digital Circuits using Discrete Simulation," Genetic Programming, 18th European Conference, EuroGP 2015. Berlin: Springer International Publishing, 2015, pp. 66-77.

[9] X. Yang, K. Mohanram, "Ambipolar electronics," Rice University Technical Report TREE1002, 2010.

Generování testu pro prostředky vestavěné diagnostiky

Robert Hülle
1. ročník prezenčního studia
Školitel: Petr Fišer, školitel specialista: Jan Schmidt

České vysoké učení technické v Praze, Fakulta informačních technologií
Thákurova 9, Praha, 16000
hullerob@fit.cvut.cz

***Abstrakt*—V tomto článku shrnuji své výsledky za 1. rok doktorského studia, porovnání poruchových modelů pro aplikačně specifické testování FPGA, zkoumání vlastností SAT instancí vzniklých v procesu ATPG. Dále nastiňuji další možný směr pokračování výzkumu, generování testu s nulovým aliasingem.**

***Klíčová slova*—test, testování, generování testu, poruchový model, FPGA, ATPG, SAT, kompakce odezvy, aliasing**

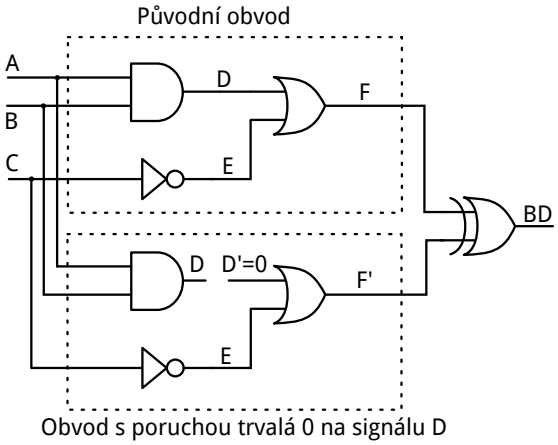
I. Úvod

S rostoucí hustotou integrace a složitosti číslicových obvodů roste také náročnost testování, ať už se jedná o generování testu, dobu aplikace testu či o plochu a spotřebu testovací logiky. Jedním ze způsobů, jak se vypořádat s rostoucí délkou testů a ceny ATE je využití prostředků vestavěné diagnostiky (built-in self-test, BIST).

Nedílnou součástí prostředků vestavěné diagnostiky je kompakce odezvy testovaného obvodu. Kompaktor odezvy je typicky tvořen sekvenčním obvodem, jehož vnitřní stav je ovlivňován výstupem testovaného obvodu. Příkladem takového obvodu je lineární zpětnovazební posuvný registr (LFSR) s paralelními vstupy (MISR).

Mimo klasického problému pokrytí poruch testovacím vektorem zde navíc vzniká problém aliasingu. Aliasing je jev, kdy porucha, která je pokrytá více testovacími vektory, vyvolá chybovou odezvu, která uvede kompaktor do stavu odpovídajícímu bezporuchovému obvodu, čímž klesá reálné pokrytí obvodu. Proti tomuto jevu lze bojovat různými prostředky, například zvětšením periody kompaktoru, či změnou typu kompaktoru [1], [2].

Cílem mé disertační práce je potlačit aliasing již ve fázi generování testovacích vektorů. Základní myšlenka je konstrukce dodatečných omezení výstupu obvodu. Výstup obvodu musí být omezen tak, aby nový testovací vektor generovaný pro nepokrytou poruchu nezpůsobil aliasing poruchy pokryté dřívějším testovacím vektorem. Zde se přímo nabízí generátor testu (ATPG) založený na řešení problému splnitelnosti booleovské formule (SAT), neboť ten umožňuje snadno vytvářet další omezení výstupu, na rozdíl od moderních strukturních algoritmů založených na původním D-algoritmu [3], mezi něž patří PODEM [4], FAN [5], SOCRATES [6] a další.



Obrázek 1. Obvod s modelovanou poruchou trvalá 0. Porucha je detekována, pokud výstup obvodu je 1.

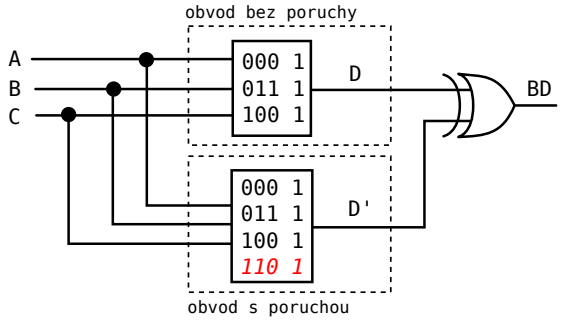
A. SAT ATPG

SAT ATPG překládá problém nalezení testovacího vektoru na řešení problému SAT. Tato ATPG využívají pokroků na poli řešičů SAT, které umožňují rychlé řešení, zejména SAT instancí vznikajících z logických obvodů. Ukazuje se, že SAT ATPG dokáží nalézt testovací vektor, nebo důkaz redundance poruchy rychleji než strukturní ATPG [7], [8].

Základní princip SAT ATPG spočívá v transformaci obvodu tak, aby jeho výstup byl jednotkový právě tehdy, když je na jeho vstup přiveden vektor, který detekuje testovanou poruchu. Toho je docíleno zdvojením obvodu a modelováním poruchy v jedné jeho polovině. Vstupy těchto dvou obvodů jsou společné, výstupy jsou spojeny hradly XOR [9].

Modelování poruchy je závislé na zvoleném poruchovém modelu. Poruchy typu trvalá 0/1 (SA) modelujeme rozpojením poruchového signálu a nastavením tohoto signálu na konstantní hodnotu, pro poruchu trvalá 0 na hodnotu 0, pro poruchu trvalá 1 na hodnotu 1. Příklad je uveden v obrázku 1.

Takovýto obvod je poté přeložen na instanci SAT v konjunktivní normální formě (CNF) jako konjunkce charakteristických funkcí všech hradel v obvodu.



Obrázek 2. Obvod s modelovanou poruchou bit-flip v LUTu.

Během přípravy SAT ATPG pro využití ve svém dalším výzkumu jsem změřil jeho vlastnosti a vlastnosti SAT instancí, které vyprodukoval.

II. SAT ATPG pro aplikačně specifické testování FPGA

Dostupná ATPG typicky podporují omezený poruchový model, nejčastěji model trvalé 0/1. Tento model je ovšem pro testování obvodu v FPGA nedostatečný [10], [11]. Jiné typy poruch se v takovýchto ATPG testují například transformací obvodu tak, aby tyto poruchy byly ekvivalentní (nějaké) poruše trvalá 0/1 v transformovaném obvodu.

Já jsem zvolil jinou cestu a implementoval jsem nový SAT-ATPG, který je schopen přímo pracovat s více poruchovými modely. Při výběru poruchových modelů jsem se zaměřil na aplikačně specifické testování FPGA, tedy testování logiky nahrané v FPGA čipu, bez nutnosti rekonfigurace. Pro tento scénář jsem zvolil modely „trvalá 0/1“ (stuck-at, SA) a „překlopení bitu“ (bit-flip, BF).

Porucha typu BF představuje změnu obsahu paměti vyhledávací tabulky (lookup table, LUT), kdy 1 bit v paměti LUTu je změněn. Tato porucha se projeví jako nesprávná hodnota na výstupu LUTu pro jeden jeho vstupní vektor.

Modelování BF poruchy při překladu obvodu do CNF provádím stejně jako v případě SA, mám tedy zdvojený obvod, s poruchovou a bezporuchovou částí. Na rozdíl od SA není v poruchové části žádná diskontinuita, pouze se v ní liší funkce uzlu (hradla či LUTu) reprezentujícího poruchový LUT. Stejně jako v případě SA je poté SAT řešičem nalezeno ohodnocení signálů, pro které se výstupy obvodů liší. Příklad je vyobrazen v obrázku 2.

Porovnal jsem vzájemný vztah poruch obou modelů, SA a BF, zejména jejich vzájemnou dominanci. Porucha SA je dominována poruchou BF vždy, pokud se nachází na vstupu či výstupu LUTu, nemusí však být dominována v jiných částech obvodu, například na vstupech či výstupech klasických hradel, například v obvodech pro rychlou propagaci přenosu (fast carry chain), které jsou součástí dnešních FPGA obvodů. Dominance v opačném směru, tedy poruchy BF dominované poruchou SA se v obvodu nevyskytují ve významném počtu, jediný případ, kdy k dominanci může dojít, je LUT implementující funkci s jedním mintermem [12].

Experimenty jsem provedl na 279 obvodech z benchmarků MCNC, LGSynth’91 [13], LGSynth’93, ISCAS’85 [14], ISCAS’89 [15] a IWLS 2005 [16]. Ze sekvenčních obvodů byly extrahovány kombinační části, poté byly obvody syntetizovány nástrojem Xilinx Vivado 2015.2 pro architekturu Artix-7. Tyto syntetizované obvody jsem převzal.

Změřil jsem podíl poruch BF pokrytých poruchami SA a podíl poruch SA pokrytých poruchami BF. Z neredundantních poruch bylo průměrně pokryto 99,6 % poruch SA a 69,5 % poruch BF. Ve většině obvodů bylo pokrytých 100 % poruch SA, s výjimkou obvodů, které obsahovaly primitivní hradla XOR a multiplexory, v některých obvodech byl také přítomný přímý spoj mezi vstupem a výstupem. Pokrytí poruch BF poruchami SA má mnohem větší rozptyl, pohybuje se od 17 % do 100 %

Překvapivý výsledek jsem získal z měření počtu redundantních poruch, kdy průměrný poměr redundantních poruch SA je dle očekávání nízký, 0,33 %. Naopak pro poruchy BF je poměr redundantních poruch průměrně 10,15 %. Celkový poměr všech redundantních poruch je pak 7,42 %.

Množství redundantních poruch v modelu BF je zapříčiněno horší kontrolovatelností a pozorovatelností, než je tomu v případu modelu SA.

Tyto výsledky jsem ve formě článku „SAT-ATPG for Application-Oriented FPGA testing“ poslal na konferenci Baltic Electronics Conference. Článek byl přijat k publikaci [12].

III. Vlastnosti SAT instancí

A. Výpočet charakteristické funkce

Ve výše zmíněném ATPG jsem implementoval dva různé způsoby generování CNF obvodu. Liší se generováním charakteristické funkce uzlu obvodu. Dále jsem měřil vliv minimalizace reprezentace uzlů v SOP na vlastnosti SAT instancí a její celkový vliv na generování testu.

1) Přímé generování charakteristické funkce: Základní myšlenka této metody je v tom, že obvodový uzel popíšeme jako logickou ekvivalenci výstupu a funkce vstupů. Tu poté algebraickými operacemi transformujeme do CNF.

V praxi stačí nalézt reprezentaci funkce uzlu a její komplement (on-set a off-set) ve formě SOP. Na tyto dvě reprezentace lze nazírat tak, že implikují výstup uzlu v hodnotě 1 (pro on-set) případně v hodnotě 0 (pro off-set). Tyto implikace odpovídají implikacím obsaženým ve výše zmíněné ekvivalenci výstupu a funkce vstupů. Použitím DeMorganových pravidel transformujeme implikaci na součin součtů (product of sums, POS), který je v součtu s výstupní proměnnou. Jednoduchou distribucí výstupní proměnné získáme polovinu charakteristické funkce (pro on-set a off-set) v CNF. Příklad takové transformace je uveden na Obrázku 3.

Výstupem tohoto generátoru je CNF s dlouhými klauzulemi, což může být nevýhoda.

2) Generování charakteristické funkce Tseitinovou transformací: Druhá metoda spočívá v Tseitinově transformaci [17] každého uzlu popsaného dvouúrovňovou logikou ve formátu PLA na dvě úrovně uzlů, v první úrovni jsou uzly AND, v druhé úrovni je uzel OR. Tato struktura vychází přímo z

$F(a,b,c) = \{0,2,3,6,7\}$

```
on-set
a b c    y
0 - 0    1   (a ∨ c ∨ y)
- 1 -    1   (¬b ∨ y)

off-set
a b c    y
1 0 -    0   (¬a ∨ b ∨ ¬y)
- 0 1    0   (b ∨ ¬c ∨ ¬y)
```

CNF

$$(a \vee c \vee y) \wedge (\neg b \vee y) \wedge (\neg a \vee b \vee \neg y) \wedge (b \vee \neg c \vee \neg y)$$

Obrázek 3. Příklad funkce zadané výčtem mintermů, její minimalizovaný on-set a off-set ve formě SOP a výsledná charakteristická funkce v CNF.

```
on-set
a b c    y
0 - 0    1
- 1 -    1
Tseitinova transformace
```

$y_1 = (\neg a \wedge \neg c)$
$y_2 = b$
$y = y_1 \vee y_2$

CNF

$$(a \vee c \vee y_1) \wedge (\neg a \vee \neg y_1) \wedge (\neg c \vee \neg y_1) \wedge$$
$$(b \vee \neg y_2) \wedge (\neg b \vee y_2) \wedge$$
$$(\neg y_1 \vee y) \wedge (\neg y_2 \vee y) \wedge (y_1 \vee y_2 \vee \neg y)$$

Obrázek 4. Příklad výpočtu charakteristické funkce v CNF pomocí Tseitinovy transformace.

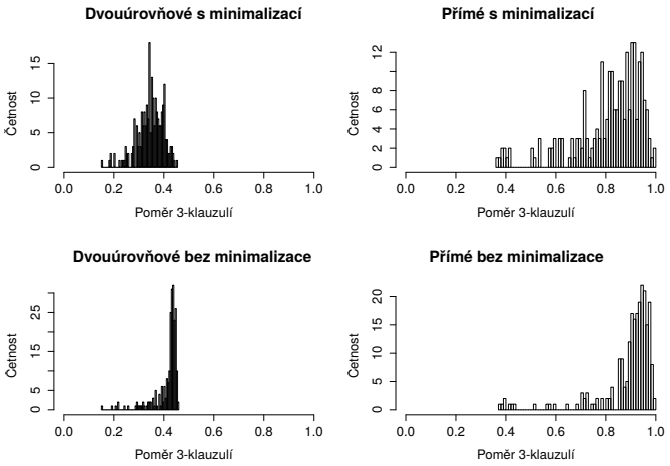
popisu uzlu jako součtu součinů. Charakteristické funkce tímto způsobem vzniklých virtuálních uzlů mohou být spočítány stejným způsobem výše uvedeným, ale protože se jedná pouze o dva druhy uzlů, můžeme si ušetřit práci použitím přepisovacích pravidel pro hradla AND a OR. Také není třeba počítat komplement (off-set) transformované funkce. Příklad takové transformace je uveden na obrázku 4.

Nevýhodou této verze je zvětšení počtu proměnných ve výsledné SAT instanci. Naopak výhodou je větší poměr dvouliterálových klauzulí.

B. Vlastnosti SAT instancí

3SAT je varianta problému splnitelnosti, kde každá klauzule má právě 3 literály. 3SAT patří mezi NP-úplné problémy a neposkytuje žádnou výpočetní výhodu proti obecnějšímu problému SAT. Vlastnosti 3SAT jsou však lépe pochopeny, konkrétně vztah mezi poměrem počtu klauzulí k počtu proměnných a mezi výpočetní složitostí náhodných 3SAT instancí je znám. Existuje fázový přechod kolem poměru 4,3 kde jsou instance nejtěžší [18], [19].

Podobně 2SAT je problém splnitelnosti, kde každá klauzule má právě 2 literály. Na rozdíl od 3SAT (a SAT) nepatří



Obrázek 5. Průměrná hodnota 2+p SAT u zpracovaných obvodů.

2SAT mezi NP-úplné problémy a jeho výpočetní složitost je polynomiální.

2+p SAT je varianta SAT, kde klauzule mají 2 nebo 3 literály. Pro náhodné instance bylo ukázáno, že pro $p > 0,4$ tedy pro poměr 3-literálových klauzulí, se vlastnosti náhodných instancí více podobají instancím 3SAT a pro $p < 0,4$ se podobají více instancím 2SAT [20].

Pro účely diskuse nad vlastnostmi instancí jsem instance vzniklé ATPG procesem transformoval na instance 2+p SAT tak, že na klauzule delší než 3 literály jsem použil klasickou redukci ze SAT na 3SAT. Tato transformace není třeba pro řešení instance, SAT řešič řeší netransformované instance.

C. Výsledky

V experimentech jsem použil 230 obvodů ze stejného benchmarku, zmíněného v předchozí sekci.

Pro každý obvod jsem vygeneroval SAT instance popisující jednotlivé poruchy, protože se vždy jedná o stejný obvod, lišící se pouze v místě poruchy, je rozptyl vlastností instancí malý, lze ho reprezentovat jednou hodnotou, průměrem. Na obrázku 5 jsem vyobrazil histogram poměru 2+p přes všechny obvody, pro obě metody generování charakteristické funkce, *s* i *bez* minimalizace reprezentace logických uzlů.

Je zřejmé, že použití metody využívající Tseitinovy transformace vede na instance s větším zastoupením 2-literálových klauzulí, kdežto metoda přímého generování vede na instance, ve kterých výrazně převažují 3-literálové klauzule. Minimalizace booleovských funkcí uzlů v obou případech mírně posunuje histogram k více 2-literálovým klauzulím. Změřený poměr naznačuje, že instance generované přímo by se měly chovat jako 3SAT, kdežto instance generované Tseitinovou transformací leží kolem fázového přechodu $p = 0,4$, chování takových instancí by tedy mělo být mezi 2SAT a 3SAT.

Stejným způsobem jsem zkoumal poměr počtu klauzulí k počtu proměnných. Při použití Tseitinovy transformace vyšel průměrný poměr 2,19 *s* a 2,16 *bez* minimalizace. Pro přímé generování jsem naměřil hodnoty 1,58 *s* a 1,43 *bez* minima-

lizace. V tomto případě neměla minimalizace velký vliv na poměr počtu klauzulí. Takovýto poměr klauzulí k proměnným v náhodném 3SAT vede na instance, které jsou téměř všechny splnitelné [18], poměr nesplnitelných instancí v měřených obvodech je však mnohem větší, což mě vede k domnění, že metriky vytvořené při zkoumání náhodných SAT instancí nemusí být nutně vhodné pro analýzu instancí vytvořených v ATPG procesu.

Změřil a porovnal jsem dobu generování a řešení SAT instancí. Jako řešič SAT instancí jsem použil Minisat, pro minimalizaci logických uzlů jsem použil Espresso.

Z dvou metod generování klauzulí nevychází ani jedna statisticky významně lepší, než druhá. Použití minimalizace ovšem vede na instance, které řešič vyřeší rychleji až o řád. Minimalizace má větší vliv na přímý generátor, nejrychleji řešené instance jsou tedy přímo generované a minimalizované.

Při srovnání doby celého generování testu, tedy včetně generování instancí, je situace jiná. Oba generátory bez minimalizace mají stejný výkon, s minimalizací roste doba generování SAT instancí víc, než uspoříme za běhu SAT řešiče.

Tyto výsledky jsem ve formě článku „On Properties of ATPG SAT Instances“ poslal na konferenci Euromicro Conference on Digital System Design. Článek získal jednu kladnou recenzi, dvě záporné recenze a nebyl přijat.

IV. Další směřování výzkumu

Pro snížení aliasingu v kompakci odezvy potřebuji poruchovou simulaci pro všechny pokryté poruchy a simulaci samotného kompaktoru odezvy. Ze znalosti stavu kompaktoru pro všechny poruchy a pro bezporuchový obvod spočítám, které odezvy obvodu by vedly na aliasing, a tyto odezvy zakóduji do SAT instance jako zakázané.

Tento přístup ovšem trpí jedním problémem: je potřeba znát stav kompaktoru po vygenerování následujícího testovacího vektoru již během jeho generování. Tento problém lze vyřešit extrakcí kombinační části kompaktoru, tedy simulace následujícího stavu je součástí výpočtu testovacího vektoru. Konceptuální obvod odražený v generované SAT instanci má tedy kromě výstupů testovaného obvodu (a jeho poruchového obrazu) také výstupy reprezentující budoucí stav kompaktoru. Podobně jako se musí lišit výstup bezporuchového obvodu a obvodu s poruchou, musí se také lišit výstup kompaktoru.

Takové řešení ovšem vede k dalším problémům, k nárůstu velikosti SAT instancí. Tento problém je potřeba analyzovat a zjistit, zda a nakolik překáží. Případně navrhnout způsob jeho zmírnění či odstranění, například výpočet stavu kompaktoru jen pro poruchy, kde hrozí aliasing a s tím související identifikace takových poruch.

Výsledkem by měla být metoda generování testu, která bude minimalizovat aliasing bez nutnosti zásahu do architektury kompaktoru. Dále by tato metoda měla jít použít i pro minimalizaci klopných obvodů v kompaktoru bez zvětšení míry aliasingu.

Poděkování

Autor děkuje za poskytnuté výpočetní a úložné zdroje programu Metacentra CESNET LM2015042 a CERIT-CS CZ.1.05/3.2.00/08.0144.

Tento výzkum byl částečně podporován z projektu ČVUT SGS16/121/OHK3/1T/18.

Tento výzkum byl částečně podporován z grantu české grantové agentury GA16-05179S.

Literatura

[1] P. D. Hortensius, R. D. McLeod, and H. C. Card, “Cellular automata-based signature analysis for built-in self-test,” *IEEE Transactions on Computers*, vol. 39, no. 10, pp. 1273–1283, Oct 1990.

[2] C. Stroud, *A Designer’s Guide to Built-in Self-Test*, ser. Frontiers in Electronic Testing. Springer, 2002.

[3] J. Roth, “Diagnosis of automata failures: A calculus and a method,” *IBM Journal of Research and Development*, vol. 10, no. 4, pp. 278–291, July 1966.

[4] P. Goel, “An implicit enumeration algorithm to generate tests for combinational logic circuits,” *IEEE Transactions on Computers*, vol. C-30, no. 3, pp. 215–222, March 1981.

[5] H. Fujiwara and T. Shimono, “On the acceleration of test generation algorithms,” *IEEE Transactions on Computers*, vol. C-32, no. 12, pp. 1137–1144, Dec 1983.

[6] M. Schulz, E. Trischler, and T. Sarfert, “SOCRATES: a highly efficient automatic test pattern generation system,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 7, no. 1, pp. 126–137, Jan 1988.

[7] R. Drechsler, G. Fey, D. Tille, and S. Eggersglüß, *Test Pattern Generation using Boolean Proof Engines*, 1st ed., 12 2009.

[8] S. Eggersglüß and R. Drechsler, “Robust algorithms for high quality Test Pattern Generation using Boolean Satisfiability,” in *IEEE International Test Conference*, Nov. 2010, pp. 1 –10.

[9] T. Larrabee, “Test pattern generation using Boolean satisfiability,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 11, no. 1, pp. 4–15, Jan. 1992.

[10] M. Rebaudengo, S. Reorda, Matteo, and M. Violante, “A new functional fault model for FPGA application-oriented testing,” in *17th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, 2002, pp. 372–380.

[11] J. Borecký, M. Kohlík, P. Kubalík, and H. Kubátová, “Fault models usability study for on-line tested FPGA,” in *14th Euromicro Conference on Digital System Design (DSD)*, Aug 2011, pp. 287–290.

[12] R. Hülle, P. Fišer, J. Schnmidt, and J. Borecký, “SAT-ATPG for Application-Oriented FPGA testing,” in *The 15th Biennial Baltic Electronics Conference*, Oct 2016.

[13] S. Yang, “Logic synthesis and optimization benchmarks user guide: Version 3.0,” Jan. 1991.

[14] F. Brglez and H. Fujiwara, “A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran,” in *Proceedings of IEEE Int’l Symposium Circuits and Systems (ISCAS’85)*. IEEE Press, Piscataway, N.J., 1985, pp. 677–692.

[15] F. Brglez, D. Bryan, and K. Kozminski, “Combinational profiles of sequential benchmark circuits,” in *IEEE International Symposium on Circuits and Systems, 1989*, May 1989, pp. 1929–1934 vol.3.

[16] C. Albrecht, “IWLS 2005 benchmarks,” Tech. Rep., June 2005.

[17] G. Tseitin, “On the complexity of derivation in propositional calculus,” in *Automation of Reasoning*, ser. Symbolic Computation, J. Siekmann and G. Wrightson, Eds. Springer Berlin Heidelberg, 1983, pp. 466–483.

[18] B. Selman, “Stochastic search and phase transitions: AI meets physics,” in *International Joint Conference on Artificial Intelligence*, vol. 1. Morgan Kaufmann Publishers Inc., 1995, pp. 998–1002.

[19] J. Balcárek, P. Fišer, and J. Schmidt, “On properties of SAT instances produced by SAT-based ATPGs,” in *Doctoral Workshop on Mathematical and Engineering Methods in Computer Science (MEMICS)*, 2009, pp. 3–10.

[20] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky, “2+p-SAT: relation of typical-case complexity to the nature of the phase transition,” *Random Structures & Algorithms*, vol. 15, no. 3-4, pp. 414–435, 1999.

Číslicový návrh spojující odolnost proti útokům a odolnost proti poruchám

Vojtěch Miškovský
1. ročník, prezenční studium
Školitel: Hana Kubátová, školitel specialista: Martin Novotný

České vysoké učení technické v Praze, Fakulta informačních technologií
Thákurova 9, 160 00 Praha 6
miskovoj@fit.cvut.cz

***Abstrakt*—Tento výzkum se zabývá možnostmi, jak zkombinovat metody číslicového návrhu pro odolnost proti poruchám a odolnost proti útokům. Tyto vlastnosti se mohou často navzájem potlačovat, jejich dosažení je navíc často doprovázeno výrazným zvýšením plochy a spotřeby. V současné době se výzkum zaměřuje na vzájemný vliv obou vlastností, v budoucnu bude cílem vytvořit návrhovou metodu zlepšující obě vlastnosti současně.**

***Klíčová slova*—odolnost proti útokům, odolnost proti poruchám, FPGA, DPA, AES, spolehlivost, bezpečnost**

I. Úvod

Častým požadavkem na číslicový návrh je odolnost proti poruchám. Za tímto účelem bylo vymyšleno velké množství návrhových metod [1], [2] Dalším důležitým aspektem je bezpečnost. V případě číslicového návrhu se bezpečností rozumí zejména odolnost proti útokům postranními kanály. I pro ni existuje dostatek návrhových technik, např. [3], [4].

Cílem tohoto výzkumu je vyšetřit vliv návrhových technik pro odolnost proti poruchám na odolnost proti útokům a opačně. Na základě získaných poznatků by pak měly vzniknout nové metody pro číslicový návrh zvyšující odolnost proti útokům i proti poruchám, které by oproti kombinaci současných technik měly snížit režii plochy a spotřeby. Výsledky by tedy měly být využitelné pro návrh bezpečnějších a spolehlivějších kryptografických systémů s nižšími náklady.

V současné době se zabýváme zejména vlivem pasivní hardwarové redundance na odolnost proti diferenciální příkonové analýze (Differential Power Analysis - DPA). Tato skupina spolehlivostních technik se vyznačuje tím, že využívá více shodných funkčních modulů, jejichž výstupy porovnává, díky čemuž může dojít k detekci či zamaskování poruchy. Zároveň však dochází k několikanásobnému zvýšení plochy a zejména spotřeby, což může ovlivnit právě odolnost proti útoku pomocí DPA.

Vyhodnocení tohoto vlivu je realizováno pomocí šifry AES implementované v FPGA. Vzhledem k tomu, že nelze dostatečně přesným způsobem modelovat chování spotřeby FPGA (ať už z důvodu jejich komplexnosti či nedostatečné zdokumentovanosti), měříme tento vliv experimentálně. Konkrétně jde o vyhodnocení závislosti počtu měření spotřeby nutných k získání šifrovacího klíče na využitých spolehlivostních návrhových metodách.

Článek se bude věnovat nejprve představením současných technologií (sekce II), dále si popíšeme implementaci šifrovacího zařízení, na které budeme útočit (sekce III), ukážeme si platformu pro realizaci útoku (sekce IV) a na závěr si ukážeme samotné výsledky měření (sekce V), vyvodíme závěry (sekce VI) a nastíníme možná budoucí pokračování výzkumu (sekce VII).

II. Současné technologie

V této sekci se seznámíme s existujícími technikami a technologiemi: návrhovými metodami pro odolnost proti poruchám, šifrou AES, diferenciální příkonovou analýzou.

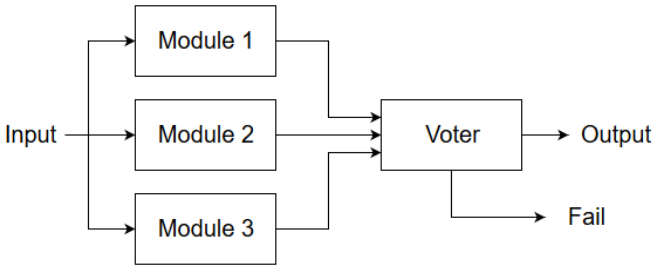
A. Návrhové metody odolné proti poruchám

Jak bylo zmíněno v úvodu, zabýváme se pasivní hardwarovou redundancí, zejména pak duplexem, TMR a NMR.

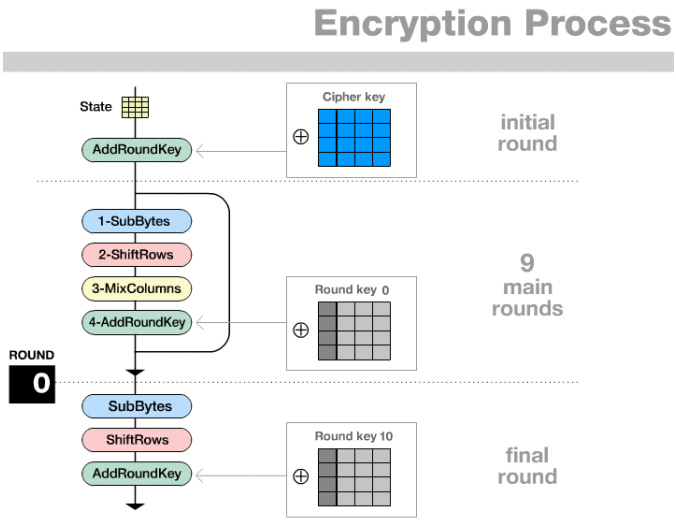
1) Duplex: Nejjednodušší spolehlivostní metoda, která funguje tak, že funkční modul je v zařízení umístěn dvakrát a výstupy obou kusů jsou porovnávány. V případě, že se vstupy neshodují, došlo na jednom z modulů k poruše. Z toho tedy vyplývá, že duplex slouží pouze pro detekci poruchy jednoho modulu, nebo i obou za předpokladu, že se nejedná o shodné poruchy.

2) TMR: TMR, anglicky Triple Modular Redundancy, je metoda využívající tři shodné funkční moduly a takzvaný majority voter. Ten vyhodnocuje výstupy všech tří modulů a volí z nich takový, který je shodný nejméně pro dva moduly. Tato metoda tedy dokáže zamaskovat poruchu jednoho z modulů a detekovat poruchy ve více modulech, pokud tyto poruchy nejsou shodné. Schéma TMR naleznete na obrázku 1.

3) NMR: NMR (N-Modular Redundancy) je v podstatě zobecněním TMR. Pro tuto metodu potřebujeme $N = k \times 2 + 1$ modulů. V tomto případě majority voter vybere takový výstup, který je shodný pro $k + 1$ modulů. Dokáže tedy zamaskovat poruchu v k modulech.



Obrázek 1. Schéma TMR



Obrázek 2. Schéma AES se 128bitovým klíčem

B. AES

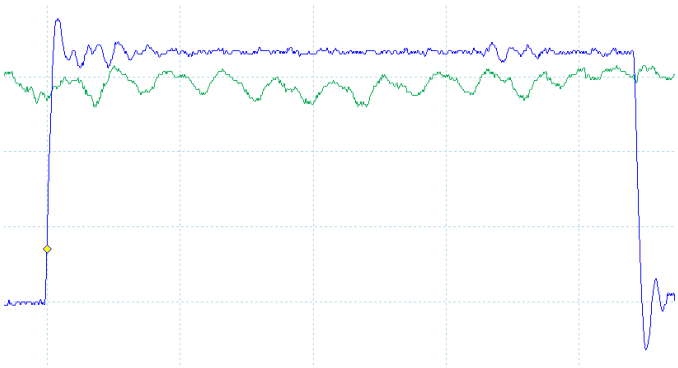
Jako šifru, na kterou budeme útočit, jsme použili AES. Jedná se o nejčastěji používanou blokovou šifru, která se používá například pro zabezpečení bezdrátových sítí Wi-Fi. Šifra využívá pevnou velikost bloku 128 bitů. Existují tři varianty lišící se délkou klíče - 128 bitů, 192 bitů a 256 bitů. Pro naše účely jsme zvolili tu nejjednodušší, 128bitovou.

AES s délkou klíče 128 bitů se skládá z deseti rund a jedné inicializační rundy. Pro použití DPA je důležité, že inicializační runda pouze aplikuje funkci XOR na otevřený text s klíčem. Další rundy se skládají z několika dílčích funkcí a využívají rozdílné rundovní klíče odvozené z původního klíče. Pro nás je důležitá funkce SubBytes. To je nelineární bijektivní funkce, čehož lze s výhodou využít při DPA, jak si ukážeme později. Schéma celého šifrovacího procesu je na obrázku 2.

Kompletní popis šifry lze dohledat ve specifikaci [5].

C. Differential Power Analysis

DPA patří mezi útoky postranními kanály. Tyto útoky jsou specifické tím, že neútočí na kryptografické principy šifry, ale na její implementaci. DPA je založeno na analýze průběhu spotřeby. Pro provedení útoku je zapotřebí znalost otevřeného textu nebo šifrovaného textu. Celý útok bychom mohli rozdělit na tři fáze:



Obrázek 3. Graf průběhu spotřeby v čase jednoho šifrování na FPGA (zeleně) a triggeru (modře)

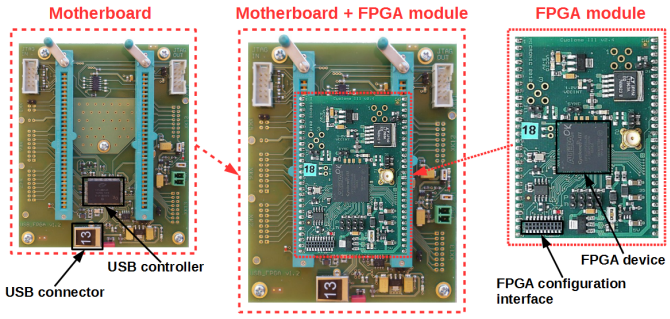
Měřící
: V této fázi je třeba změřit průběh spotřeby šifrovacího zařízení pro různé otevřené texty. Množství těchto průběhů potřebných k získání klíče je jeden z ukazatelů odolnosti proti DPA. Příklad grafu jednoho průběhu spotřeby je na obrázku 3.

Hypotetická
: Pro všechny naměřené průběhy je potřeba spočítat hypotetickou hodnotu, na které je přímo závislá spotřeba, která by tedy pro velké množství průběhů měla se spotřebou korelovat. Obvykle se bere hodnota závislá na jednom bytu klíče a jednom bytu otevřeného textu. V našem případě počítáme hammingovu váhu funkce SubBytes aplikované na otevřený text XOR šifrovací klíč, což odpovídá inicializační rundě a prvnímu kroku první rundy AES. Tato hodnota je tedy závislá na klíči i otevřeném textu, navíc je funkce SubBytes nelineární, díky čemuž nebude spotřeba korelovat se sousedními hodnotami klíče. Hypotetickou hodnotu musíme spočítat pro každou možnou hodnotu klíče (0-255) a každý byte klíče (1-16).

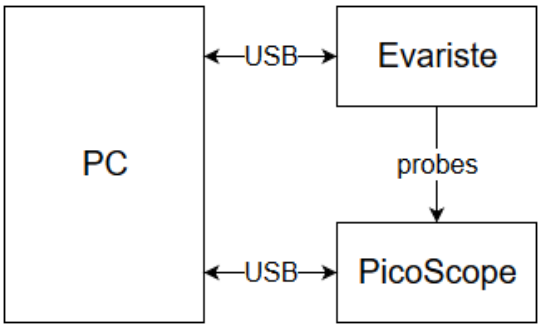
Výpočetní
: Nyní je potřeba najít korelaci mezi reálnou spotřebou a hypotetickou spotřebou. Z prvních dvou fází jsme získali matice reálné spotřeby (otevřené texty $\times$ měření v čase) a hypotetických hodnot (možné hodnoty klíče $\times$ otevřené texty). Výpočtem korelace (Pearsonova korelačního koeficientu) mezi vektory těchto matic získáme matici korelací (možné hodnoty klíče $\times$ měření v čase), kde nejvyšší absolutní hodnota by měla odpovídat danému bytu klíče a času, kdy se tento nejvíce projevil na spotřebě. Toto musíme provést pro každý byte klíče. [6],[7]

III. Implementace

Pro implementaci útoku byla zvolena platforma Evariste III [8] s využitím FPGA modulu s čipem Altera Cyclone III. Tato platforma byla vytvořena pro vytváření generátorů náhodných čísel, ale také pro analýzu spotřeby. Zařízení obsahuje USB



Obrázek 4. Platforma Evariste III



Obrázek 5. Hardwarové schéma měření

řadič, který je využit pro nahrávání klíče a otevřeného textu. Platforma Evariste III je zobrazena na obrázku 4.

Pro účely měření jsem vytvořil vlastní implementaci šifry AES v jazyce VHDL a dále moduly pro implementaci variant odolných proti poruchám, duplex TMR a NMR.

IV. Měření

Popis měření můžeme rozdělit na hardwarovou část a softwarovou část.

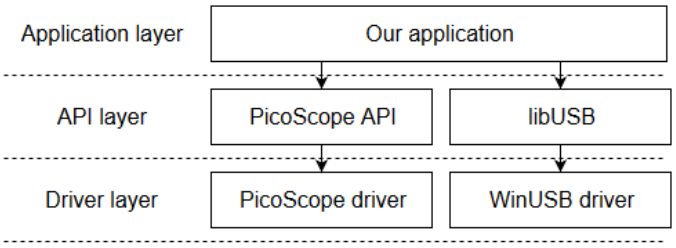
A. Hardware

Měření spotřeby bylo prováděno pomocí osciloskopu PicoScope 6404D. Tento osciloskop podporuje až 2,5GS/s. Data z osciloskopu byla získávána prostřednictvím USB. Taktéž pomocí USB probíhala komunikace se zařízením Evariste. Schéma hardwarové platformy je vyobrazeno na obrázku 5.

B. Software

Nejprve bylo pro měření využíváno PicoScope software dodávaného s osciloskopem a pro výpočty Wolfram Mathematica 10. Tuto kombinaci využíváme na fakultě pro DPA na SmartCard, ovšem pro potřeby FPGA, kde je pro získání klíče potřeba daleko více průběhů (otevřených textů), se ukázalo být nedostatečné, bylo tudíž potřeba vytvořit jiné řešení.

Pro měření i výpočet jsem naprogramoval aplikaci v C++, která využívá PicoScope API pro komunikaci s osciloskopem prostřednictvím jeho ovladače a knihovnu libusb pro komunikaci se zařízením Evariste za použití ovladače WinUSB. Tato architektura je znázorněna na obrázku 6.



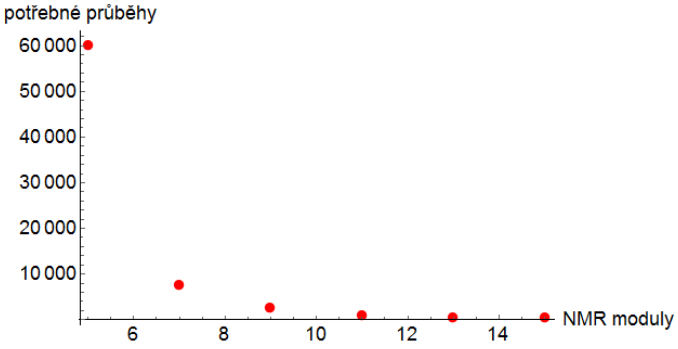
Obrázek 6. Sotwarové schéma měřicí aplikace

Tabulka I
POROVNÁNÍ ČASU V SEKUNDÁCH POTŘEBNÉHO PRO VÝPOČET DPA PŘI POUŽITÍ RŮZNÉHO SOTFWARU PRO RŮZNÉ POČTY MĚŘENÍ

	100	**1,000**	**10,000**	**100,000**	**1,000,000**
Math.&Pico.	28	266	7,496	n/a	n/a
C++ app	1	12	118	1,729	21,361

Tabulka II
ZÁVISLOST POČTU NAMĚŘENÝCH PRŮBĚHŮ SPOTŘEBY NUTNÝCH K ZÍSKÁNÍ KLÍČE PRO RŮZNÉ POČTY MODULŮ V NMR

počet modulů v NMR	5	7	9	11	13	15
počet měření	60,000	7,500	2,500	800	500	495



Obrázek 7. Graf závislosti počtu naměřených průběhů spotřeby nutných k získání klíče pro různé počty modulů v NMR

Aplikace využívá pro výpočet Pearsonova korelačního koeficientu jednoduchý sekvenční algoritmus, který je urychlen předpočítáním průměrů a rozptylů jednotlivých sloupců matic, neboť tyto jsou využívány opakovaně.

Naše aplikace, ač neoptimalizovaná, se ukázala být několikanásobně rychlejší než využití kombinace PicoScope software a Wolfram Mathematica, jak ukazuje tabulka I.

V. Výsledky

V současné době se nám podařilo získat šifrovací klíč pouze při nasazení NMR pro 5 a více modulů. Počet naměřených průběhů nutný pro získání klíče naleznete v tabulce II. Čím více potřebných průběhů, tím je aplikace odolnější proti útoku.

Z grafu na obrázku 7 se zdá, že závislost počtu potřebných měření na počtu modulů je exponenciální. To lze snadno vysvětlit. Aby se do zařízení vešel potřebný počet modulů, byla použita implementace pouze části šifry, konkrétně inicia-

lizační XOR otevřeného textu a klíče a dále funkce SubBytes, což je přesně ta část šifry, na kterou útočíme pomocí DPA. Bohužel to také znamená, že výstupem této části šifry je právě ta hodnota, kterou počítáme v hypotetické fázi útoku (viz II-C). Tento výstup se následně propaguje do majority voteru, který má faktoriální plochu (porovnává všechny $(N-1)/2$ velké podmnožiny výstupů modulů) a tedy zřejmě i spotřebu. Toto je nejpravděpodobnější vysvětlení exponenciálně vypadající závislosti, která je ve skutečnosti spíše faktoriální. Tyto výsledky ovlivněné neúplnou implementací AES tedy mají nízkou vypovídající hodnotu.

Pro řešení této situace bude potřeba výstup před vstupem do voteru zamaskovat nelineární funkcí (například aplikací funkce SubBytes podruhé) a měření provést znovu, aby výsledky lépe odpovídaly reálné implementaci. Je bohužel velmi pravděpodobné, že změna způsobí výrazný nárůst potřebného počtu měření, které tedy bude potřeba optimalizovat.

VI. Závěr

Aplikace vytvořená pro urychlení měření i výpočtu diferenciální příkonové analýzy se ukázala být více než desetinásobně rychlejší oproti kombinaci softwaru dodávaného s osciloskopem a Wolfram Mathematica, čímž plní svůj účel. Toto zrychlení je zřejmě způsobeno vyšší rychlostí kompilovaného programu oproti Mathematicou interpretovanému skriptu. S ohledem na velký rozdíl mezi náročností DPA na FPGA a SmartCard však bude vhodné provést další optimalizace této aplikace, zejména s ohledem na paralelizmus a efektivní využití pamětí cache procesoru. Vzhledem k povaze výpočtu lze zvážit i využití výpočtů na GPU.

Naměřené výsledky pro NMR s různým počtem modulů vykazují zdánlivě exponenciální závislost počtu potřebných naměřených průběhu na počtu modulů. Vysvětlení této závislosti je navrženo v sekci V. Pro získání relevantních dat bude třeba měření upravit a zopakovat. Výsledky těchto nových měření by měly být součástí prezentace.

Ačkoli tedy výstupem práce zatím nejsou zcela relevantní výsledky, lze na základě naměřených dat odhadovat, že základní metody pasivní redundance opravdu snižují odolnost proti diferenciální příkonové analýze, i když zřejmě méně, než by se zdálo na základě dosavadních měření.

VII. Budoucí práce

Pro získání dalších výsledků bude potřeba optimalizovat naši aplikaci pro DPA. Pak bude možné získat výsledky nejen pro NMR, ale i pro jeden modul AES, což je pro další posun nezbytné. S vylepšenou aplikací se pak budeme moci zaměřit na další návrhové metody pro odolnost proti poruchám.

DPA není jediným útokem postranními kanály, tudíž by bylo vhodné se věnovat i ostatním, například diferenciální chybové analýze (Differential Fault Analysis - DFA). Díky rozšířenému okruhu uvažovaných útoků získáme relevantnější informace o tom, které spolehlivostní návrhové metody mají jaký vliv na odolnost proti útokům.

Další možností pokračování výzkumu je zkoumat vliv návrhových metod pro odolnost proti útokům na odolnost proti poruchám. Tím získáme náhled z opačné strany a budeme moci určit, které metody jsou vhodné pro použití ve spolehlivých systémech.

Finálním cílem celého výzkumu je využít data získaná v předchozích krocích a vytvořit metody číslicového návrhu zvyšující odolnost proti útokům a odolnost proti poruchám současně, ať už se bude jednat o úpravu některých současných metod, či zcela nové návrhové metody.

Poděkování

Tento výzkum byl podporován grantem GA16-05179S Grantové agentury České Republiky, „Výzkum vztahů a společných vlastností spolehlivých a bezpečných architektur založených na programovatelných obvodech“ (2016-2018) a projektem SGS16/042/OHK3/1T/18 ČVUT.

Literatura

[1] D. Pradhan, *Fault-tolerant computer system design*. Upper Saddle River, N.J: Prentice Hall PTR, 1996.

[2] I. Koren, *Fault-tolerant systems*. Amsterdam Boston: Elsevier/Morgan Kaufmann, 2007.

[3] K. Tiri and I. Verbauwhede, “A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation,” in *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*, vol. 1, Feb 2004, pp. 246–251 Vol.1.

[4] S. Yang, W. Wolf, N. Vijaykrishnan, D. N. Serpanos, and Y. Xie, “Power attack resistant cryptosystem design: a dynamic voltage and frequency switching approach,” in *Proceedings of the conference on Design, Automation and Test in Europe-Volume 3*. IEEE Computer Society, 2005, pp. 64–69.

[5] F. I. P. S. P. F. 197), “Advanced Encryption Standard (AES),” 2001.

[6] C. Paar, *Implementation of Cryptographic Schemes 1*, Ruhr University Bochum, 2015.

[7] L. T. McDaniel III, “An investigation of differential power analysis attacks on FPGA-based encryption systems,” Ph.D. dissertation, Virginia Polytechnic Institute and State University, 2003.

[8] V. Fischer, F. Bernard, and P. Haddad, “An open-source multi-FPGA modular system for fair benchmarking of true random number generators,” in *Field Programmable Logic and Applications (FPL), 2013 23rd International Conference on*. IEEE, 2013, pp. 1–4.

Evoluční hardware v síťových aplikacích

David Grochol
2. ročník, prezenční studium
Školitel: Lukáš Sekanina

Vysoké učení technické v Brně
Božetěchova 2, 612 66, Brno, ČR
igrochol@fit.vutbr.cz

***Abstrakt*—Tento článek se zabývá využitím evolučních algoritmů v oblasti síťových aplikací. V rámci článku jsou představeny cíle disertační práce a dosavadní práce představující využití kartézského a lineárního genetického programování ve dvou případových studiích. První studie se zabývá klasifikací aplikačních protokolů v FPGA a druhá se věnuje návrhu speciální hašovací funkce pro síťové aplikace.**

***Klíčová slova*—síťová aplikace, lineární genetické programování, kartézské genetické programování, SDM**

I. Úvod

Počítačové sítě jsou v poslední letech využívány stále větším počtem zařízení a uživatelů. S tímto fenoménem roste množství dat, která musí být přenášená pomocí sítě. S rostoucím množstvím dat se musí přizpůsobovat technologie umožňující přenos těchto dat. Kromě přenosových technologií je potřeba přizpůsobovat rychlost i dalších aplikací starajících se o provoz sítě, sledování stavu síťových prvků, monitorování provozu a systémů zajišťujících bezpečnost.

U dnes používaných vysokorychlostních sítí s propustností až 100 Gb/s je stále častější hardwarová akcelerace síťových aplikací. Toto řešení s sebou nese i své problémy. Aby bylo řešení pomocí hardware robustní, byl by potřeba velmi specifický hardware, který se vyznačuje vysokou cenou. Proto je vhodnější použít obecnější ale konfigurovatelné hardwarové komponenty, které jsou řízeny pomocí software. Návrh hardwarových komponent je možný pomocí konvenčních metod, které vyžadují perfektní znalost problematiky, nebo využitím technik evolučního návrhu, které nevyžadují tak perfektní znalost problematiky, ale dovolují v některých případech dosáhnout lepších parametrů systémů.

Ve své práci se zabývám využitím evolučních technik v návrhu a optimalizaci síťových aplikací. Tyto síťové aplikace, určené pro sítě s rychlostí až 100 Gb/s, je potřeba navrhovat příp. optimalizovat zejména z pohledu zpoždění. Aby bylo možné využít evolučního algoritmů (EA), je potřeba vhodně upravit konstrukci EA s ohledem na použití v rámci síťových aplikací. V některých případech je potřeba také vylepšit efektivitu fungování EA, jak z pohledu evoluce, tak z pohledu rychlosti běhu samotného EA.

V kapitole II seznamuji čtenáře stručně s variantami genetického programování a pojmem síťová aplikace z pohledu této práce. V kapitole III popisují cíle práce a dosavadní výzkumnou činnost. Závěrečná kapitola IV shrnuje nejdůležitější body textu.

II. Evoluční algoritmy a síťové aplikace

Genetické programování (GP) [1], [2] umožňuje automaticky navrhovat programy. V práci využívám dvě varianty genetického programování a to konkrétně kartézské GP (CGP) [3], [4], které využívá pro reprezentaci problému acyklické orientované grafy. CGP používá relativně malé populace jedinců a využívá pouze mutaci pro vytváření nových jedinců. Jednou z aplikací CGP je optimalizace obvodů pro FPGA (Field Programmable Gate Array). Další variantou GP je lineární GP (LGP) [5]. V LGP jsou kandidátní programy reprezentovány posloupností instrukcí. Pomocí LGP jsou obvykle navrhovány relativně krátké programy, které je ale možné přesně doladit pro potřeby konkrétní aplikace.

A. Síťové aplikace

Síťové aplikace [6], [7] můžeme mimo jiné chápat jako komplexní systémy, které pracují v síti. Jejich úkolem je poskytovat služby uživatelům a zajišťovat bezpečnost sítí, monitorování provozu apod.

Většina síťových aplikací pracuje nad nejvyšší vrstvou TCP/IP modelu, tedy na aplikační vrstvě. Síťové aplikace (aplikační protokoly) můžeme rozdělit do dvou skupin. První skupinu tvoří uživatelské protokoly, které poskytují služby přímo uživateli (např. HTTP, SMTP, SSH, FTP) a druhou systémové protokoly, které zajišťují síťové funkce (např. SNMP, DNS).

Prostředky pro bezpečnost a monitorování provozu spolu úzce souvisí. Při monitorování sítě je kontrolován provoz na síti, a pokud je detekována anomálie, může být provoz filtrován. Filtrování síťového provozu je jedním z mechanismů posilující bezpečnost. S rostoucí rychlostí počítačových sítí (dnes až 100 Gb/s) jsou potřeba stále výkonnější aplikace/zařízení pro monitorování a bezpečnost sítí. Pro určité aplikace (zejména z pohledu bezpečnosti) je potřeba provádět analýzy v reálném čase.

1) Evoluční algoritmy v počítačových sítích: Evoluční algoritmy mohou být v počítačových sítích využity pro návrh a optimalizaci prakticky na všech úrovních, počínaje samotným návrhem topologie sítě [8]. EA byly úspěšně použity při směrování protokolů v síti na základě více QoS parametrů

[9]. Dalším příkladem je návrh speciální hašovací funkce. V rámci práce [10] byl navržen způsob, jak navrhovat speciální hašovací funkce pro hašování IP adres v hardware.

III. Cíle disertační práce a dosažené výsledky

EA v sítích již využity byly, ale zejména pro optimalizaci nebo návrh relativně jednoduchých komponent. Otázkou je, zda je evoluční přístup vhodný i pro optimalizaci a návrh složitých obvodových komponent moderních síťových zařízení, jako je například systém SDM (Software Defined Monitoring) [11].

Cíle disertační práce:

1) Seznámit se s vybranými síťovými aplikacemi.
2) Prostudovat stávající evolučními algoritmy.
3) Navrhnout a implementovat evoluční algoritmy pro vybrané síťové aplikace.
4) Vylepšit stávající síťové aplikace využitím evolučních algoritmů zaměřených na optimalizaci zpoždění.
5) Ověřit a experimentálně vyhodnotit vylepšení síťových aplikací na reálných síťových datech.

Ve své práci bych se rád zaměřil na systém softwarově řízeného monitorování (SDM). Jedná se o systém hardwarové akcelerace monitorovacích a bezpečnostních aplikací. Základní princip je postavený na softwarově řízeném předzpracování síťových dat v hardware. Systém se skládá z mnoha částí, které jsou navrženy konvenčním způsobem. Díky dělení zátěže mezi software a hardware může tento systém pracovat v sítích s propustností až 100 Gb/s. Cílem disertační práce bude ukázat, že optimalizací stávajících a návrhem nových aplikací pomocí evolučních algoritmů je možné zvyšovat efektivitu systému SDM. Dalším cílem bude vytvořit metodiku pro optimalizaci zpoždění struktur vytvářených pomocí GP, kde je možné problém řešit jako jednokriteriální, vícekriteriální. Následuje popis dvou případových studií.

A. Rychlá klasifikace aplikačních protokolů v FPGA optimalizovaná pomocí CGP

Klasifikace aplikačních protokolů je důležitá součást systému SDM. Na základě identifikovaného aplikačního protokolu mohou být nastavena pravidla pro další zpracování, případně odfiltrování dat. Důležitým požadavkem na klasifikátor je velmi nízká odezva, pro 100 Gb/s linky je nutné dosáhnout zpoždění menší než 7 ns. Proto je klasifikátor umístěn v hardwarové části systému SDM.

V rámci mé práce byl navržen klasifikátor aplikačních protokolů implementovaný v FPGA a optimalizovaný pomocí CGP, klasifikující protokoly HTTP, SMTP, SSH a později SIP na základě aplikačních dat. Pro určení přesnosti a úplnosti klasifikace byla použita reálná data zachycena na propojeních sítí CESNET a ACONET (CESACO) a CESNET a PIONIER (CESPIO), navíc byla použita i speciální sada pro protokol SIP a SSH (DATA SIP). Pro tuto práci jsou zajímavé pouze pakety protokolu TCP a UDP. Datová sada byla anotována pomocí L7 filtru [12].

Navržený klasifikátor identifikuje aplikační protokol na základě několika bajtů aplikačních dat prvního datového paketu toku. Tyto bajty budou v dalším textu nazývány "vzor". Například pro protokol HTTP a metodu GET je nejdelší vzor prvního paketu: "GET /". Nejprve byl navržen co možná nejpřesnější klasifikátor (CL-acc) obsahující co možná nejkomplexnější vzory protokolů. Další klasifikátor CL-cmp je kompromisem, kde je délka vzorů zkrácena na 4 znaky (např. "GET "") a poslední verze klasifikátoru je snahou o vytvoření co možná nejmenší implementace (CL-lat), kde jsou pouze znaky, které se vyskytují alespoň dvakrát ve vzorech na stejné pozici a každý vzor musí obsahovat alespoň 3 znaky (např. "*ET /"). Všechny vzory jsou prezentovány v [13].

V FPGA je klasifikátor připojen k sběrnici o šířce 512 bitů (součást SDM), přes kterou jsou posílána všechna data ze sítě. Každý rámec začíná hlavičkami nejnižších úrovní jako Ethernet, IPv4 nebo IPv6 a TCP nebo UDP. Důsledkem je, že aplikační data mohou začínat na různých pozicích, konkrétně s ofsetem 2 bajty z pozice 0 nebo 2 + 4k bajtů, kde $k = 1, ..., 16$.

První úroveň klasifikátoru tvoří kodéry s osmi vstupy kódující znaky vzorů aplikačních protokolů (celkem je 64 kodérů). Jelikož aplikační data mohou začínat s ofsetem 4, jsou navrženy 4 typy kodérů (c1, c2, c3, c4). Kodéry na výstupu produkují kód 2 z N. Rozdělení znaků mezi kodéry je uvedeno v [13].

Druhou úrovní klasifikátoru představují komparátory. Komparátor porovnává výstupy kodérů, a pokud nalezne požadovaný vzor, je na jeho výstupu nastaven odpovídající bit identifikující daný aplikační protokol (0001 - HTTP, 0010 - SMTP, 0100 - SSH, 1000 - SIP, 0000 - neznámý). Velikost vstupu komparátoru odpovídá délce nejdelšího vzoru. Třetí úroveň tvoří hradlo OR, které spojuje výstupy komparátorů do jediného čtyřbitového výstupu klasifikátoru.

Implementace probíhala v následujících krocích: Klasifikátor byl implementován konvenčním způsobem a následně byly optimalizovány vybrané komponenty (kodéry). Poté byl klasifikátor implementován s využitím optimalizovaných kodérů a na závěr byla provedena verifikace kvality klasifikace.

Všechny tři klasifikátory (CL-acc, CL-cmp, CL-lat) byly popsány behaviorálně pomocí VHDL a syntetizovány do FPGA systému SDM. Optimalizační kritérium syntézy bylo nastaveno na zpoždění obvodu.

Pro optimalizaci všech kodérů každého klasifikátoru byla použita standardní verze CGP pracující s dvou-vstupovými hradly. Použití hradel s šesti vstupy se nezdá být pro návrh vhodné z důvodu velkého prohledávaného prostoru. Detailní nastavení CGP je popsáno v [13].

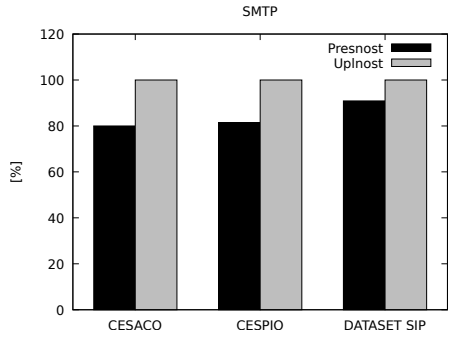
Tabulka I
Výsledky syntézy pro Xilinx Virtex-7 XC7VH580T FPGA.

Klasifikátor	LUTs	Flip Flop	Zpoždění [ns]
CL-acc	2352	0	6.410
CL-acc+CGP	1909	0	6.113
CL-cmp	1549	0	6.093
CL-cmp+CGP	1073	0	5.604
CL-lat	1625	0	5.943
CL-lat+CGP	1217	0	5.139
Yamagaki/Clark	10431	2326	77.504 (16 x 4.844)
AMTH	10547	2190	71.536 (16 x 4.671)

Získané výsledky návrhu jsou pro všechny kodéry všech klasifikátorů publikovány v [13]. Výsledky potvrdily předpoklad, že optimalizované kodéry klasifikátoru CL-lat jsou menší než kodéry CL-cmp a kodéry CL-cmp jsou menší než kodéry CL-acc. Tyto kodéry byly syntetizovány do FPGA. V tabulce I vidíme počet LUT, Flip-Flop a zpoždění pro všechny syntetizované obvody (optimalizované obvody pomocí CGP jsou označeny jako "+CGP"). Pro srovnání jsou přidány dvě implementace klasifikátoru vytvořené konvenčně pomocí regulárních výrazů. Tyto klasifikátory obsahují 16 paralelních jednotek, aby bylo dosaženo propustnosti 100 Gb/s.

Zjednodušením klasifikátoru CL-acc na klasifikátory CL-cmp a CL-lat a s využitím optimalizace pomocí CGP bylo uspořeno 48,2 % prostoru (LUT) a zpoždění zkráceno o 19,8 % oproti implementaci CL-acc.

Kvalita klasifikace byla ověřena offline s využitím softwarového modelu klasifikátoru na všech třech datových sadách. Jako metriky kvality byly zvoleny přesnost a úplnost.



Obrázek 1. Přesnost a úplnost v procentech pro protokol SMTP na všech třech datových sadách s použitím klasifikátoru CL-lat.

Přesnost určuje počet toků správně klasifikovaných. Na obrázku 1 jsou výsledky pro protokol SMTP pro nejméně přesný klasifikátor CL-lat. Nejméně přesná je klasifikace pro protokol SMTP, který je identifikován pomocí relativně krátkého a obecného vzoru. Pro systém SDM je ovšem důležitější metrika úplnosti, která určuje počet identifikovaných toků. Pro systém SDM je z pohledu bezpečnosti důležité zachytit všechna hledaná data (protokoly). Pokud jsou nějaká data zachycena navíc, mohou být později odstraněna. Úplnost 100 % znamená, že žádná data nebyla ztracena.

Návrh, implementace a optimalizace základní funkční jednotky klasifikátoru CL-acc (9 kodérů a komparátor) pro protokoly HTTP, SMTP a SSH a ohodnocení kvality tohoto klasifikátoru bylo publikováno na mezinárodní konferenci *European Conference on the Applications of Evolutionary Computation* [14]. Výše popsaná rozšířená verze byla publikována v impaktovaném časopise *Applied Soft Computing* (IF = 2,857) [13].

B. Evoluce hašovací funkce pro hašování síťových toků

Hašovací funkce se v síťových aplikacích používají k mnoha účelům, například, jak bylo popsáno výše, k hašování IP adres. Další možností je hašování síťových toků, tedy pětice zdrojová a cílová adresa (2 x 32b), zdrojový a cílový port (2 x 16b) a transportní protokol (8b), tedy 104 bitů. Pomocí této pětice je možno jednoznačně v síťových aplikacích identifikovat každý tok. V systému SDM se hašování síťových toků využívá zejména v softwarové části, pro ukládání informací o jednotlivých tocích, případně informací o jejich zpracování pro různé uživatelsky specifické aplikace. V systému SDM je použita hašovací tabulka s lineárním seznamem pro řešení kolizí [15]. Tento typ tabulky klade speciální požadavky na hašovací funkci. Pro tabulku je nejvhodnější hašovací funkce, která produkuje co nejméně kolizí. Ovšem při hašování síťových toků vzniká poměrně velké množství kolizí, proto je vhodnější hašovací funkce, která neprodukuje dlouhé seznamy, ale produkuje větší množství krátkých seznamů. Pokud budou seznamy u jednotlivých záznamů krátké, bude jejich prohledávání méně časově náročné.

Evolučně navržená hašovací funkce má být určena primárně pro software. Z toho důvodu se jeví jako vhodné použít LGP pro tento návrh. První experimenty s návrhem ukázaly, že navržené hašovací funkce se stejnou množinou funkcí, jaké obsahují konvenční hašovací funkce (prvočísla, logické operace, sčítaní a násobení) a fitness funkcí zaměřenou pouze na minimalizaci kolizí, nepřinesly výrazné vylepšení, jak z pohledu kvality hašování (počet kolizí), tak rychlosti výpočtu.

Pro zjednodušení návrhu hašovací funkce byla dimenze vstupního vektoru snížena z 5 na 3 slova velikosti 32 bitů. Zdrojová a cílová adresa je zachována v původní podobě, ale ze zdrojového portu (zp), cílového portu (cp) a transportního protokolu (tp) je vytvořeno jedno 32 bitové slovo následujícím způsobem:

$$((zp << 16) \vee cp) \oplus tp.$$

Reálná data obsahují zejména dva transportní protokoly (TCP a UDP), takže snížení dimenze nezpůsobí výraznou ztrátu informace. Fitness funkce je založena na počtu kolizí, ale zohledňuje i délky seznamu. Nechť K_i je počet klíčů mapovaných do slotu i v hašovací tabulce h. Pak je fitness funkce $f(h)$ počítána jako:

$$f(h) = \sum_{i=1}^{s} g_i, \text{ kde} \quad (1)$$

$$g_i = \begin{cases} 0 & \text{if } K_i \leq 1 \\ \sum_{j=2}^{K_i} j^2 & \text{if } K_i \geq 2 \end{cases} \quad (2)$$

a s je počet slotů. Tato fitness funkce penalizuje kandidátní funkce, které produkují dlouhé seznamy v hašovací tabulce s lineárním seznamem. Menší hodnota fitness znamená lepší řešení.

Délka kandidátního programu byla omezena na maximálně 12 instrukcí, bylo použito osm 32 bitových registrů. Sada instrukcí obsahuje rotaci vpravo, XOR a sčítání. Populace měla velikost 200 jedinců a maximálně bylo provedeno 1000 generací. Na obrázku 2 je hašovací funkce, která reprezentuje nejlepší nalezené řešení. Dále byly pro srovnání vybrány další evolučně navržené hašovací funkce. LGPhash2 je vybrána jako velice jednoduchá funkce, LGPhashMult je navržena s instrukční sadou která navíc obsahuje násobení a LGPhash20inst

```
unsigned int LGPHash1 (unsigned int * input ){
  r[0] = input[0]
  r[1] = input[1]
  r[2] = input[2]

  r[1] = r[1] + r[2]
  r[2] = r[1] + r[2]
  r[4] = r[0] + r[2]
  r[0] = r[1] + r[4]
  r[3] = 0x5BE0CD19
  r[2] = rotr(r[3], r[4])
  r[0] = r[0] + r[2]
  r[0] = 0xA54FF53A + r[0]
  return r0 ⊕ (r0 >> 16)
}
```

Obrázek 2. Evolučně navržená hašovací funkce LGPHash1.

je omezena na maximálně 20 instrukcí. Další nastavení LGP a navržené hašovací funkce jsou publikovány v [16].

V rámci experimentů byly všechny hašovací funkce implementovány v jazyce C a všechna měření probíhala na procesoru Intel XEON E5-2630. Byly vytvořeny tři datové sady obsahující 20000 (DataSet1, použitá jako trénovací sada), 50000 (DataSet2) a 100000 (DataSet3) trénovacích vektorů. Pro srovnání byly vybrány obecné konvenční hašovací funkce (DJBHash, DEKHash, FVNHash, One At Time, lookup3, Murmur2, Murmur3, CityHash), speciální hašovací funkce pro hašování síťových toků (XORhash) a obecné evolučně navržené hašovací funkce (GPhash a EFhash). V tabulce II jsou výsledky měření doby výpočtu hašovací funkce pro každý vektor trénovací sady (průměrná hodnota 20 nezávislých měření). Navržené hašovací funkce LGPhash1 a LGPhash2 jsou nejrychlejší. Dále byl proveden test na počet kolizí, kde se ukázalo, že počet kolizí je srovnatelný s ostatními hašovacími funkcemi. Podrobné výsledky jsou publikovány v [16].

Použitím evolučních technik pro návrh specializované hašovací funkce jsme dosáhli u funkce LGPhash1 zkrácení výpočetního času o 10.4%, 4.2% a 3.0% pro DataSet 1, 2 a 3 oproti nejlepší konvenčně navržené specializované XORhash. Navíc LGPhash1 dosahuje zlepšení výpočetního času 48.5%, 31.4% a 26.9% oproti hašovací funkce murmur3, která je typicky používána v SDM. Pro návrh hašovacích funkcí byla využita vlastní paralelní varianta LGP prezentovaná v [17].

Tabulka II
PRŮMĚRNÝ ČAS PRO VLOŽENÍ DATOVÉ SADY DO HAŠOVACÍ TABULKY.

Hašovací funkce	Čas $[ms]$ DataSet1	DataSet2	DataSet3
DJBHash	1.783	5.036	13.254
DEKHash	1.592	4.591	12.199
FVNHash	1.678	4.647	12.373
One At Time	2.365	6.269	15.763
lookup3	1.275	3.736	9.931
Murmur2	1.314	3.820	10.153
Murmur3	1.590	4.434	11.568
CityHash	3.089	7.883	19.237
XORHash	0.913	3.174	8.708
GPHash	1.936	6.229	15.813
EFHash	2.323	16.282	56.921
LGPhash1	*0.818*	*3.039*	*8.446*
LGPhash2	***0.756***	***2.852***	***8.057***
LGPhashMult	0.912	3.349	9.096
LGPhash20inst	0.916	3.242	8.954

IV. ZÁVĚR

Cílem disertační práce je aplikování evolučních algoritmů na vybrané síťové aplikace, nebo jejich netriviální části a dosáhnout tím jejich vylepšení a to z pohledu funkčnosti, časové náročnosti nebo příkonu. První výsledky návrhu a optimalizace aplikací v rámci systému SDM již byly publikovány. Otevřenou otázkou je, jak pracovat se zpožděním v GP.

Plán dalších prací zahrnuje využití multikriteriální optimalizaci v LGP i CGP, návrh programů pomocí LGP s ohledem na možnosti vektorizace instrukcí a zohlednění velikosti obvodu (počet LUT) v FPGA na základě predikce během evoluce. Dále tyto přístupy budou porovnány s dalšími optimalizačními metodami např. s ABC.

PODĚKOVÁNÍ

Tato práce byla podporována projektem Vysokého učení technického v Brně FIT-S-14-2297.

LITERATURA

[1] J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*. MIT press, 1992, vol. 1.
[2] W. Banzhaf and P. Nordin et al., *Genetic programming: an introduction*. Morgan Kaufmann San Francisco, 1998, vol. 1.
[3] J. F. Miller and P. Thomson, "Cartesian genetic programming," in *Genetic Programming*. Springer, 2000, pp. 121–132.
[4] J. F. Miller, *Cartesian genetic programming*. Springer, 2011.
[5] M. F. Brameier and W. Banzhaf, *Linear genetic programming*. Springer Science & Business Media, 2007.
[6] A. S. Tanenbaum, "Computer networks, 4-th edition," 2003.
[7] F. Halsall, *Computer Networking and the Internet, 5/e*. Pearson Education India, 2006.
[8] R. M. Morais and C. Pavan et al., "Genetic algorithm for the topological design of survivable optical transport networks," *Journal of Optical Communications and Networking*, vol. 3, no. 1, pp. 17–26, 2011.
[9] L. Barolli and A. Koyama et al., "A genetic algorithm based routing method using two qos parameters," in *Proceedings. 13th International Workshop on Database and Expert Systems Applications.* IEEE, 2002, pp. 7–11.
[10] R. Dobai and J. Kořenek, "Evolution of non-cryptographic hash function pairs for fpga-based network applications," in *2015 IEEE Symposium Series on Computational Intelligence*. IEEE, 2015, pp. 1214–1219.
[11] L. Kekely, V. Pus, and J. Korenek, "Software defined monitoring of application protocols," in *INFOCOM, 2014 Proceedings IEEE*, 2014, pp. 1725–1733.
[12] E. Sommer and M. Strait, "Application Layer Packet Classifier for Linux," Jan. 2009. [Online]. Available: http://l7-filter.sourceforge.net/
[13] D. Grochol and L. Sekanina et al., "Evolutionary circuit design for fast fpga-based classification of network application protocols," *Applied Soft Computing*, vol. 38, pp. 933–941, 2016.
[14] D. Grochol and L. Sekanina et al., "A fast fpga-based classification of application protocols optimized using cartesian gp," in *18th European Conference, EvoApplications*. Springer, 2015, pp. 67–78.
[15] W. D. Maurer and T. G. Lewis, "Hash table methods," *ACM Computing Surveys (CSUR)*, vol. 7, no. 1, pp. 5–19, 1975.
[16] D. Grochol and L. Sekanina, "Evolutionary design of fast high-quality hash functions for network applications," 2016, GECCO, Denver.
[17] D. Grochol and L. Sekanina, "Comparison of parallel linear genetic programming implementations," in *Mendel 2016: Recent Advances in Soft Computing*. Springer, 2016.

Evoluční snižování příkonu: Od obvodů na úrovni tranzistorů po neuronové sítě na čipu

Vojtěch Mrázek
2. ročník, prezenční studium
Školitel: Lukáš Sekanina, specialista: Zdeněk Vašíček

Vysoké učení technické v Brně
Božetěchova 2, 612 66 Brno
imrazek@fit.vutbr.cz

***Abstrakt*—Snižování příkonu integrovaných obvodů je v dnešní době, například u mobilních aplikací, velmi důležité. Práce se zabývá využitím evolučních algoritmů pro optimalizaci příkonu kombinačních obvodů. Zaměřuje se zejména na aplikaci této optimalizace v reálných systémech. Na čtyřech vybraných aplikacích ukazuje možnosti snížení příkonu pomocí evolučního algoritmu. Byly navrženy nové prvky vhodné do technologické knihovny definující strukturu jednotlivých hradel na úrovni tranzistorů, byla snížena spotřeba mediánových filtrů, byly představeny nové metody aproximačního řazení a zefektivněna energetická náročnost klasifikace pomocí neuronových sítí. Všechny tyto aplikace spojuje stejná metoda návrhu využívající evoluční přístup.**

***Klíčová slova*—Evoluční návrh, tranzistorová úroveň, aproximace, neuronové sítě, řazení**

I. Úvod

Evoluční návrh obvodů je metoda, která používá biologií inspirované prohledávací algoritmy pro syntézu a optimalizaci elektronických obvodů. Tento návrh umožnil získat mnoho zajímavých výsledků, ovšem má několik základních nedostatků, které je nutné řešit. Prvním problémem je, že navržené obvody často nereflektovaly požadavky HW komunity. Jedním z důvodů je fakt, že optimalizace většinou probíhala s ohledem na počet výpočetních prvků, což nemusí přímo odpovídat skutečným vlastnostem obvodů z důvodu různé složitosti jednotlivých prvků, výstupní kapacity, přepínací aktivity a podobně. Druhou nevýhodou je nemožnost dobré škálovatelnosti evolučního návrhu. Problém škálovatelnosti je obtížně řešitelný, ale existují techniky (například využití pokročilých verifikačních technik či dekompozice), které umožňují evolučně optimalizovat i složitější obvody.

Proto je cílem této práce navrhnout metodiku, která umožní upravit evoluční návrh obvodů tak, aby navržené obvody byly použitelné v reálných případech. Jako vhodné parametry pro optimalizaci se jeví plocha, zpoždění a příkon. V dnešní době vysokého stupně integrace už vliv plochy není tak důležitý oproti požadavkům na příkon. Díky rozvoji přenosných zařízení, jako jsou mobilní technologie, wearable elektronika a podobně, je právě spotřeba elektronických zařízení klíčovou otázkou.

V práci je rozšířen evoluční návrh, zejména kartézské genetické programování [1], o možnost navrhovat obvody s nízkým příkonem. Práce ukazuje, že redukce spotřeby elektrické energie je možné dosáhnout na více úrovních abstrakce (popisu) obvodů. Pohybuje se na strukturální doméně, jako jsou tranzistory a hradla, a využívá snížení příkonu s využitím znalosti funkce na doméně behaviorální.

II. Cíle disertační práce

Je známo, že existuje řada konvenčních přístupů pro redukci spotřeby kombinačních obvodů. Příkon je možné snižovat jak na úrovni technologické (volba vhodné technologie a knihovny [2]), tak na úrovni zapojení jednotlivých částí obvodů — tranzistorů [3] i hradel. Další úsporu můžeme dosáhnout využitím znalosti aplikace těchto obvodů a redukcí obvyklé funkčnosti. Typickým příkladem je aproximační počítání, kdy prvky pracují nepřesně za předpokladu, že výsledná aplikace je částečně imunní vůči chybám *(error resilient)* [4]. Abychom mohli příkon za pomoci navrženého algoritmu snižovat, musíme ho být schopni správně odhadnout, abychom určili vhodnost provedené změny obvodu.

Mimo to se také ukazuje, že evoluční algoritmy, zejména kartézské genetické programování, umožňují efektivně navrhovat kombinační obvody na úrovni tranzistorů [5], [6], [7], hradel [8], [9] či aproximačních obvodů [10]. Ovšem tyto metody nebyly kombinovány. Z dosavadních znalostí můžeme formulovat hypotézu disertační práce: *Za pomoci evolučního algoritmu upraveného pro konkrétní aplikaci je možné snížit celkový příkon kombinačních obvodů popsaných na různých úrovních oproti řešením poskytovaným konvenčním přístupem.* Byly stanoveny následující cíle pro disertační práci:

1) Vytvoření výpočetně nenáročné metody pro odhad příkonu číslicového obvodu.
2) Návrh evolučního algoritmu využívajícího odhad příkonu pro optimalizaci celkové spotřeby.
3) Využití akcelerace pro urychlení evolučního návrhu.
4) Experimentální ověření pro různé aplikace a úrovně popisu.
5) Porovnání navrženého přístupu s konvenčními metodami.

Demonstrační aplikace na různých úrovních popisu budou následující:

- Optimalizace obvodů technologické knihovny *(úroveň tranzistorů)*.
- Zefektivnění klasifikace pomocí neuronových sítí *(úroveň hradel a funkce)*.
- Snížení spotřeby mediánových filtrů v oblasti obrazových filtrů a filtrů signálů *(úroveň funkce)*.
- Aplikace přibližných řadicích sítí *(úroveň funkce)*.

III. Navržené metody a získané výsledky

Několik případových studií úzce souvisejících s tématem jsem již provedl a publikoval, nebo jsou příslušné odborné články připravovány. Tato kapitola poskytuje pouze základní přehled dosažených výsledků, podrobnější informace nalezneme v odkazovaných publikacích. Mimo níže zmíněné publikace byl vytvořen článek o multikriteriálním návrhu aproximačních aritmetických obvodů [11].

A. Optimalizace obvodů popsaných na úrovni tranzistorů

Abychom byli schopni navrhovat obvody na úrovni tranzistorů, je nutné upravit evoluční algoritmus, který je obvykle stavěn pro návrh obvodů na úrovni hradel. Úprava spočívá v zavedení nového kódování obvodů, změně výpočtu fitness funkce a zavedení vhodného odhadu spotřeby. V rámci porovnání s klasickými přístupy budou navržené obvody porovnány vůči technice přepisu z hradlové úrovně.

V článku [12] byla představena nová reprezentace obvodů popsaných na úrovni tranzistorů. Díky použití vlastního diskrétního simulátoru, který pracoval s vícehodnotovou logikou, se podařilo evolučně navrhnout obvody o velikosti až 30 tranzistorů. To byl poměrně velký pokrok oproti předchozím výzkumům, kde se cílilo na obvody do 10 tranzistorů [13], [7].

Abychom urychlili celkový evoluční návrh obvodů, byl vytvořen akcelerátor využívající čip Xilinx Zynq. v hardware se akceleruje simulace obvodů pro 2^n vstupních kombinací, kde n je počet vstupních signálů. Návrh se podařilo zrychlit až $4.7\times$ oproti procesoru Intel Xeon. Samotná akcelerovaná simulace obvodů je dokonce $25\times$ rychlejší než stejný diskrétní algoritmus na procesoru. Výsledky byly publikovány na konferenci ICES [14].

Předcházející práce využívající evoluci se většinou zabývaly optimalizací na počet tranzistorů. Je zřejmé, že na spotřebu mají vliv i další parametry, zejména přepínací aktivita jednotlivých tranzistorů. Proto jsem navrhl metodu, která odhaduje příkon obvodu popsaného na úrovni tranzistorů a je inspirovaná metodami používanými pro odhad spotřeby na úrovni hradel. Využívá se výsledků z diskrétní simulace, kde ke každému tranzistoru je možné určit četnost výskytu kombinace nastavení vstupu *source* a vstupu *gate*. Pro aktivní stavy se pak vypočítá pravděpodobnost přepnutí ze stavu a do stavu b (kde stav je dvojice hodnoty *source* a *gate*). V simulátoru SPICE byla určena spotřeba pro každou možnou dvojici přepnutí, kde pak součtem součinů spotřeby přechodů mezi stavy a jejich pravděpodobností dojde k odhadu spotřeby. Ukázalo se, že tato metoda přináší zajímavé výsledky, kdy jsme schopni snížit příkon o 4 – 12 % s tím, že zpoždění obvodu zůstane přibližně zachováno, nebo se zlepší. Tímto způsobem byly optimalizovány obvody obsahující až 300 tranzistorů. Navržená metoda optimalizace byla prezentována na konferenci EUC [15].

B. Optimalizace a aproximace mediánových filtrů

Existuje několik typů filtrů, které jsou postaveny na řazení prvků pomocí řadicí sítě. Typickým případem je mediánový filtr. Pro tyto aplikace dokáží evoluční algoritmy najít kvalitní implementaci pomocí elementárních komponent [16]. Ovšem s rostoucím počtem vstupů tyto filtry narůstají do velkých rozměrů a jejich energetická náročnost roste. Proto bylo navázáno na výsledky prezentované v [10] a byl navržen efektivní algoritmus pro optimalizaci filtrů na bázi řadicích sítí i za cenu rozumného snížení přesnosti. Vzhledem k tomu, že optimální řadicí sítě jsou poměrně dobře prozkoumané [17], [18], zaměřím se spíše na optimalizaci filtrů vzniklých z optimálních řadicích sítí [19].

1) Metrika vyhodnocení přesnosti: Stejně jako u všech přibližných výpočtů je nutné definovat odchylky od správného řešení. Klasický přístup pro výpočet chyby používaný v předcházejících výzkumech byl založen na výpočtu sumy odchylek pro zadaná vstupní data. Pokud bychom pro mediánový filtr s 9 vstupy (na 8 bitech) počítali chybu přes všechny vstupy, byl by celkový počet kombinací $2^{9^8} = 2^{72} \sim 4.7 \cdot 10^{21}$. Tento počet vstupních kombinací je neřešitelný, proto se používá pouze podmnožina vstupních kombinací. Navíc výsledné ohodnocení není příliš vypovídající.

Z těchto důvodů jsem navrhl novou metriku pro určení chyby nazvanou *permutační princip*, která je využitelná pro návrh mediánového filtru. Stejně tak je aplikovatelná i pro řadicí sítě. Knut dokázal, že pro ověření funkčnosti řadicí sítě nemusíme použít všechny možné kombinace, ale stačí nám pouze všechny kombinace 0 a 1 (tzv. zero-one princip) [17]. Podobně můžeme dokázat, že pro ověření funkčnosti n vstupového mediánu můžeme použít permutace množiny $S = \{0, 1, \ldots, n-1\}$. Pokud budeme sledovat výstup i-tého prvku, tak pro všechny permutace množiny S bude mít plně funkční řadicí síť výstup i. Když síť nebude plně funkční, tak rozdíl hodnoty výstupu a i udává tzv. poziční odchylku, jejíž rozložení můžeme sledovat. Sledování odchylky snižuje složitost z exponenciální na faktoriálovou. Mimo to získáme přehlednější ohodnocení, které je navíc nezávislé na výběru podmnožiny vstupních dat. Tato metrika je představena v publikacích [20], [21]. Navíc v časopise [21] *(Q2, IF=1.143)* byl představen formální důkaz o tom, že otestováním všech permutací kompletně ověříme řadicí síť.

Ukázalo se, jak je prezentováno v odeslané publikaci [22], že jsme schopni snížit složitost z $n!$ na 2^n upravením *zero-one principu*. S využitím formálního aparátu, jako je BDD (Binary Decision Diagram — binární rozhodovací stromy, řešitelné např. pomocí knihovny BuDDy), jsme schopni získat rozložení chyb ve velmi krátkém čase.

2) Přibližné mediánové filtry: Mediánový filtr je specifický druh komparátorové sítě, která má n vstupů a jeden výstup. Byly navrženy dva druhy implementace — softwarová a hardwarová. Oba druhy implementace vyžadují specifické kódování problému. Pro softwarovou implementaci se jedná o zapojení dvouvstupých bloků s jedním výstupem realizujících operaci výběru minima nebo maxima. Hardwarová implementace vychází z použití *compare&swap* bloků. Vzhledem k paralelnímu zpracování jsou jednotlivé operace naplánovány pomocí ASAP algoritmu. Výstupem plánování je počet bloků, počet registrů pro zřetězenou architekturu a její latence.

Vstupem evolučního algoritmu byl, kromě nastavení parametrů běhu, plně funkční obvod a omezení (*contraints*) požadovaného obvodu. Omezena byla maximální velikost a latence. Algoritmus pracoval ve dvou fázích. Během první fáze byl výchozí obvod zmenšován tak, aby vyhovoval požadavkům. Při druhé fázi byla evolučně optimalizována kvalita (určena pomocí permutačního či BDD principu) při splnění všech omezení.

Navržené obvody byly testovány v reálných aplikacích zpracování obrazu či signálů. Na obrázku 1 můžeme vidět kvalitu filtrace obrazů pomocí aproximačních mediánových filtrů navržených pomocí představeného algoritmu. Můžeme si všimnout, že při implementaci využívající 14 operací je jen nepatrný rozdíl oproti plně funkční implementaci, přičemž úspora energie je více jak 50 %.

(a) Poškozený obrázek (b) Mediánový filtr (c) Aproximace 22-ops

(d) Aproximace 16-ops (e) Aproximace 14-ops (f) Aproximace 10-ops

Obrázek 1: Detailní výřez obrázku, který byl poškozen 10% náhodným šumem typu pepř-sůl. Obrázek byl filtrován b) 9-vstupým mediánem a aproximačními filtry využívajícími c) 22 (73 %) operací, d) 16 (53 %), e) 14 (46 %) a f) 10 (33 %) operací.

3) Analýza spotřeby mediánových filtrů: Jak již bylo řečeno, byly optimalizovány dva druhy implementací aproximačních mediánových filtrů — hardwarová a softwarová. Hardwarové implementace byly realizovány pomocí zřetězené linky. Tyto obvody byly syntetizovány s využitím 45 nm technologie. Ukázalo se, že celková spotřeba nejvíce koreluje s váženým součtem počtu komparátorů a registrů. Tento výpočet spotřeby a plochy byl potom použit v evolučním návrhu při omezení velikosti obvodu. Výsledky budou odeslány k publikaci [22].

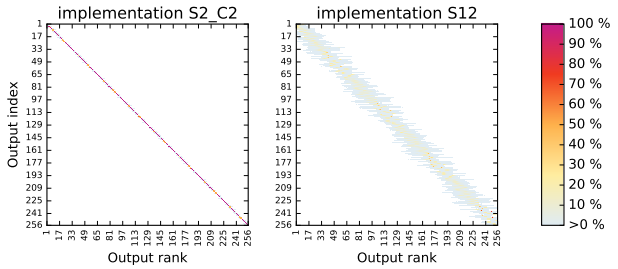
Dále byly aproximovány mediánové filtry pro vestavěné zařízení (mikroprocesory). Na čtyřech architekturách byla změřena spotřeba mediánového filtrování vstupních dat. Analýzou spotřeby reálných systémů se ukázalo, že celková spotřeba přímo závisí na počtu operací a to, že aproximací filtru ve vestavěném systému jsme schopni energii ušetřit.

C. Přibližné řadicí sítě

Vzhledem k tomu, že genetické algoritmy přinesly zajímavé výsledky v oblasti řadicích sítí, byla tato oblast vybrána jako další demonstrační aplikace. Řadicí sítě, jejichž výsledkem je určitá permutace vstupů, nachází uplatnění nejen v oblasti řazení v HW akcelerátorech, ale také v aritmetických operacích či řízení přepínacích sítí [23].

Při analýze konvenčních způsobů se ukázalo, že je nejvýhodnější stavět síť z menších prvků, podobně jak to dělá algoritmus Bitonic. Menší prvky představují řadicí a spojovací sítě. Spojovací sítě se od klasických liší v tom, že spojují dvě seřazené posloupnosti do jedné. Při evolučním návrhu byl využit permutační princip vyhodnocení kvality kandidátní řadicí sítě.

Podařilo se nalézt implementace těchto menších bloků s různou kvalitou a velikostí. Evolučně navrženými bloky byly nahrazeny různé části plně funkční sítě. Na obrázku 2 je znázorněna kvalita (četnost odchylek) pro dvě vybraná řešení s 256 vstupy. Implementace S2_C2 využívá 89 % operací s tím, že v 99 % případů se bude výsledek lišit maximálně o jednu pozici oproti seřazené posloupnosti. Druhá vybraná implementace S12 využívá 46 % operací s tím, že v 95 % případů bude odchylka do 10 pozic. Ukázalo se, že při použití stejných bloků pro 1024 vstupů je kvalita řešení (t.j. kvartily odchylek) stejná, pouze se liší redukce operací. Výsledky implementace navržených řešení v FPGA a v ASIC obvodech byly odeslány na konferenci PATMOS [24].



Obrázek 2: Histogram četnosti výskytů jednotlivých hodnot z posloupnosti $\{1 \ldots 256\}$ na jednotlivých pozicích pro dvě vybrané implementace aproximačních řadicích sítí s 256 vstupy.

D. Optimalizace násobiček pro NN

Neuronové sítě (NN) představují další velkou skupinu aplikací, jejichž celkovou spotřebu je dobré snížit. Nejnáročnější a nejčastější operací je násobení, a proto je vhodné optimalizovat zejména je. Vzhledem k tomu, že pro realizaci násobení je řada velmi efektivních algoritmů [2], je velmi malý prostor pro snížení celkové spotřeby. Pokud však budeme tolerovat chybu na výstupu, prostor se rapidně zvětšuje. Důležité je správně určit, jakou chybu si můžeme dovolit. Jako testovací případ bylo zvoleno rozpoznávání písmen MNIST databáze.

Pokud použijeme násobičky, které vykazují 5% chybu výstupu vůči rozsahu, dojde ke snížení celkové spotřeby výpočetní jednotky NN. Při porovnání schopnosti klasifikace však klesne úspěšnost z 94.16 % na 10.77 %. Podrobnější zkoumání však ukázalo, že problém je v tom, že přibližně 80 % násobení má mít výsledek 0. Pokud navrhneme násobičku tak, aby ve všech případech kromě násobení 0 měla chybu do 5 % a při násobení 0 by výsledek byl přesný, celková přesnost klasifikace stoupne až na 94.20 %.

V dalším kroku bylo využito genetické programování pro návrh 8 a 12 bitových násobiček pro různé maximální chyby s tím, že násobení nulou je vždy přesné. Ukázalo se, že například 91% redukce příkonu násobení vede k poklesu přesnosti klasifikace po přetrénování o méně než 2.8 % u SVHN datasetu, u datasetu MNIST je pokles přesnosti v řádu desetin procent. Výsledky byly přijaty na HW konferenci ICCAD'16 [25].

IV. Závěr

Z výše uvedených ukázek můžeme vidět, že metodika snižování příkonu pomocí genetického programování kombinuje znalosti nejen z oblasti evolučního návrhu, ale také i z oblasti návrhu obvodů s ohledem na příkon. Toto využití evolučních algoritmů nebylo zatím v literatuře publikované. Navíc výsledky návrhu integrovaných obvodů byly přijaty HW komunitou na specializovaných konferencích. Disertační práce by měla celkovou metodiku návrhu nízkopříkonových kombinačních obvodů pomocí nekonvenčního přístupu představit a ukázat její výhody na několika příkladech. Hlavní myšlenka je založena na kombinaci vhodného kódování problému, správného odhadu spotřeby a zejména na analýze cílové aplikace. Celkový přínosem práce je zejména prokázání toho, že evoluční návrh může přinášet zajímavé výsledky optimalizace příkonu na různých úrovních popisu.

Proto byl s ohledem na hypotézu a cíle disertační práce vytvořen následující plán. Některé cíle již byly splněny a ostatní jsou rozpracované.

- Evoluční optimalizace obvodů popsaných na úrovni tranzistorů [12], [14] a optimalizace jejich spotřeby [15].
- Návrh nové metriky pro přibližné mediánové filtry a jejich aplikace [20], [21].
- Návrh přibližných mediánových filtrů a řadicích sítí pomocí formálních metod [22] *(rozpracováno)*.
- Návrh a aplikace přibližných řadicích sítí [24].
- Snížení spotřeby neuronových sítí [25] *(rozpracováno)*.

Poděkování

Tato práce vznikla za podpory projektu Vysokého učení technického v Brně FIT-S-14-2297 Architektury paralelních a vestavěných počítačových systémů.

Reference

[1] J. F. Miller, Ed., *Cartesian genetic programming.*, ser. Natural Computing Series. Berlin: Springer, 2011.

[2] N. H. Weste and D. Harris, *CMOS VLSI design: a circuits and systemes perspective*, 3rd ed. Boston, USA: Addison-Wesley, 2005.

[3] D. Soudris, C. Piguet, and C. Goutis, *Designing CMOS Circuits for Low Power*, ser. European low-power initiative for electronic system design. Springer, 2002.

[4] H. Cho, L. Leem, and S. Mitra, "ERSA: Error resilient system architecture for probabilistic applications," *IEEE Tr. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 4, pp. 546–558, April 2012.

[5] A. Stoica, R. Zebulum, X. Guo *et al.*, "Taking evolutionary circuit design from experimentation to implementation: some useful techniques and a silicon demonstration," *IEEE Proc. Computers and Digital Techniques*, vol. 151, no. 4, pp. 295–300, July 2004.

[6] L. Zaloudek and L. Sekanina, "Transistor-level evolution of digital circuits using a special circuit simulator," in *Evolvable Systems: From Biology to Hardware*, ser. LNCS, vol. 5216. Springer Verlag, 2008.

[7] M. Trefzer, "Evolution of Transistor Circuits," Ph.D. dissertation, Ruprecht-Karls-Universität Heidelberg, 2006.

[8] Z. Vasicek, "Cartesian gp in optimization of combinational circuits with hundreds of inputs and thousands of gates," in *Genetic Programming, 18th European Conference, EuroGP 2015*, ser. LCNS 9025. Springer International Publishing, 2015, pp. 139–150.

[9] Z. Vasicek and L. Sekanina, "How to evolve complex combinational circuits from scratch?" in *2014 IEEE International Conference on Evolvable Systems Proceedings*, 2014, pp. 133–140.

[10] Z. Vasicek and L. Sekanina, "Evolutionary approach to approximate digital circuits design," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 3, pp. 432–444, 2015.

[11] R. Hrbacek, V. Mrazek, and Z. Vasicek, "Automatic design of approximate circuits by means of multi-objective evolutionary algorithms," in *Proceedings of the 11th International Conference on Design & Technology of Integrated Systems in Nanoscale Era*. IEEE Computer Society, 2016, pp. 239–244.

[12] V. Mrazek and Z. Vasicek, "Evolutionary design of transistor level digital circuits using discrete simulation," in *Genetic Programming*, ser. LNCS. Springer International Publishing, 2015, vol. 9025, pp. 66–77.

[13] J. Walker, J. Hilder, and A. Tyrrell, "Towards evolving industry-feasible intrinsic variability tolerant cmos designs," in *IEEE Congress on Evolutionary Computation*, 2009, pp. 1591–1598.

[14] V. Mrazek and Z. Vasicek, "Acceleration of transistor-level evolution using xilinx zynq platform," in *IEEE Int. Conf. on Evolvable Systems (ICES)*, Dec 2014, pp. 9–16.

[15] V. Mrazek and Z. Vasicek, "Automatic design of low-power VLSI circuits: Accurate and approximate multipliers," in *IEEE Int. Conf. on Embedded and Ubiquitous Computing (EUC)*, Oct 2015, pp. 106–113.

[16] S.-S. Choi and B. R. Moon, "Isomorphism, normalization, and a genetic algorithm for sorting network optimization." in *GECCO*, 2002, pp. 327–334.

[17] D. E. Knuth, *The Art of Computer Programming, Volume 3: (2Nd Ed.) Sorting and Searching*. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 1998.

[18] D. Bundala and J. Zavodný, "Optimal sorting networks," in *Language and Automata Theory and Applications*, ser. LNCS. Springer International Publishing, 2014, vol. 8370, pp. 236–247.

[19] N. Devillard, "Fast Median Search: An ANSI C Implementation," 1998, http://ndevilla.free.fr/median/median.pdf.

[20] V. Mrazek, Z. Vasicek, and L. Sekanina, "Evolutionary approximation of software for embedded systems: Median function," in *Proc. of the Genetic and Evolutionary Computation Conference (GECCO)*, ser. ACM, 2015, pp. 795–801.

[21] Z. Vasicek and V. Mrazek, "Trading between quality and non-functional properties of median filter in embedded systems," *Genetic Programming and Evolvable Machines*, p. 35 *(přijato)*.

[22] Z. Vasicek and V. Mrazek, "Quality-driven design of low-power approximate median architectures with provable error and suitable for high-throughput systems," in *(připraveno k odeslání)*, 2017, p. 6.

[23] K. E. Batcher, "Sorting networks and their applications," in *Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference*, ser. AFIPS '68 (Spring). New York, NY, USA: ACM, 1968, pp. 307–314.

[24] V. Mrazek and Z. Vasicek, "Automatic design of arbitrary-size approximate sorting networks with error guarantee," in *26th International Workshop on Power and Timing Modeling, Optimization and Simulation (odesláno)*, 2016, p. 8.

[25] V. Mrazek, S. S. Sarwar, L. Sekanina, Z. Vasicek, and K. Roy, "Design of power-efficient approximate multipliers for approximate artificial neural networks," in *2016 International Conference On Computer Aided Design (ICCAD) (přijato)*, 2016, p. 7.

Návrh metód generovania BIST pre vnorené pamäte v systémoch na čipe

Juraj Šubín
2. ročník, denná forma
Školiteľ: Elena Gramatová

Slovenská technická univerzita, Fakulta informatiky a informačných technológií
Ilkovičova 2, 84216 Bratislava 4, SR
juraj.subin@stuba.sk

***Abstrakt*—Príspevok opisuje návrh nového systému automatického generovania architektúry BIST pre testovanie vnorených pamätí integrovaných v systémoch na čipe. Existuje niekoľko podobných systémov, no niektoré z nich sú špecificky zamerané na konkrétne technológie, využívané v návrhárskych firmách, iné neberú do úvahy niektoré dôležité parametre. Chýba teda systém, dostatočne flexibilný k požiadavkám používateľa (návrhára), ktorý by bol dostupný pre použitie najmä na akademickej úrovni. Úlohou navrhovaného systému je rozdeliť pamäte, integrované v systéme na čipe, do testovacích okolí, ktoré testujú pamäte sériovo alebo paralelne, v závislosti od rôznych typov zadaných požiadaviek, ako napr. čas testovania, navýšenie plochy na čipe, spotreba energie. Výstupom systému má byť syntetizovateľný VHDL kód každého bloku vygenerovanej architektúry BIST. Vstupy od používateľa a jeho interakcia so systémom sú potrebné na to, aby bol systém schopný vygenerovať vyhovujúcu architektúru BIST spĺňajúcu požiadavky používateľa.**

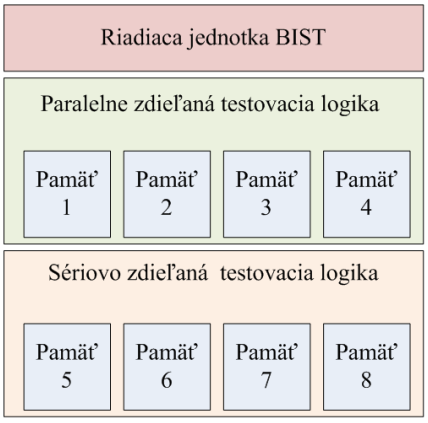
Kľúčové slová—systém na čipe; vnorená pamäť; testovanie; BIST; VHDL.

I. Úvod

Systémy na čipe (SoC – *system on chip*) sú integrované do každodenne používaných elektronických zariadení, ktoré je potrebné tetsovať po ich výrobe resp. počas ich životnosti. Súčasné technológie poskytujú možnosti vývoja zložitejších systémov s väčšou funkcionalitou, čo prispieva aj k zložitosti ich testovania. Veľká pozornosť je sústredená na testovanie pamätí vnorených v SoCs, pretože v súčasnosti zaberajú najväčší podiel z ich plochy (cca 86 %) [1]. Na jednom SoC môže byť integrovaných stovky až tisícky pamätí rôzneho typu a veľkosti, pričom všetky je potrebné otestovať. Vstavané samočinné testovanie (BIST – *built-in self-test*) je stále vhodným prístupom k testovaniu veľkého počtu vnorených pamätí. Architektúra BIST je navrhovaná ako jednoúčelová testovacia logika, ktorá je integrovaná v SoC spolu so všetkými ostatnými komponentami, a preto je každá architektúra BIST cielene navrhovaná pre potreby daného SoC. Schopnosť detegovania porúch v pamäti závisí od voľby testovacieho algoritmu. V súčasnosti najrozšírenejšie sú algoritmu typu march, ktorých hlavnými výhodami sú lineárna zložitosť, dostatočné pokrytie porúch a používanie pravidelných testovacích vzorov, preto sú aj vhodné pre použitie v BIST. Výber konkrétneho algoritmu march závisí od žiadaného pokrytia porúch a dĺžky trvania testu. Iným dôležitým aspektom, ktorý je potrebné vziať do úvahy pri testovaní pamätí, je výber adresnej schémy pre testovací algoritmus. Je známych niekoľko špeciálnych kódov, z ktorých je možné vyberať v závislosti od cieľu testovania. Napríklad, lineárna adresná schéma je jednoduchá a využívaná prevažne na detekciu porúch v jednej bunke alebo porúch spárených buniek v pamäťovej matici. Komplementárna adresná schéma je užitočná pre stresovanie adresného dekódera a pre detekciu dynamických porúch zapríčinených rýchlymi zmenami viacerých signálov adresného dekódera v jednom čase. Grayov kód je dobre známa adresná schéma, ktoré má využitie najmä pri testovaní s nízkou spotrebou energie, nakoľko po sebe idúce adresy majú Hammingovu vzdialenosť vždy rovnú 1 [2].

V súčasnosti existuje niekoľko prístupov k testovaniu veľkého počtu pamätí vnorených v SoC. Najzákladnejším prístupom je použitie obvodu BIST pre každú pamäť v SoC. Tento prístup je náročný na plochu na čipe a môžu nastať problémy s riadením testovania a zbieraním informácií o detegovaných poruchách. Mierne vylepšený prístup je použitie nezdieľaných testovacích okolí (každá pamäť má pridané vlastné testovacie okolie), riadených centralizovanou riadiacou jednotkou. Testovacie okolie pozostáva z logiky potrebnej na otestovanie pamäte (adresný generátor, údajový generátor, riadiaci blok, komparátor a komunikačné rozhranie). Nakoľko testovacie okolia sú v tomto prípade nezdieľané, problém s vysokým navýšením plochy na čipe pretrváva. Najnovší prístup využíva výhody vyplývajúce zo zdieľania testovacích okolí, ktoré sú riadené centralizovanou riadiacou jednotkou (obr. 1). Pamäte, ktoré zdieľajú testovacie okolie, môžu byť testované buď sériovo, alebo paralelne. Sériové testovanie je efektívne pre dosiahnutie nízkej spotreby energie počas testovania. Výhodou paralelného testovania je dosiahnutie krátkeho času testovania. Využitím hybridnej BIST architektúry (kombinácia sériového a paralelného testovania) je možné nájsť kompromis medzi časom testovania, navýšením plochy na čipe a spotrebou energie. V súčasnosti sa hybridné testovanie javí ako najlepší prístup k

testovaniu veľkého počtu pamätí integrovaných v SoC. Avšak, návrh takejto hybridnej BIST architektúry pre konkrétny SoC nie je jednoduchá úloha. Je potrebné prihliadať na viacero parametrov, ako napr. počet, typ a veľkosť pamätí, spotreba energie, navýšenie plochy na čipe, aplikácia testu, dĺžka testovania atď. Ručný návrh optimálnej architektúry BIST je časovo náročný proces a nájdenie optimálneho riešenia nemusí byť garantované. Z uvedeného dôvodu môže byť systém pre automatické generovanie architektúry BIST veľmi užitočný a efektívny. Existuje niekoľko podobných riešení, no niektoré z nich sú špecificky zamerané na konkrétne technológie, využívané v návrhárskych firmách, iné na druhej strane neberú do úvahy niektoré dôležité parametre. To je hlavná motivácia na návrh novej metódy generovania architektúry BIST na testovanie vnorených pamätí v SoC, ktorá bude flexibilná k požiadavkám používateľa (návrhára).



Obr. 1. Všeobecná schéma hybridnej BIST architektúry.

Článok je rozdelený do piatich častí vrátane úvodu a záveru. Druhá časť opisuje exitujúce prístupy a systémy automatického generovania BIST, tretia časť je venovaná opisu novej metódy generovania architektúry BIST, štvrtá časť je zameraná na ciele a tézy dizertačnej práce a piata časť je záverom.

II. Systémy automatického generovania BIST

V súčasnosti sú známe štyri systémy automatického generovania architektúry BIST pre vnorené pamäte na čipe [3] – [6]. Niektoré z nich boli vyvinuté pre špecifické potreby návrhárskych firiem a sú nedostupné na výskum. V tabuľke 1 sa nachádza porovanie týchto systémov podľa troch zvolených hľadísk. Porovnávané systémy sú pre jednoduchosť označené ako BIST_1 až BIST_4 s ich citáciou, nakoľko v publikáciách nemajú tieto systémy priradený špecifický názov. Hlavnou nevýhodou systému BIST_1 je, že všetky pamäte sa začínajú testovať paralelne v rovnakom čase. Tento fakt bráni systému dosiahnuť optimálne riešenie z pohľadu všetkých požiadaviek uvedených v tabuľke. Systém BIST_2 nepodporuje zdieľané sériové testovacie okolie, preto nemožno optimalizáciu architektúry zacieliť na spotrebu energie. Vstupné údaje systému BIST_3 nie sú v publikácii uvedené, preto nemožno povedať, či je parameter spotreby energie braný do úvahy pri snahe vytvoriť čo najmenší počet skupín pamätí. Architektúru v systéme BIST_4 možno optimalizovať len z pohľadu navýšenia plochy na čipe. Žiadny z uvedených systémov nepodporuje všetky základné typy pamätí – SRAM, DRAM, ROM. Ďalej chýba možnosť voľby testovacieho algoritmu a použitej adresnej schémy. Formát výstupu takisto neuvádza žiaden zo systémov. Uvedené nedostatky sú motiváciou pre návrh nového systému pre automatické generovanie architektúry BIST pre testovanie vnorených pamätí v SoC s novo navrhnutými metódami a algoritmami, kde nebudú použité štandardy IEEE1149.1 [7] alebo IEEE1687 [8].

TABUĽKA I. POROVNANIE EXISTUJÚCICH SYSTÉMOV

Prístup / Porovnanie	**Požiadavky**	**Druhy testovacích okolí**	**Poradie testovania pamätí**
BIST_1 [3]	plocha na čipe spotreba energie čas testu	nezdieľané, zdieľané paralelné& sériové	test pre všetky pamäte štartuje v rovnakom čase
BIST_2 [4]	plocha na čipe čas testu dĺžky prepojení	zdieľané paralelné	1 skupina pamätí testovaná v jednom čase
BIST_3 [5]	area overhead čas testu dĺžky prepojení najmenší počet skupín	zdieľané paralelné& sériové	získavané výpočtom
BIST_4 [6]	plocha na čipe	zdieľané paralelné& sériové	získavané výpočtom
navrhovaný systém	plocha na čipe spotreba energie čas testu	nezdieľané, zdieľané paralelné& sériové	získavané výpočtom

III. METÓDY GENEROVANIA ARCHITEKTÚRY BIST

Navrhovaný systém generovania architektúry BIST je založený na hybridnej metóde testovania a mal by byť flexibilný a škálovateľný vzhľadom na požiadavky používateľa. Vnorené pamäte budú testované cez testovacie okolia pozostávajúce z adresného a údajového generátora, riadiaceho bloku, komparátora a komunikačného rozhrania. Testovacie okolia budú riadené centralizovanou riadiacou jednotkou BIST. Tento prístup je použitý aj v iných BIST architektúrach, ale zložitosť a komplexnosť testovacích okolí a riadiacej jednotky tu závisí od požiadaviek používateľa a návrhových obmedzení spojených s konkrétnym SoC. Architektúra BIST môže byť prepojená aj na blok vstavanej samočinnej opravy (BISR – *built-in self-repair*). Preto je viac efektívne generovať architektúru BIST podľa požiadaviek používateľa ako použiť štandardnú architektúru a integrovať ju na SoC, pričom by sa nebrali do úvahy niektoré špecifické parametre. Pre vývoj takéhoto systému je potrebné navrhnúť niekoľko metód pre vygenerovanie optimálnej architektúry BIST vzhľadom na rôzne požiadavky, a to metóda voľby adresných schém, metóda rozdelenia pamätí do testovacích okolí, metóda zostavenia riadiaceho bloku na riadenie BIST všetkých pamätí vnorených v SoC, metóda ohodnotenia žiadaných parametrov, podľa ktorých sa bude riadiť

rozdelenie pamätí a pod. Vygenerovaná architektúra BIST bude jedinečná pre danú kombináciu žiadaných parametrov, typy pamätí integrovaných v SoC. Predpokladá sa využitie týchto parametrov, rozdelených do troch skupín:

I. Parametre pamätí
- počet a veľkosť pamätí,
- typ pamätí (DRAM, SRAM, ROM),
- rozhranie pamätí,
- frekvenica operácií zápisu a prečítania,
- zdieľané / nezdieľané testovacie okolie.

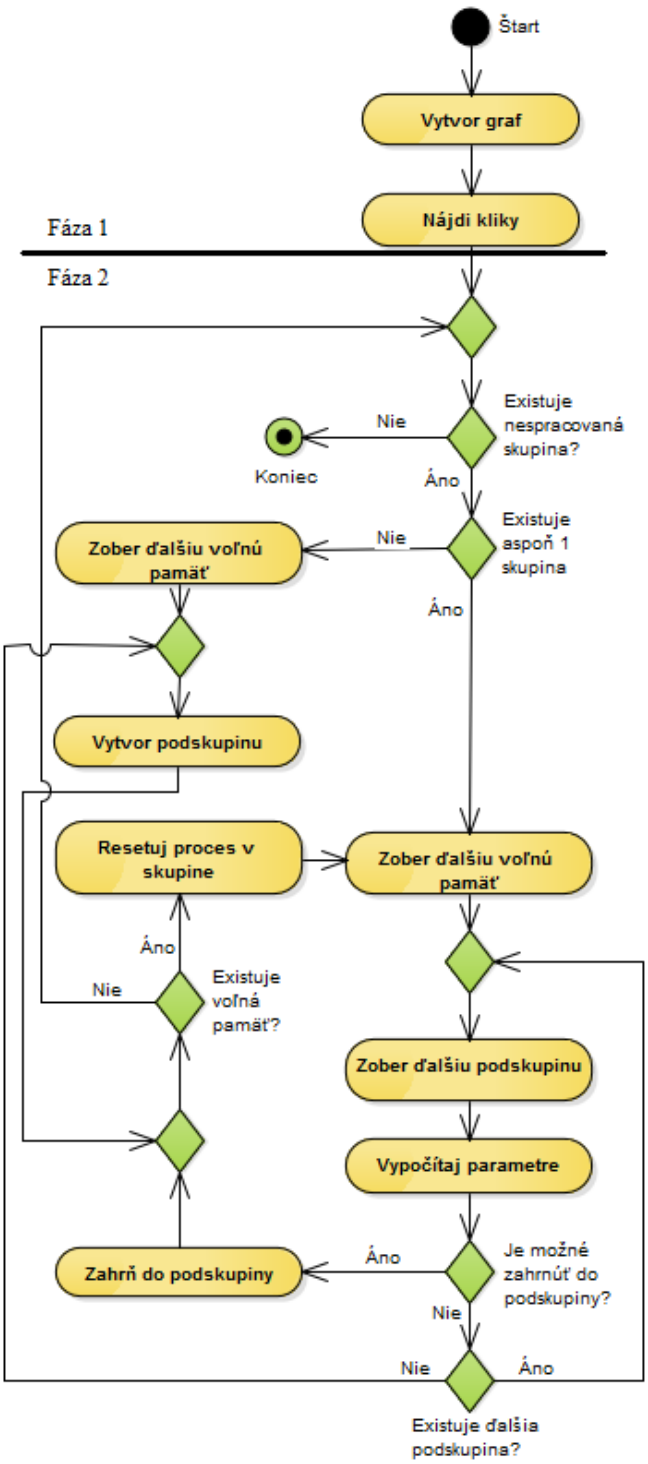
II. Parametre testovania
- typ testu (rôzne algoritmy march),
- adresné schémy (napr. lineárna, komplementárna, atď.),
- čas testu,
- obmedzenia v spotrebe energie.

III. Špecifické parametre
- požiadavka optimalizácie architektúry,
- obmedzenia v navýšení plochy na čipe,
- prepojenie na BISR,
- používateľsky definované skupiny pamätí.

Základom systému je rozdelenie pamätí do testovacích okolí, ktoré si vyžaduje jednak návrh algoritmu na ich rozdelenie v závislosti od typu pamätí a zvoleného cieľu optimalizácie architektúry, návrh metód ohodnotenia parametrov spotreby energie, pridanej plochy a času testovania, ktoré budú súčasťou tohto algoritmu. Jedna pamäť môže byť priradená len pod jedno testovacie okolie. Počet pamätí pod jedným testovacím okolím a typ testovacieho okolia budú vždy ovplyvnené hraničnými hodnotami troch parametrov: časom testu (ak je zadaný), spotrebou energie a navýšením plochy na čipe. Nový algoritmus pre nájdenie optimálneho rozdelenia pamätí do testovacích okolí využíva grafové metódy a je zobrazený na obr. 2. Algoritmus pozostáva z dvoch fáz. Prvá fáza je tzv. inicializačná fáza, ktorej cieľom je vytvoriť neorientovaný graf reprezentujúci kompatibility pamätí z hľadiska ich testovania. Každá pamäť je reprezentovaná vrcholom grafu. Ak sú 2 vrcholy grafu prepojené hranou, znamená to, že dané pamäte sú kompatibilné z hľadiska testovania a môžu byť priradené pod jedno testovacie okolie, či už sériové, alebo paralelné. V navrhovanom algoritme sú dve pamäte považované za kompatibliné z hľadiska testovania v prípade, ak:
- sú rovnakého typu,
- pracujú na rovnakej operačnej frekvencii,
- budú testované rovnakým testom march,
- bude použitá rovnaká adresná schéma.

Ďalšou časťou fázy 1 je nájdenie najväčších kompletných podgrafov (kliky) grafu a vytvorenie skupín pamätí kompatibilných pre testovanie.

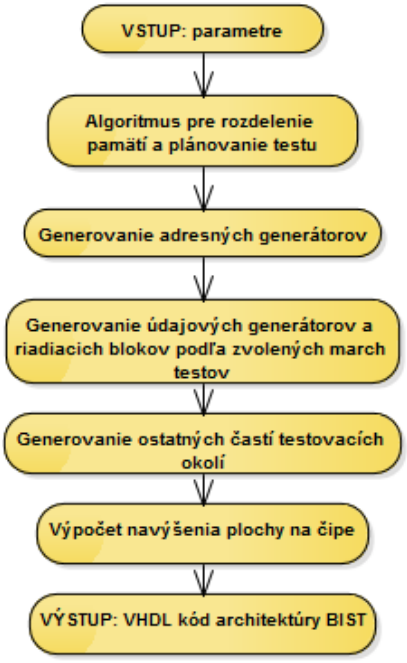
V druhej fáze algoritmu sa vykonavajú výpočty a prerozdeľovacie rozhodnutia. Cieľom tejto fázy je analyzovať vytvorené skupiny pamätí jednu po druhej a rozdeliť ich ďalej



Obr. 2. Algoritmus rozdelenia pamätí do testovacích okolí.

do podskupín, ak je to potrebné. Metóda na takéto rozdelenie musí akceptovať špecifické parametre, uvedené vyššie, a zadané obmedzenia pre vytváranie podskupín. Po ukončení tejto fázy sú už pamäte rozdelené do finálnych podskupín, pre ktoré budú vygenerované zdieľané testovacie okolia. Prioritou algoritmu je dosiahnuť optimálnosť generovanej architektúry z pohľadu zvoleného parametra (parametrov) optimalizácie.

Výsledkom systému automatického generovania BIST je optimálna architektúra BIST s týmito výstupmi: VHDL model blokov BIST, prepojenie na BISR, ak je žiadané, optimálny plán testu, odhad spotreby energie počas testovania, výpočet navýšenia plochy. Implemetáciou metódy vznikne nový akademický systém pre automatické generovanie architektúry BIST – MBISTGen s grafickým používateľským rozhraním. Základná schéma systému MBISTGen je zobrazená na obr. 3.



Obr. 3. Všeobecná schéma systému MBISTGen.

IV. CIELE DIZERTAČNEJ PRÁCE

Cieľ dizertačnej práce je návrh metód na automatické generovanie architektúry BIST pre vnorené pamäte v SoC. Vstupom sú požiadavky používateľa (spotreba energie, dĺžka testu a pridaná plocha v SoC), parametre pamätí, existujúce metódy testovania porúch v pamätiach. Výsledná architektúra BIST bude opísaná v jazyku VHDL. Preto tézy dizertačnej práce sú:

• Špecifikácia požiadaviek a obmedzení na architektúru BIST na návrh optimálneho BIST na veľký počet vnorených pamätí.

• Návrh a vývoj blokov BIST, ich modelovanie vo VHDL, flexibilných z pohľadu počtu a rôznorodosti pamätí vnorených do čipu, ako aj testov march či aplikácie testov pri použití rôznych adresných schém.

• Návrh optimálnych architektúr sériovo-paralelných BIST, škálovateľných na počet pamätí, čas testovania a minimalizáciu spotreby energie pri testovaní.

• Návrh novej metódy na rozdelenie pamätí integrovaných v SoC na sériové a paralelné testovanie v závislosti od požiadaviek návrhára na plochu, čas testovania a spotrebu energie počas testovania.

• Overenie navrhnutej metódy na automatické generovanie BIST vo VHDL pre vybrané SoC a zadané požiadavky na testovanie pamätí s ich rôznymi zostavami a typov.

V. ZÁVER

Stúpajúca miera integrácie čoraz väčšieho počtu jadier a funkčných blokov na čipe si vyžaduje nové prístupy k riešeniu ich testovania, a to so zameraním sa na testovanie pamätí, ktoré zaberajú najväčšiu časť plochy čipu. Návrh vstavanej architektúry vykonávajúcej samočinné testovanie vysokého počtu pamätí nie je triviálna úloha. Z analýzy existujúcich systémov automatického generovania BIST pre pamäte v SoC bol identifikovaný priestor pre vývoj nového systému automatického generovania testovacej architektúry ponúkajúceho navyše oproti existujúcim prístupom napríklad možnosť voľby testovacieho algoritmu, adresnej schémy, alebo poskytujúceho výstup vo formáte VHDL. Ďalšia práca bude sústredená na implementáciu navrhnutého algoritmu a vhodných blokov BIST.

POĎAKOVANIE

Táto práca bola podporovaná Agentúrou na podporu výskumu a vývoja na základe národného projektu VEGA 1/0616/14.

LITERATÚRA

[1] International Technology Roadmap for Semiconductors (ITRS). 2011 Edition. Test and Test Equipment. http://www.itrs.net.

[2] A.J. van de Goor, H. Kukner, S. Hamdioui, "Optimizing Memory BIST Address Generator Implementations," in IEEE International Conference on Design & Technology of Integrated Systems in Nanoscale Era, 2011, p. 6.

[3] L. Zaourar, Y. Kieffer, A. Wenzel, "A Complete Methodology for Determining Memory BIST Optimization under Wrappers Sharing Constraints," in Asia Symposium on Quality Electronic Design (ASQED), 2011, pp. 46-53.

[4] Ch. Tzuo-Fan, Ch. Wen-Chi, L. Chien-Mo, Ch. Yao-Wen, L. Kuan-Yu, Ch. Ming-Tung, T. Min-Hsiu, T. Chih-Mou, "BIST Design Optimization for Large-Scale Embedded Memory Cores," in IEEE/ACM International Conference on Computer-Aided Design - Digest of Technical Papers, 2009, pp. 197-200.

[5] A.B. Kahng, I. Kang, "Co-Optimization of Memory BIST Grouping, Test Scheduling, and Logic Placement," in IEEE Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014, p. 1-6.

[6] M. Miyazaki, T. Yoneda, H. Fujiwara, "A Memory Grouping Method for Sharing Memory BIST Logic," in Asia and South Pacific Conference on Design Automation, 2006, pp. 671-676.

[7] Standard Test Access Port and Boundary-Scan Architecture, IEEE Std 1149.1, 2008. [online] Available at: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=938734 [Accessed 11 Jan. 2016].

[8] Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device, IEEE Std 1687, 2014. [online] Available at: http://www.techstreet.com/ieee/products/vendor_id/3931 [Accessed 11 Jan. 2016].

Koncept Field Programmable Neural Networks odolný proti poruchám

Martin Krčma
2. ročník, prezenční studium
Školitel: Zdeněk Kotásek

Vysoké Učení technické v Brně
Božetěchova 2, 612 66 Brno, ČR
ikrcma@fit.vutbr.cz

***Abstrakt*—Tento článek popisuje koncept Field Programmable Neural Networks (FPNN) sloužící pro implementaci neuronových sítí v FPGA, prezentuje model a techniky napomáhající zvýšení odolnosti struktur založených na tomto konceptu proti poruchám. Dále popisuje jeden z provedených experimentů s jednou s prezentovaných technik.**

I. Úvod

V oblasti návrhu systémů odolných proti poruchám existují tři jasně oddělitelné skupiny metodik: a) metodiky konstrukce systémů odolných proti poruchám s garantovaným chováním systému jako systému odolného proti poruchám [1][2], b) metodiky pro detekci chybného chování systému [3], c) metodiky umožňující systému se z poruchy zotavit a obnovit jeho funkci [4].

V našem předchozím výzkumu jsme se zabývali všemi výše zmíněnými metodikami [5]. Dále jsme se v minulosti zaměřili primárně na vývoj těchto metodik pro klasické digitální systémy, zatímco nyní se zabýváme i návrhem systémů založených na neuronových sítích odolných proti poruchám. Je to specifická oblast která vyžaduje zvláštní přístupy. Principy vyvinuté naším týmem v této oblasti patří do oblasti výše uvedené v sekci c).

V tomto článku se zaměřujeme na jednu z možných implementací neuronových sítí v FPGA - na tzv. Field Programmable Neural Networks (FPNN) a na způsoby, jak zvýšit jejich odolnost proti poruchám. Článek má následující strukturu - zaprvé popíšeme koncept FPNN, dále představíme rozšíření modelu o odolnost proti poruchám a nakonec uvedeme experimentální výsledky.

II. Field Programmable Neural Networks

Koncept FPNN [6] je navržen tak, aby zjednodušoval implementaci neuronových sítí v hradlových polích FPGA tím, že je pro to svými vlastnostmi vhodnější. Zjednodušení spočívá hlavně v hlavní vlastnosti FPNN - flexibilní struktuře, která umožňuje dosáhnout značného sdílení zdrojů za zachování vysoké paralelizace výpočtu.

FPNa jsou jednou z možnách implementací neuronových sítí v FPGA. Stejně jako jiné koncepty, je i tato zranitelná vůči poruchám, hlavně vůči, pro FPGA příznačným, poruchám typu SEU (Single Event Upset), tedy změna hodnoty bitu v důsledku dopadu nabité částice.

FPNN byla původně definována pomocí formálního modelu [6], který jsme přizpůsobili našim potřebám a rozšířili o další možnosti včetně odolnosti proti poruchám.

A. Základní pojmy konceptu FPNN

FPNN jsou definovány [6] jako orientovaný graf (N, E). Uzly (prvky N) a hrany (prvky E) reprezentují dva typy výpočetních jednotek. Uzly jsou nazývány *aktivátory* a nahrazují původní neurony. Hrany slouží jako aproximace původního synaptického propojení a jsou tvořeny sekvencemi propojených jednotek nazývaných *spoje*. Právě implementace hran sekvencemi dílčích jednotek umožňuje flexibilitu struktury FPNN. Oba typy jednotek (aktivátory a spoje) jsou společně nazývány *neurální zdroje* a disponují několika operátory zajišťujícími výpočet. Aktivátory jsou vybaveny dvěma operátory. Prvním je *iterační operátor* i, který je binární funkcí nad oborem reálných čísel. Tento operátor slouží ke sběru potenciálu, podobně jako v neuronu. Operátor je cyklicky aplikován na své vstupy, kterými jsou vstup aktivátoru a jeho vnitřní registr. Výstup se uloží opět do vnitřního registru. Po jistém počtu cyklů je obsah registru použit jako vstup pro druhý unární reálný *funkční operátor* f, který provádí výpočet aktivační funkce. Výstup tohoto operátoru je použit jako výstup celého aktivátoru.

Spoje disponují množinami *afinních operátorů*, které slouží pro aproximaci násobení vahami. Operátor provádí přepočet vstupu x pomocí hodnot W a T (1). Aproximace vah je prováděna sekvencemi propojených spojů realizujících sekvenci aplikací afinních operátorů. Každá původní synapse je tedy aproximována nějakou posloupností spojů (afinních operátorů). Počet operátorů může být buď stejný jako počet všech aktivátorů, od kterých do spoje proudí data, kdy pro data z každého tohoto aktivátoru je vyhrazen jeden operátor a FPNN prování zcela přesnou aproximaci původní sítě. Druhým možným případem je, kdy má spoj stejný počet operátorů, jako je počet všech přímo připojených sekvencí spojů. Zde již dochází ke sdílení operátoru mezi skupinami synapsí aproximovanými danými sekvencemi spojů a tím dochází ke snížení aproximační síly FPNN, zároveň však k jeho menší náročnosti

na zdroje. Třetím případem je to, když má spoj pouze jeden operátor pro všechny aproximované synapse. Aproximační síla je nejnižší, obecná násobička realizující operátor ale může být nahrazena konstantní násobičkou, čímž dojde k další úspoře zdrojů. Problematiku převodu vah na afinní operátory jsme popsali v [7].

$$\alpha_{(p,n)} = W_n(p) \times x + T_n(p); W_n(p), T_n(p) \in \mathbb{R}; \\ p, n \in N; (p, n) \in E \quad (1)$$

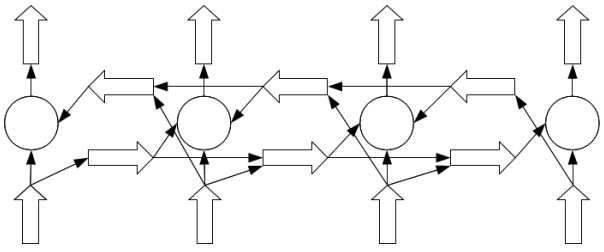
Propojení neurálních zdrojů je povoleno pouze pro kombinace aktivátor-spoj a spoj-spoj. To dává možnost vytvořit téměř libovolné struktury aktivátorů propojených pomocí sekvencí spojů. Kombinace aktivátor-aktivátor není povolena, protože není v souladu s funkcí neuronových sítí.

Pokud jsou definovány pouze graf (N, E), iterační a funkční aktivátory, pak se tato struktura nazývá FPNA (Field Programmable Neural Array) [6] a definuje celou třídu potenciálních instancí. Pokud jsou definovány i parametry závislé na struktuře - struktura propojení neurálních zdrojů, afinní operátory a počty cyklů iteračních operátorů, jedná se o FPNN, které je instancí nějakého FPNA.

B. Mřížové FPNN

Zvláštním typem FPNN je *mřížové FPNN*. Je to FPNN, které má taková strukturální omezení, která si vynucují, aby nabralo podobu mříže s aktivátory v průsečících. Důvodem pro zavedení takové struktury je to, že tato struktura je podobná propojovací matici FPGA, čímž je takové FPNN snadněji namapovatelné do tohoto zařízení.

Příklad mřížového FPNN je na Obrázku 1. Na tomto obrázku kruhy znázorňují aktivátory, široké šipky spoje a tenké šipky datové propoje mezi neurálními zdroji. Orientace šipek ukazuje směr procházejících dat. Z obrázku je zřejmé, že každý aktivátor má na výstupu jeden spoj, který realizuje propojení s následující vrstvou. A to buď přímo na jiný aktivátor v následující vrstvě nebo na sekvenci spojů, která zajišťuje propojení s ostatními aktivátory ve vrstvě. Jsou zde dvě takové sekvence vedoucí opačným směrem.



Obrázek 1. Mřížové FPNN složené z aktivátorů (kruhy), spojů (silné šipky) a datových propojů (tenké šipky)

Tento model činí FPNN flexibilní, snadno konstruovatelné a rozšiřitelné. Nicméně, přináší jistou časovou i prostorovou režii díky potřebě implementace obsluhy komunikace.

III. Odolnost proti poruchám

Pro potřeby modelování, popisu metod, modelování a simulace jsme vytvořili následující formální model jednotky odolné proti poruchám sloužící jako základní stavební blok vyšších struktur založených na množině asynchronně komunikujících výpočetních jednotek. Tento model je zobecněním modelu neurálních zdrojů odolných proti poruchám sloužící ke konstrukci FPNN odolných proti poruchám, který byl popsán v [8]. Ten byl založen na konceptu FPNA/FPNN. Účelem modelu je dát základ základním stavebním prvkům výpočetní architektury odolné proti poruchám, a nabídnout různé možnosti zlepšení této vlastnosti. *Model nabízí možnost využít technik založených na redundanci a technik založených na změně parametrů a rekonfiguraci*.

Definice III.1 (Jednotka odolná proti poruchám)**.** je množina $rUnit^n = \{S, s, R, U, I, C, m, c, D\}$, kde:

- S je množina nastavení jednotky (množina hodnot jeho parametrů)
- $R = \{unit_1, unit_2, .., unit_n\}$ je množina n identických jednotek s nastavením $s \in S$
- $U = \{0, 1\}^n$ je množina binárních příznaků určujících aktivitu jednotek
- I je operátor identity (x je vstup jednotky): $I(x) = x; x \in \mathbb{R}$
- C je konstantní operátor (x je vstup jednotky): $C(x) = c; x, c \in \mathbb{R}$
- m je stupeň majority
- $D \in \{f, r, i, t\}$ mód činnosti:
  - f - výstup je brán z první aktivní jednotky,
  - r - výstup je brán jako majorita prvních m aktivních jednotek,
  - i - výstup je brán z operátoru identity,
  - t - výstup je brán z konstantního operátoru.

Příklad jednotky zabezpečeného pomocí TMR:

$$TMRUnit = \{\{s\}, s, \{u_1, u_2, u_3\}, \{1, 1, 1\}, \emptyset, 2, r\} \quad (2)$$

Model definuje jednotku, která je složena z n identických kopií výpočetní jednotky (neurálního zdroje), které počítají paralelně. Aktivita jednotek je určena příznaky v množině U. Jednotky, jež mají patřičný příznak nastaven, pracují, ostatní jsou deaktivované. To, z které z nich je brán výstup celé jednotky je určeno módem, v němž jednotka pracuje. V módu f je výstup jednotky brán z první pracující jednotky (jednotky s nejnižším indexem). V tomto módu je možné jednotku používat jako systém zabezpečený záložními kopiemi. Pokud je první výpočetní jednotka postižena poruchou, je možné změnou příznaků přepnout na další funkční výpočetní jednotku. Jednotky tedy slouží jako záložní.

V módu r je výstup brán jako m násobná majorita výstupů všech aktivních jednotek. V tomto módu může tedy jednotka pracovat jako systém zabezpečený metodou DMR či TMR.

V ostatních módech jednotka využívá dvou zvláštních operátorů, jenž slouží k dočasném či trvalému zotavení z poruchy. Prvním operátorem je operátor identity, aktivní v

módu i. Tento operátor má stejnou hodnotu výstupu jako vstupu. Kopíruje tedy svůj vstup na svůj výstup a slouží tak jako registr. Důvodem, proč využít tento operátor, je možnost obnovení z poruchy, aniž by byl potřeba velký zásah do jednotky. Za předpokladu, že síť je dostatečně robustní a ztráta této jednotky nebude mít kritický dopad na její výkon, můžeme tuto jednotku v případě poruchy nahradit registrem, čímž znovu obnovíme komunikaci mezi těmi částmi sítě, jež porušená jednotka propojovala. Odstíníme také zbytek obvodu od nepředvídatelného vlivu porušené jednotky a stabilizujeme jej ve známém stavu. Popřípadě můžeme operátor využít k dočasnému snížení dopadu poruchy, dokud nebude provedena účinnější oprava. V případě vyšší tolerance k odchylkám ve výkonu systému můžeme operátor využít i trvalému zotavení i v případě, že jeho aktivace neobnoví výkon na plnou hodnotu.

Druhým operátorem je konstantní operátor. Tento operátor je aktivní v módu t a nastavuje na výstup jednotky předem danou konstantu c. Stejně jako operátor identity je účelem konstantního operátoru částečné či plné zotavení z poruchy dané jednotky a uvedení systému do stabilního stavu. I zde je předpokladem dostatečná robustnost systému nebo dostatečně vysoká tolerance k možným odchylkám ve výkonu systému. Na rozdíl od operátoru identity však konstantní operátor umožňuje vzít v potaz chování jednotky. Jeho použitím můžeme na výstup jednotky vystavit hodnotu, která má např. statisticky nejvyšší výskyt, a jednotka tak bude v daném poměru případů poskytovat korektní výsledky a výkon systému nebude ovlivněn. Je možné použít např. i medián, různé typy průměrů či různé heuristiky určující nejlepší hodnotu c. Použití obou operátorů nevyžaduje přeučení sítě.

Dalším možným způsobem zotavení z poruchy je změna parametrů. Každá jednotka obsahuje množinu parametrů s určujících její funkci. Dále je zde množina S obsahující další potenciální množiny nastavení parametrů jednotky, na něž jsme schopní změnou s jednotku přepnout. Tím můžeme potenciálně dosáhnout zotavení, nebo kompenzace vlivu poruchy jednotky. Jelikož je zde uvažovaný systém tvořený sítí propojených, asynchronně komunikujících jednotek vytvořených dle popsaného modelu, znamená to, že změna v kterékoliv jednotce systému ovlivní všechny jednotky připojené na její výstup. Tímto způsobem může vzniknuvší porucha ovlivnit značnou část systému. Stejným způsobem jsme ale schopni chybu kompenzovat. Vhodnou změnou parametrů *ostatních* jednotek můžeme dosáhnout stavu, kdy systém počítá správné, nebo přibližně správné výsledky i za přítomnosti neopravené poruchy. Toto je tedy cesta, jak provést zotavení z poruchy, již není možné, nebo jen obtížně, opravit. Rovněž můžeme tuto metodu použít pro dočasnou kompenzaci vlivu poruchy, než bude provedena plná oprava a zotavení.

IV. Experimentální výsledky

Aktivace operátoru identity spojů má za následek jejich odstranění ze sekvence, čímž dojde k odebrání jejich afinních operátorů z aproximačních sekvencí a tím k narušení aproximace vah. To může mít, v závislosti na momentální situaci, různý dopad. Různé spoje a jejich afinní operátory mají různou míru vlivu na výsledek výpočtu, která závisí na konkrétní hodnotě operátorů a na hodnotě operátorů ostatních spojů v řetězci. V některých případech může odstranění spoje mít jen nepatrný vliv, jindy zcela zásadní. Pokud předpokládáme, že nějaký spoj je zásadní (například má afinní operátory s vysokými, nebo naopak nízkými hodnotami), a hodláme použít techniku operátorů identity jako jednu z technik odolnosti proti poruchám, můžeme využít upraveného procesu mapování [7], [8] neuronové sítě na FPNN, které tento scénář do výsledného FPNN započítá. Základním principem je přenesení aproximovaných vah na přímé předchůdce zabezpečovaného spoje (který se do mapování vůbec nezapočítává, jako by už byl jeho operátor identity zapnut), tedy do jejich afinních operátorů. Díky tomu jsou synapse původně aproximované zabezpečovaným spojem aproximovány jeho předchůdci. Zabezpečovaný spoj je pak namapován tak, aby zlepšoval výslednou aproximaci. Vzhledem k tomu, že použití této techniky zvýší sdílení afinních operátorů mezi aproximovanými synapsemi v předchůdcích, dojde ke snížení přesnosti aproximace. Toto snížení je opět závislé na konkrétní situaci. Použitelnost techniky je tedy odvislá od konkrétní situace.

S popsanou technikou jsme prováděli experimenty, z nichž jeden je popsán v této sekci. Experimentovali jsme se základní neuronovou sítí pro výpočet logického exkluzivního součtu, která nám díky své jednoduchosti umožňuje ilustrovat mnoho kombinací použití identity operátoru a použití upraveného mapování pro snížení negativního dopadu.

Původní síť a odvozené mřížové FPNN (jeho výpočet byl simulován [9]) měly dva vstupy, tři neurony ve skryté vrstvě a jeden výstup. Jako první krok jsme měřili dopad poruch spojů ve skryté vrstvě na správnost klasifikace (prováděné FPNN) všech čtyř vstupních vektorů. Tabulka I obsahuje informace o tom, jak nijak nezabezpečené FPNN ovlivní ztráta jednoho spoje. Ve sloupci *Nesprávně* je uvedeno, kolik ze vstupních čtyř vektorů bylo klasifikováno nesprávně. Ve sloupci *Shoda* je souhrnná procentuální shoda aktuálního výsledku s původním (korektním výsledkem) klasifikace. Sloupec *Zabezpečitelný* udává, zda je na daný spoj aplikovatelný upravený algoritmus.

Tabulka I
DOPAD ZRÁT SPOJŮ NA NA KLASIFIKACI 4 VSTUPNÍCH VEKTORŮ

Chybějící spoj	Nesprávně	Shoda [%]	Zabezpečitelný
-	0	100	-
(n4,n6)	0	100	Ne
(n4,n3)	0	100	Ano
(n3,n4)	1	75	Ano
(n4,n5)	1	75	Ano
(n3,n6)	1	75	Ne
(n5,n4)	1	75	Ano
(n1,n3)	2	50	Ne
(n2,n5)	2	50	Ne
(n5,n6)	2	50	Ne

Tabulka II obsahuje výsledky experimentů s mapovacím algoritmem. Byly provedeny experimenty se všemi možnými kombinacemi ztrát spojů (včetně žádných ztrát - tedy plně fungujícího FPNN, kdy byl měřen dopad použití mapovacího algoritmu na neporušené FPNN) a zabezpečení spojů pomocí

mapovacího algoritmu (zabezpečitelných spojů). Ve sloupci *Nesprávně* je uvedeno, kolik ze vstupních čtyř vektorů bylo klasifikováno nesprávně. Ve sloupci *Shoda* je souhrnná procentuální shoda aktuálního výsledku s původním (korektním výsledkem) klasifikace.

Tabulka II
DOPAD ZTRÁT SPOJŮ A JEJICH ZAPEZPEČENÍ NA KLASIFIKACI 4 VSTUPNÍCH VEKTORŮ

Zabezpečené spoje	Porušené spoje	Nesprávně	Shoda [%]
(n4,n5)	-	0	100
(n4,n5)	(n4,n5)	0	100
(n3,n4)	-	1	75
(n3,n4)	(n3,n4)	2	50
(n5,n4)	-	1	75
(n5,n4)	(n5,n4)	1	75
(n4,n5),(n3,n4)	-	0	100
(n4,n5),(n3,n4)	(n4,n5)	0	100
(n4,n5),(n3,n4)	(n3,n4)	0	100
(n4,n5),(n3,n4)	(n4,n5),(n3,n4)	0	100
(n4,n5),(n5,n4)	-	1	75
(n4,n5),(n5,n4)	(n4,n5)	0	100
(n4,n5),(n5,n4)	(n5,n4)	1	75
(n4,n5),(n5,n4)	(n4,n5),(n5,n4)	1	75
(n3,n4),(n5,n4)	-	1	75
(n3,n4),(n5,n4)	(n3,n4)	2	50
(n3,n4),(n5,n4)	(n5,n4)	1	75
(n3,n4),(n5,n4)	(n3,n4),(n5,n4)	2	50
(n3,n4),(n4,n5), (n5,n4)	-	1	75
(n3,n4),(n4,n5), (n5,n4)	(n3,n4)	2	50
(n3,n4),(n4,n5), (n5,n4)	(n4,n5)	1	75
(n3,n4),(n4,n5), (n5,n4)	(n5,n4)	1	75
(n3,n4),(n4,n5), (n5,n4)	(n3,n4),(n4,n5)	2	50
(n3,n4),(n4,n5), (n5,n4)	(n4,n5),(n5,n4)	1	75
(n3,n4),(n4,n5), (n5,n4)	(n3,n4),(n5,n4)	2	50
(n3,n4),(n4,n5), (n5,n4)	(n3,n4),(n4,n5), (n5,n4)	2	50

Z tabulky je zřejmé, že při aplikaci upraveného mapovacího algoritmu došlo v pěti případech ke zvýšení odolnosti proti poruchám, kdy po injektáži poruchy počítalo FPNN stále správné výsledky. Ve čtyřech případech naopak došlo k tomu, že použití techniky snížilo přesnost aproximace, které se projevilo i v neporušeném stavu.

V. ZÁVĚR

Jak výsledky experimentů naznačují, v některých případech aplikace mapovacího algoritmu napomohla zvýšení odolnosti FPNN vůči vlivu aktivace operátorů identity. V jiných případech výkon zůstal stejný a ve zbytku se výkon FPNN snížil až na polovinu. Tyto výsledky jsou očekávané, protože jak robustnost FPNN vůči poruše, tak i dopad a využitelnost upraveného mapovacího algoritmu jsou přímo odvislé od konkrétní váhové funkce v konkrétní síti a tím pádem na daném jednom FPNN. Pro určení možností aplikace operátorů identity a upraveného mapovacího algoritmu je tedy nutné vyvinout heuristiku schopnou určit, na které konkrétní spoje jsou tyto techniky aplikovatelné s přínosem.

VI. BUDOUCÍ ČINNOST

Ve svém dalším výzkumu se hodlám věnovat dalšímu rozvoji technik zabezpečování FPNN, implementaci a rozšiřování modelů FPNN odolných proti poruchám a jeho kombinace s jinými technikami zvyšování odolnosti proti poruchám. Důležitou částí práce bude implementace experimentů v FPGA a testování efektivity technik pomocí injektáže poruch do designu. Následovat bude srovnání s jinými způsoby implementace z hlediska odolnosti proti poruchám a zhodnocení přínosu použití zabezpečených FPNN a technik s nimi souvisejícími. Rovněž budou provedeny experimenty s reálnými aplikacemi v podobě implementace zabezpečených řadičů.

Dále se budu věnovat výzkumu možných způsobů využití rekonfigurace FPGA pro zotavení se systému z poruch, přičemž se budu věnovat hlavně online variantě rekonfigurace. Její využití bude mimo jiné právě v oblasti FPNN při využití techniky operátorů identity. Princip tohoto využití bude spočívat v tom, že pokud oblast FPGA, kde leží spoj, bude porušena, provede se na základě analýzy zbývajících prostředků rekonfigurace, která vytvoří registr, obnoví datové a synchronizační propojení a realizuje tak operátor identity daného spoje. Rovněž by touto metodou bylo možné provádět změny parametrů a jiné způsoby zotavení z poruch.

PODĚKOVÁNÍ

Tato práce byla podpořena Ministerstvem školství, mládeže a tělovýchovy z Národního Programu Udržitelnosti (NPU II); projektem IT4Innovations excellence in science - LQ1602. Dále byla tato práce podpořena Vysokým Učením Technickým v Brně pod číslem FIT-S-14-2297 a grantem ARTEMIS JU čísla 641439 (ALMARVI).

LITERATURA

[1] J. A. Cheatham, J. M. Emmert, and S. Baumgart, "A survey of fault tolerant methodologies for fpgas," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 11, no. 2, pp. 501–553, 2006.

[2] U. Sharma, "Fault tolerant techniques for reconfigurable platforms," in *A2CWiC '10: Proceedings of the 1st Amrita ACM-W Celebration on Women in Computing in India*. USA: ACM, 2010, pp. 1–4.

[3] G. Yan, Y. Han, and X. Li, "Svfd: A versatile online fault detection scheme via checking of stability violation," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst*, vol. 19, pp. 1627–1640, 2011.

[4] M. G. Gericota, L. F. Lemos, G. R. Alves, and J. M. Ferreira, "Online self-healing of circuits implemented on reconfigurable fpgas," in *IOLTS '07: Proceedings of the 13th IEEE International On-Line Testing Symposium. Washington, DC, USA*. IEEE Computer Society, 2007, pp. 217–222.

[5] S. Martin, K. Jan, K. Zdenek, and M. Lukas, "Fault tolerant system design and seu injection based testing," *Microprocessors and Microsystems. Amsterdam: Elsevier Science*, vol. 37, pp. 155–173, 2013.

[6] B. Girau, *FPGA Implementations of Neural Networks*. Springer US, 2006, ch. FPNA: Concepts and Properties, pp. 63–136.

[7] M. KRCMA, J. KASTIL, and Z. KOTASEK, "Mapping trained neural networks to fpnns." in *IEEE 18th International Symposium on Design and Diagnostics of Electronic Circuits and Systems. Belgrade.* IEEE Computer Society, 2015, pp. 157–160.

[8] ——, "Fault tolerant field programmable neural networks." in *Nordic Circuits and Systems Conference (NORCAS)*, 2015, pp. 1–4.

[9] M. Krcma, *The neural networks acceleration in FPGA. Master thesis.* Faculty of Information Technology, Brno University of Technology; Brno, 2014.

Heterogénne smerovanie v kapilárnych sieťach internetu vecí s použitím kompozitnej metriky

Ondrej Perešíni
2. ročník, PhD. prezenčné štúdium
Školiteľ: Tibor Krajčovič

Fakulta informatiky a informačných technológií, Slovenská technická univerzita v Bratislave
Ilkovičova 2, 842 16 Bratislava 4
ondrej.peresini@stuba.sk

***Abstrakt*— Rozvoj zariadení internetu vecí je priamo závislý od dostupnosti lacnej a energeticky efektívnej komunikačnej technológie. Súčasne používané komunikačné technológie nedokážu efektívne zabezpečiť všetky kladené požiadavky a takto vzniknuté siete majú nedostatky v podobe obmedzenej priepustnosti, krátkeho dosahu alebo vysokej ceny. Neexistuje univerzálna technológia, ktorá by pokryla všetky popisované aspekty a z tohto dôvodu vzniká zoskupenie rôznych heterogénnych komunikačných technológií do jednej spoločnej kapilárnej siete. Rôzne aplikácie majú pritom aj rôzne požiadavky na komunikačné parametre a teda aj použitú komunikačnú technológiu. Súčasné smerovacie protokoly nedokážu efektívne pokrývať túto potrebu a na základe rôznych komunikačných požiadaviek dynamicky vyberať medzi rôznymi technológiami. Na zabezpečenie efektívnej komunikácie navrhujeme vylepšenie smerovacieho protokolu RPL a použitej hybridnej metriky zohľadňujúcej požiadavky aplikácií.**

Kľúčové slová—Kapilárne a heterogénne siete, Internet Vecí, sieťové smerovanie, protokol RPL

I. Úvod

Digitalizácia a inteligentné riadenie domácností sa stáva neoddeliteľnou súčasťou budovaných a aj existujúcich domácností. Vývoj takýchto systémov prináša množstvo otázok a výziev, ktoré smerujú najmä na oblasť ich zabezpečenia, komunikácie a kompatibility medzi rôznymi systémami. Práve do tejto oblasti vstupuje myšlienka internetu vecí (IoT), ktorej cieľom je pospájať všetky zariadenia do jedného systému, ktorý bude zdieľať pripojenie do internetu. Všetky takéto zariadenia pritom poskytujú určitú funkcionalitu, ktorá prináša dodatočnú hodnotu pre používateľa, keďže inteligentné objekty vzájomne kooperujú s fyzickými alebo virtuálnymi zdrojmi. Takéto zdroje sú charakteristické vysokým stupňom heterogénnosti, čím poskytujú rôznu funkcionalitu aj zariadeniam, ktoré boli pôvodne len jednoúčelové. Aby bolo možné takéto zariadenia vzájomne sieťovo poprepájať, tak je potrebné navrhnúť vhodný model kapilárnych sietí, ktorý pozostáva z rôznych bezdrôtových, mobilných a pevných sietí. Každý modul má individuálne a špecifické požiadavky na prenosové parametre a popri tom je nevyhnutné minimalizovať energetické nároky tak, aby niektoré moduly mohli na obmedzený zdroj energie fungovať aj niekoľko rokov.

II. Internet Vecí

Internet vecí je reprezentovaný množstvom zariadení, ktoré spolu vzájomne komunikujú a prinášajú dodatočnú funkcionalitu, ktorú by jednotlivé zariadenia samostatne nedokázali priniesť. Dobrým príkladom je napríklad domáca televízia a inteligentný termostat. Zatiaľ čo termostat obsahuje teplotný senzor vďaka ktorému dokáže posielať príkazy na regulovanie teploty, tak nedokáže zobrazovať nejaké podrobnejšie informácie o nameraných teplotách. A práve tu vstupuje idea internetu vecí, ktorá termostatu umožní využiť funkcie iných zariadení v sieti. V tomto konkrétnom prípade môžeme uvažovať napríklad o televízii, ktorá termostatu poskytne funkciu zobrazenia. Používateľ si vďaka tomu môže pozrieť grafické priebehy teploty a výkonu vykurovania alebo chladenia, čo by samostatný termostat nedokázal. Samozrejme aj termostat môže mať grafickú obrazovku, avšak to by zbytočne navyšovalo cenu a komplexnosť takéhoto senzoru, pričom práve myšlienka internetu vecí umožní získať takúto doplnkovú funkcionalitu bez zbytočného navyšovania ceny a komplexnosti riešenia.

Zariadenia internetu vecí musia vzájomne komunikovať, pričom táto komunikácia nemusí byť riešená pomocou jedného typu siete, ale je možné aplikovať heterogénne kapilárne siete. Kapilárne siete vznikajú najmä z dôvodu rozdielnych komunikačných požiadaviek zo strany rôznych zariadení. Zatiaľ čo senzory nepotrebujú vysokú dátovú priepustnosť, tak kamerový systém si nevystačí s úzkym komunikačným pásmom. Každá komunikačná technológia tak prináša rôzne výhody a zároveň nevýhody a vhodným prepojením takýchto technológií je možné dosiahnuť optimálne komunikačné vlastnosti.

III. Kapilárne siete

Bez možnosti zdieľania si informácií sú IoT zariadenia len jednoúčelové bez ďalšej pridanej hodnoty. Pre zabezpečenie vhodného komunikačného prepojenia je nutné použiť kapilárne siete, ktoré pozostávajú z pevných, mobilných a bezdrôtových sietí. Zatiaľ čo smerovanie v pevných a mobilných sieťach je štandardizované, tak pri bezdrôtových sieťach existuje vysoký potenciál pre ďalšie zlepšenia. V tejto práci sa sústreďujeme na infraštruktúrne založené bezdrôtové siete a Ad-Hoc siete.

A. Infraštruktúrny model a Ad-Hoc siete

Väčšina súčasne používaných bezdrôtových sietí je založená na infraštruktúrnom modeli, ktorý obsahuje zariadenia dvoch kategórií: prístupové body a koncové zariadenia. Základnou vlastnosťou takýchto sietí je, že všetky koncové zariadenia sa pripájajú len na zvolený prístupový bod, cez ktorý realizujú všetky svoje dátové prenosy. Komunikáciu moderuje a riadi centrálna autorita, ktorá je reprezentovaná jedným alebo viacerými prístupovými bodmi a tie určujú ktoré koncové zariadenia môžu v danom čase vysielať na zdieľanom komunikačnom kanále. Nevýhodou infraštruktúrnych sietí je vyššia cena na vybudovanie, citlivosť na útoky, ktoré dokážu narušiť alebo úplne znemožniť komunikáciu s centrálnou autoritou, ale aj maximálna dosahovaná komunikačná rýchlosť.

Ad-Hoc sieť pozostáva z rovnocenných zariadení ktoré neobsahujú centrálnu riadiacu autoritu, ale všetky bezdrôtové zariadenia sú rovnocenné a spadajú pod jednu komunikačnú doménu. Každý komunikačný uzol tak dokáže komunikovať so susedným uzlom aj bez použitia prístupového bodu. Toto prináša autonómnosť od bezdrôtovej infraštruktúry, väčšie pokrytie bezdrôtovej domény, menšiu energetickú náročnosť pri použití nízkoenergetických komunikačných technológií ale aj rýchlu konfiguráciu nových modulov. Nespornou výhodou je aj nízka cena, ktorá sa nezvyšuje o cenu prístupových bodov a flexibilita celého modelu, ktorý nie je závislý od niekoľkých modulov v sieti, čo prináša flexibilitu a umožňuje samo-uzdravovanie takejto siete v prípade výpadku určitých komunikačných uzlov.

B. Jednotný model Ad-hoc sietí

Zariadenia komunikujú v rámci svojej bezdrôtovej domény priamo so susednými zariadeniami, avšak v prípade vzdialenejších uzlov je nutné použiť viacskokové (Multi-hop) spojenie a zodpovedajúce smerovacie protokoly. Použitie vhodných smerovacích protokolov výrazne ovplyvňuje parametre vzniknutej Ad-hoc siete a vo veľkej miere určuje maximálnu priepustnosť a oneskorenie v rámci takejto siete. Je mimoriadne dôležité použiť optimálny smerovací protokol, ktorý sprístupní plný potenciál vzniknutej Ad-hoc siete.

Smerovacie protokoly musia byť dostatočne robustné a zároveň aj dostatočne rýchle, tak aby sa dokázali efektívne vysporiadať so stratovosťou datagramov a prípadnej mobilite zariadení. Nemenej dôležitým aspektom je energetická spotreba, ktorá sa zvyšuje s vyššou frekvenciou komunikácie z dôvodu zmeny polohy a smerovacích algoritmov ako aj výpočtových nárokov samotných algoritmov. Jednou z možností redukcie zmien v smerovaní je zväčšenie veľkosti bezdrôtovej domény, avšak táto zmena by sa negatívne prejavila na celkovej komunikačnej kapacite Ad-hoc siete. Súčasným trendom je minimalizácia veľkosti komunikačných domén, čo s narastajúcim počtom zariadení kladie vyššie nároky na návrh a optimalizáciu smerovacích algoritmov. Horná hranica prenosovej kapacity pre Ad-hoc sieť nerastie, avšak existuje niekoľko metód ako túto kapacitu alternatívne navyšovať:

- minimalizovať počet preposielaní v doméne
- použiť rôzne komunikačné technológie (kapilárne siete)
- minimalizovať interferencie
- použiť viacero komunikačných frekvencií (kanálov)
- použiť rôzne polarizované a umiestnené antény
- zmeniť formu komunikácie na lokálnu (umiestniť najčastejšie komunikujúce uzly čo najbližšie k sebe)
- komunikačné brány prepojiť inou vysokorýchlostnou linkou a odbremeniť tak zvyšok siete

C. Smerovanie v Ad-hoc sieťach

V Ad-hoc sieti fungujú všetky uzly ako smerovače a podieľajú sa na budovaní a udržiavaní smerovacích tabuliek. Cieľový uzol môže byť dosiahnuteľný rôznymi trasami a je potrebné zabezpečiť smerovací algoritmus výberu najvhodnejšej cesty podľa rôznych kritérií na prenos. Takéto uzly môžu v čase meniť svoju polohu a tým vstupovať alebo sa strácať z dosahu ostatných uzlov. Zložité smerovacie algoritmy prinášajú zvýšené nároky na prenosový kanál a spotrebu energie.

Najjednoduchším spôsobom šírenia datagramu je záplava (flooding), kde odosielateľ pošle datagram všetkým susedným uzlom. Každý takýto uzol následne prepošle prijatý datagram ďalším susedom a keďže sa jedná o všesmerové vysielanie, tak takýto datagram sa dostane aj k uzlom, ktorý ho už prijali. Datagram nemusí dosiahnuť cieľový uzol v prípade ak je takýto uzol osamostatnený mimo dosahu akéhokoľvek uzla v segmente, alebo nastane problém skrytého terminálu a vzájomnej kolízii viacerých datagramov. Medzi výhody záplavy patrí jednoduchosť a nenáročnosť na hardvérové a softvérové prostriedky uzlov. V prípade nízkej frekvencie posielania datagramov môže byť takéto riešenie aj efektívnejšie ako zložité smerovacie protokoly. Nevýhodou sú extrémne rýchlo rastúce nároky na priepustnosť siete v prípade vysokého počtu uzlov a datagramov.

Smerovacie protokoly pre Ad-hoc siete rozdeľujeme na trojicu hlavných kategórií [1]:

- proaktívne protokoly, ktoré si trvalo udržiavajú aktívnu smerovaciu tabuľku. Do tejto kategórie patria tradičné Link-State a Distance-Vector protokoly
- reaktívne protokoly budujú smerovacie tabuľky len v prípade požiadavky na komunikáciu.
- hybridné, ktoré sú kombináciou predošlých kategórií.

Okrem rozdielu v implementácii sú rôzne kategórie smerovacích protokolov odlišné najmä v dĺžke oneskorenia komunikácie a množstve komunikačných dát nutných pre vybudovanie spojenia. Zatiaľ čo si proaktívne smerovacie protokoly udržujú aktuálnu smerovaciu tabuľku, tak reaktívne protokoly musia pred zahájením komunikácie najskôr vybudovať spojenie pomocou viacerých dopytov na jednotlivé uzly. Tieto dopyty oneskorujú nadviazanie spojenia a generujú ďalšie koordinačné prenosy. Naopak proaktívne protokoly musia udržiavať aktívne spojenia a smerovacie tabuľky, čo je energeticky náročnejšie. Medzi hlavných zástupcov Ad-hoc smerovacích protokolov patrí proaktívny Distance-Vector smerovací protokol RPL (IPv6 Routing Protocol for Low-Power and Lossy Networks) a reaktívny protokol LOAD-ng (Lightweight On-demand Ad-hoc Distance vector routing

protocol - next generation). Pre Ad-hoc siete s vysokým počtom komunikačných uzlov a preposielaných datagramov sú výhodnejšie proaktívne protokoly. Pri priamom porovnaní smerovacích protokolov RPL a LOAD-ng vykazuje RPL výhodnejšie komunikačné vlastnosti aj parametre a preto sme ho vybrali ako najvhodnejší protokol pre zabezpečenie smerovania vo vzniknutej sieti [1].

IV. Vybraný protokolový zásobník heterogénnych sietí

Pri budovaní rozsiahlych heterogénnych sietí pre zariadenia internetu vecí je možné využiť viacero komunikačných technológií a protokolov. Protokolový zásobník musí byť nezávislý od použitej komunikačnej technológie na fyzickej vrstve. Medzi najpoužívanejšie komunikačné technológie fyzickej vrstvy v rámci zariadení internetu vecí patrí IEEE 802.11 (Wi-Fi), IEEE 802.15.1 (Bluetooth) a IEEE 802.15.4. Každá technológia má pritom iný protokol sieťovej vrstvy, pričom prevládajú protokoly štandardizované medzinárodnou organizáciou IETF. Zatiaľ čo v robustnejších (Wi-Fi) sieťach sa používa plnohodnotný protokol IPv6, tak v energeticky efektívnych a pomalších sieťach je využívaný skrátený a rýchlejší protokol 6LoWPAN [2]. Takéto siete sú zväčša založené na Ad-hoc modeli s protokolom RPL, ktorý umožňuje efektívnu komunikáciu medzi viacerými decentralizovanými uzlami. Na vyšších vrstvách môže byť následne použitý protokol TCP alebo UDP a aplikačný protokol CoAP zabezpečujúci interoperabilitu medzi rôznorodými systémami.

A. Smerovací protokol RPL (IPv6 Routing Protocol for LLNs)

RPL je Distance-Vector smerovací protokol fungujúci na sieťovej vrstve protokolového zásobníka komunikujúci cez protokol IPv6 v LLN (6LoWPAN). Pre zabezpečenie smerovania vytvára protokol RPL smerovo orientovaný acyklický graf (Destination Oriented Directed Acyclic Graph DAG) [3]. Graf je vytváraný pomocou funkcie výpočtu metriky (Objective Function OF), ktorá je v základe založená len na počte skokov medzi zdrojovým a cieľovým uzlom. DAG graf minimalizuje vzdialenosť medzi hraničným smerovacím uzlom LBR (LLN Boarder Router) [4] a akýmkoľvek iným uzlom v sieti tak, že popisuje najkratšiu vzdialenosť medzi takýmito uzlami. Hraničný DAG uzol pripojený k IPv6 chrbticovej sieti má v grafe pridelenú metriku s hodnotou 1 pričom okolitým uzlom posiela informácie prostredníctvom DIO informačného rámca (DAG Information Option). Susedné uzly po prijatí DIO rámca následne vypočítajú svoju metriku na základe informácií v DIO a vzdialenosti od susedného uzla od ktorého prijali DIO rámec. Každý uzol si pritom na základe najnižšej metriky z DIO rámcov zvolí suseda, cez ktorého bude smerovať svoju komunikáciu k hraničnému uzlu [5]. Výpočet metriky na základe údajov od susedných uzlov a hodnotou hrán medzi nimi je znázornený v rovniciach (1-4).

$$Metri\ '_i = Metric_j + w_{i,j} \quad (1)$$

$$Metric"_i = Metric_k + w_{i,k} \quad (2)$$

$$Metric'_i < Metric"_i \quad (3)$$

$$Metric_i = \min\{Metrica'_i, Metric"_i\} \quad (4)$$

Pre proaktívne získanie DAG informácií si uzly môžu preposielať aj DIS rámce (DODAG Information Solicitation). Na nasledujúcom obrázku je znázornený proces propagácie DIO rámcov pre sformovanie DAG grafu. Zatiaľ čo prerušenými čiarami sú znázornené komunikačné cesty, tak plná čiara indikuje hrany acyklického DAG grafu a teda aj sformované spoje medzi susednými uzlami.

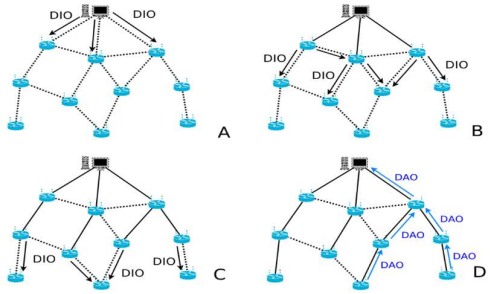


Fig. 1. Šírenie DIO rámcov pre výpočet metriky [6].

DIO rámce sú periodicky preposielané z každého uzla po uplynutí časového intervalu stanoveným minimálnou hodnotou *Imin*. Tento interval sa exponenciálne zvyšuje, čím minimalizuje dátovú výmenu a zahltenie siete po rýchlom sformovaní sieťovej topológie, ktorá trvá len niekoľko kôl výmeny DIO rámcov. V prípade straty komunikácie medzi uzlom a jeho susednou komunikačnou bránou sa aktivuje mechanizmus lokálnej opravy, ktorá zruší platnosť lokálnej subdomény DAG grafu a spustí prepočet novej komunikačnej brány a DAG grafu. Taktiež dôjde k zresetovaniu hodnoty časového intervalu posielania DIO rámcov na hodnotu *Imin*, čím sa urýchli formovanie grafu. Hraničný DAG uzol taktiež preposiela DIO správy, ktoré umožňujú vykonať globálnu opravu v prípade výraznej zmeny topológie.

Uzly po pripojení do DAG grafu alebo po zmene komunikačnej brány prepošlú svoju adresu a všetky dostupné adresy uzlov zo svojej DAG subdomény v DAO rámci (Destination Advertisement Option) [7], ktorý je prešírený až k hraničnému DAG uzlu. Tento mechanizmus slúži na budovanie smerovacej tabuľky hraničného uzla pre smerovanie ku koncovým uzlom.

B. Návrh rozšírenej metriky pre smerovací algoritmus

Výpočet metriky založený na počte skokov (Hop Count) je pre nami zadefinované pokročilé smerovanie na základe aplikačných požiadaviek nedostatočné a preto navrhujeme rozšírenie o ďalšie elementárne metriky, ktoré zodpovedajú rôznorodým komunikačným parametrom v rámci IoT siete. Hodnota opisovaných metrík stúpa so zhoršujúcou sa komunikačnou linkou a preto je pre zabezpečenie komunikácie vybraná linka s najnižšou metrikou. Jednotlivé elementárne metriky môžeme rozdeliť na metriku linky a metriku uzla podľa toho či zodpovedá stavu uzla alebo linky medzi dvoma uzlami. Výpočet navrhovaných elementárnych metrík je znázornený v rovniciach (5-13).

$$\textbf{Hop Count} = HC^{i,j} = w_{i,j} = w(i \oplus \mathrm{j}) \geq 1 \quad (5)$$

$$\textbf{Link Load} = LL^{i,j} = \sum_i^{j-1} \frac{Cap^{i,i+1}}{Cap^{i,i+1} - Load^{i,i+1}} \geq 1 \quad (6)$$

$$\textbf{Link Quality} = LQL^{i,j} = \sum_i^{j-1} \left(\sum_{k=1}^{7} p_k^{i,i+1} * k \right) \geq 1 \quad (7)$$

$$\textbf{Trip Time} = RTT^{i,j} = \sum_i^{j-1} \tau(i, i+1) = t_j - t_i \geq 1 \quad (8)$$

$$\textbf{Secure Level} = STL^{i,j} = \sum_i^j s_i \geq 1; \quad 1 \leq s_i \leq def_{max} \quad (9)$$

$$\textbf{Expected Tx. Count} = ETX^{i,j} = \frac{s+f}{s} \geq 1 \quad (10)$$

$$\textbf{Fwd. Cnt} = EFX^{i,j} = \frac{1}{P_{succ(i,j)}} = \prod_i^j \frac{sf_i + ff_i}{sf_i} \geq 1 \quad (11)$$

$$\textbf{Sleep} = SS^i = \frac{1}{Duty^i{}_{cycle}} \geq 1; \quad 0 \leq Duty^i{}_{cycle} \leq 1 \quad (12)$$

$$\textbf{Remaining Energy} = RE^i = \frac{Volt^i{}_{max}}{Volt^i{}_{curr}} \geq 1 \quad (13)$$

Medzi doplňujúce metriky môžeme zaradiť aj dostupnosť zdroja napájania, použité spektrum, použitú komunikačnú technológiu, či čas poslednej komunikácie. Kombináciou všetkých elementárnych metrík do jednej komplexnej kompozitnej metriky získame presnejšie správanie sa smerovania zodpovedajúce požadovaným komunikačným parametrom. Vplyv elementárnych častí metriky na celkovú kompozitnú metriku $M^{i,j}$ môžeme vypočítať podľa vzorca (14).

$$\alpha, \beta, \gamma, \delta, \varepsilon, \zeta, \eta, \vartheta, \kappa \geq 0 \quad (14)$$

$$M^{i,j} = \alpha * LL^{i,j} + \beta * LQL^{i,j} + \gamma * RTT^{i,j} + \delta * STL^{i,j} + \varepsilon * \\ * ETX^{i,j} + \zeta * EFX^{i,j} + \eta * HC^{i,j} + \vartheta * \sum_i^j SS^i + \kappa * \sum_i^j RE^i$$

Kompozitná metrika tak môže zodpovedať základnému správaniu sa smerovania na základe metriky Hop Count, ale zároveň môže detailnejšie charakterizovať jednotlivé parametre komunikačných liniek. Dôležité je pritom zachovať podmienku, aby pre každý jeden DAG graf boli koeficienty kompozitnej metriky rovnaké pre všetky uzly a z tohto dôvodu zavádzame viacero RPL inštancií. Počet inštancií počas komunikácie je možné dynamicky zvyšovať alebo znižovať podľa aktuálne dostupných výpočtových a pamäťových prostriedkov. Každá aplikácia vyššej vrstvy si tak na základe preferencií komunikačných parametrov vyberie alebo vytvorí nový konkrétny smerovací profil zodpovedajúci jej požiadavkám. Keďže počet takýchto profilov je obmedzený z dôvodu zachovania nízkych výpočtových nárokov, tak podľa kategórie jednotlivých zariadení môže protokol pracovať v dvoch režimoch: ukladací a transparentný (Storing a Non-Storing). V transparentnom (Non-Storing) režime si uzly neukladajú žiadne smerovacie informácie okrem údajov o susednom uzle cez ktorý komunikujú s hraničným DAG smerovačom. Tento režim umožňuje efektívnu komunikáciu medzi koncovými uzlami a hraničným uzlom, avšak horizontálna komunikácia medzi jednotlivými koncovými uzlami musí byť vždy smerovaná cez hraničný uzol, čo zvyšuje zahltenie siete. V ukladacom (Storing) režime si všetky uzly ukladajú smerovacie informácie o ostatných uzloch zo svojej DAG subdomény, čím urýchľujú komunikáciu medzi koncovými uzlami v rámci jednej subdomény DAG grafu. Nevýhodou takéhoto riešenia je vysoká náročnosť na pamäťové a výpočtové prostriedky, ktoré lineárne narastajú s počtom uzlov v DAG subdoméne a počtom použitých profilov. Pre siete pozostávajúce z obrovského počtu uzlov s obmedzenými prostriedkami by nebolo možné použiť ukladací režim a z tohto dôvodu sa používa agregácia IPv6 adries a aj nové hybridné režimy MERPL [8], ktoré kombinujú výhody oboch riešení. Vhodnou kombináciou hybridného režimu MERPL a navrhovanej kompozitnej metriky je možné dosiahnuť presnejšie správanie smerovacieho algoritmu pri minimálnom náraste výpočtových nárokov. Komunikácia v rámci takto vzniknutej sieti je schopná detailnejšie rozlišovať medzi častokrát protichodnými sieťovými požiadavkami rôznych aplikácií a zabezpečiť optimálne smerovacie parametre bez výrazných zásahov do existujúcej sieťovej infraštruktúry.

V. Zhodnotenie

Súčasné smerovacie algoritmy sa sústreďujú len na jednu komunikačnú technológiu s jednoduchou metrikou. Použitím kapilárnych sietí sa tieto protokoly stávajú zastaranými. V rámci práce sme navrhli niekoľko nových metód na tvorbu optimálnej kompozitnej metriky pre smerovanie v kapilárnych sieťach. Aplikáciou kompozitnej metriky dokážu IoT zariadenia efektívnejšie využívať dostupné prostriedky a zohľadňovať jednotlivé komunikačné parametre tak, aby čo najviac zodpovedali rôznorodým a častokrát aj protichodným požiadavkám aplikácií. Implementácia takéhoto smerovacieho algoritmu v kapilárnych sieťach aktívne prispeje k zlepšeniu a akcelerácii komunikácie medzi IoT zariadeniami.

Poďakovanie

Autori ďakujú za finančnú podporu v rámci projektu Vega 1/0616/14.

Referencie

[1] Radoi, I.E., Shenoy, A., Arvind, D.K.: Evaluation of Routing Protocols for Internet-Enabled Wireless Sensor Networks. ICWMC 2012. 978-1-61208-203-5. Copyright 2012 IARIA.

[2] Mulligan, G.: The 6LoWPAN Architecture. 6LoWPAN Working Group, IETF. EmNets2007, 25-26.6.2007, Cork, Ireland. Copyright 2007 ACM.

[3] Yunis, J.P., Dujovne, D.: Energy efficient routing performance evaluation for LLNs using combined metrics. 978−1−4799−4269−5/14, Copyright 2014 IEEE.

[4] Vasseur, J.P., Agarwal, N., Hui, J., Shelby, Z., Bertrand, P., Chauvenet, C.: RPL: The IP routing protocol designed for low power and lossy networks. Internet Protocol for Smart Objects (IPSO) Alliance. 2011.

[5] Karkazis, P., Leligou, H.C., et al.: Design of primary and composite routing metrics for RPL-compliant Wireless Sensor Networks. 2012.

[6] Tripathi, J., Oliveira, J.C.: Proactive versus Reactive Revisited: IPv6 Routing for Low Power Lossy Networks. 978-1-4673-5239-0/13. IEEE.

[7] Tsvetkov, T.: RPL: IPv6 Routing Protocol for Low Power and Lossy Networks. Seminar SN SS2011, 10.2313/NET-2011-07-1_09, July 2011.

[8] Gan, W., Shi, Z., Zhang, Ch., Sun, L., Ionescu, D.: MERPL: A More Memory-efficient Storing Mode in RPL. ICON 2013. 978-1-4799-2084-6/13, Copyright 2013 IEEE.

Configurable Reprogramming Scheme for Over-the-Air Updates in Networked Embedded Systems

Ondrej Kachman
2nd year, full-time study
Supervisor: Ladislav Hluchý, Consultant: Marcel Baláž

Institute of Informatics, Slovak Academy of Sciences
Dúbravská cesta 9, Bratislava, Slovak Republic
ondrej.kachman@savba.sk

***Abstract*—Networked embedded systems are nowadays used in various applications and the number of devices used in such systems grows by millions each year. Physically inaccessible, resource constrained low-power devices sometimes require remote over-the-air reprogramming. For the last 20 years, many reprogramming schemes have been developed and built upon. These schemes are required to be fast and energy effective. The amount of operations executed on the target devices should be minimal, delta files shared on the network should be very small. This paper analyzes existing solutions and proposes a new, configurable reprogramming scheme, that can provide more control over the process of the device reprogramming.**

Keywords—networked embedded system, over-the air update, reprogramming, low-power device

I. Introduction

Recent advances in the internet-of-things technologies enabled fast development of various systems of collaborating computational devices, also called networked embedded systems. Wireless sensor networks (WSNs) are the first example of such systems. WSNs are used to collect various data from their environment and mostly consist of low-power devices. These systems evolved into cyber-physical systems (CPS) in the last 10 years. These systems collect and process data, then take actions based on the results of processed information. CPS structure usually includes small low-power devices as sensors and actuators, various networking devices for networking and powerful computers for data processing [1].

A. Low-power devices and their reprogramming

This paper is focused on the low-power devices often found in the described systems. These devices create an interface between physical and cyber world. The amount of the low-power devices in CPS or WSNs may vary, but it can often reach hundreds. Some of the devices may not be physically accessible after their deployment. These devices are battery powered and communicate through wireless network interface. They are expected to run for months or years after their deployment. Firmware of the devices developed under the test conditions may malfunction in the real environment. In that case, reconfiguration is required [2]. If simple reconfiguration of some parameters does not help, firmware requires reprogramming. To prevent the waste of energy, update data shared on the network and the number of operations executed on the target devices must be minimal. This is the main reason for the development of energy efficient reprogramming schemes for low-power devices.

B. Related work

The fist solutions developed in early 2000's were loading the full firmware image onto the target device, then using bootloader to replace the old firmware version. Incremental reprogramming, loading many firmware versions one after another, resulted in rapid battery depletion. This triggered the development of block based reprogramming schemes. These schemes found updated blocks of the firmware, then sent these blocks to the target devices reducing amount of the data shared on the network and also the number of written memory cells. Latest reprogramming schemes use byte based differencing, providing the best results.

1) Remote incremental linking [3]*:* In 2005, remote incremental linking scheme was proposed. This scheme introduced slop regions – a free space in the program memory between the firmware functions. This approach enabled functions to grow and shrink in their slop regions, reducing the shifts and relocation changes in the firmware, making it possible to generate smaller delta files.

2) Zephyr [4]*:* Another roprogramming scheme proposed function indirection table to reduce the impact of function shifts on the delta generation. This approach created a table that pointed to all functions and the firmware would call functions through this table. However, this table only handled `call` instructions and not relative calls and jumps. Jumping to and from the table also resulted in worse execution time of a firmware.

3) Hermes [5]*:* Built over Zephyr, this scheme allocates fixed addresses for variables, further ruducing the delta files size. This approach scans through the source files before the compiler is invoked and puts initialized and uninitialized variables into assigned structures, preserving their order when

the compiler generates `.bss` (for uninitialized variables) and `.data` (for initialized variables) sections.

4) R3 reprogramming scheme [6]*:* This approach is focused on the relocatable code and object files. The object files for many low-power platforms have standard executable and linkable format (ELF). The files generated by compiler have relocatable format. Reference instructions in `.text` sections need their addresses resolved by linker. R3 sets these relocatable entries to zero, generates small metadata for a loader located on the target device, that resolves relocations during boot. Authors demonstrated improvement in delta size over existing solutions.

5) Q-diff scheme [7]*:* This scheme is also focused on the object files. It also takes advantage of slop regions and improves the technique with possibility of placing a function and its slop region into a different part of the memory. This scheme changes layout of .bss and .data sections, requiring changes to instructions with indirect addressing of variables. To prevent changes to relative jumps, Q-diff adds new code blocks to the end of the memory, then points `call` instructions to those blocks. This may add more execution time to the firmware. Solution claims to be platform independent, however, authors do not address the problem of platform specific relocation types and how they identify relative or fixed reference instructions. One of the main benefits of Q-diff is that the update process does not require external memory and reboot to finish successfuly.

II. Definition of a problem

Previous chapter provides some insight into some of the most relevant reprogramming schemes for constrained low-power devices. This chapter sums up the state-of-the art and describes, what can be done to improve the existing solutions.

A. Compilers, linkers and object files summary

Most of the analyzed literature avoids direct changes to compilers or linkers. This is very important, as the manufacturers of the devices usually provide compiler and linker for their platforms that is able to generate the most optimized code. There are approaches that altered registers allocated during compilation, but they worsened firmware execution time after every update.

The best solution is to work with the object files. These files have standard ELF format and can be examined in their relocatable form, before linking, or in their executable form, after linking. By examining the relocatable files, linking process can be managed better.

B. Platform independency summary

Even solutions that claim to be platform independent cannot be fully independent. The authors of such solutions may have focused on the same family of devices only, or avoided explanation of their definition of platform independent.

Every microcontroller family uses different relocation types. In order to be able to work with relocations, we must obtain the definitions of relocation types for chosen platforms. Every relocation type is calculated differently, some relocations have the same value at every memory position (`call` instructions, `load` from and `store` to RAM), some may change their value at different address (`rjmp` and `rcall` instructions).

Furthermore, if the software alters the source code directly in the object files, it must include the complete instruction set of a chosen platform, making it even more platform specific. This is the case of Q-diff approach, that changes indirect instructions. To sum up, no solution can be fully platform independent.

C. Memory fragmentation summary

Slop regions fragment program memory. Different approaches argue that it is a waste of space and inefficient use of program space. There are also some speculations, that fragmented memory consumes more energy.

If the linker is configured to provide slop regions to functions, it places them to the different parts of the program memory, resolves relocations correctly and generates executable file. No additional instructions are generated and the code is optimized by the compiler in the previous stage. As the most low-power devices currently use NAND flash memory for their firmware, the access time is the same. The point of slop regions is to use the program space to the full potential and enable to generate smaller deltas. It is not a waste of space nor inefficient.

We carried out some experiments and evaluated energy consumption of a fragmented firmware. These experiments and their results are described in the following chapter.

D. Proposed solution

We propose configurable reprogramming scheme. Various methods have their own advantages and for different reprogramming strategies, different solutions may perform better. If there are not many incremental updates, there is no need to make unnecessary changes to function or variable placements. Requirements for our scheme:

- Enable memory fragmentation and defragmentation. Fragmented memory with slop regions is better for frequent incremental updates with small deltas. Defragmented memory is the default state and has slightly better energy consumption.
- Do not alter the source code. The source code is optimized for the best performance by the compiler, added instructions cause worse execution time. The reprogramming scheme also does not need to include processor specific instruction sets.
- Take advantage of the relocatable entries. These can be read from the object files before linking. Their final address can be found out from the executable file. These entries can be used for memory fragmentation and can be stored on the target device, consuming memory but enabling small deltas, or they can be sent to the device saving memory but using larger deltas.
- Enable updates that do not require external memory of a device. Apply the update on-the-fly and do not reboot

the device. Only clear the stack and registers if necessary.
- The solution must enable differencing algorithms to generate as small delta files as possible to prevent network congestion and overall waste of energy on the network communication.

III. Using Relocatable Code for Memory Fragmentation

This chapter describes, how will our solution use relocatable entries in object files to fragment and defragment program memory. First, we perform and experiment, that measures energy consumption of a chosen device with defragmented and fragmented memory. Then, we describe how relocations are resolved and which relocations have to be altered in a function when it is shifted.

A. Energy consumption of a fragmented memory

We perform an experiment on an ATmega32u4 microcontroller with 32KB of NAND flash program memory. We base this experiment on the energy consumption estimation model for NAND flash memories [8]. The model suggests that the energy consumed during activation of a memory cell depends on how far from the previously activated region the cell is. The memory regions are created by 2^k bytes and activation of each region consumes E_k energy. Formal representation:

$$i \to j = \sum_{k=0}^{N(i,j)} E_k \qquad (1)$$

i and j represent memory address. Term N(i,j) represents the largest changed region – $2^{N(i,j)}$ bytes.

$$N(i,j) = \lfloor \log_2 (i \oplus j) \rfloor \qquad (2)$$

The energy E_k varies. The most energy is consumed by activation of a different page. Activation of cells within the same page does not consume as much energy.

The test firmwares consisted of two or five jumps between the different pages of memory. The microcontroller was powered by a stable source with the voltage of 5V. We observed current consumption of each test scenario. The results of experiments are in the Table 1.

TABLE I. AVERAGE CURRENT CONSUMPTION FOR FRAGMENTED MEMORY

Firmware	Max. activated regions	Average current (mA)	Description
1	2	28,61	Loop on a single page
2	7	29,80	Loop on a two consecutive pages
3	11	30,06	Loop on a two shifted consecutive pages
4	14	30,20	2 pages, 16KB jump
5	15	30,28	2 pages, 32KB jump
6	14	30,27	5 jumps throughout the whole 32KB

We observe the increase in the current when we need to activate more regions for jumps. The worst case scenario is 5% worse than the best case scenario. For expected lifetime of 5 years, this could prolong the device's battery life by 3 months. However, it is not possible to keep the whole firmware on a single page. Second scenario, with 2 consecutive pages, saves only 1,4% - 2 weeks out of 5 years. This is more realistic. This means that fragmented memory does require more energy, but not unacceptably more. Temporary fragmentation of a memory should not deplete the battery too fast and it can help to generate smaller delta files and save energy on a network communication.

B. Shifting functions in the program memory

This is currently in a development stage and has not been tested yet. For each function that is shifted, we must change relative instructions within this function and also all relative instructions pointing to this function. Fig. 1. illustrates a simple scenario with a shift of a function_2. Two relative jumps must be changed in the firmware in order for it to work correctly, a relative call to itself does not have to be changed.

To find the relocations and their values, we must explore both relocatable and executable object files:
- For every relocatable file, list all the `.text` sections with relocatable entries and store their offsets, sizes and types.
- Resolve the final addresses of `.text` sections from the executable file. These addresses are assigned during linking.
- Using the offsets of relocatable entries, find all final values of the generated relocations.

Now, when the function is to be shifted, we know exactly which relocations must be altered. Note, that we do not add any additional instructions to the code, only alter the existing relative instructions (and possibly some calls to the shifted function).

Currently, the tool that extracts all relocatable entries from the object files is complete. It requires platform specific relocation types to determine which relocations are relative. We have yet to program an update agent responsible for function shifts on the target devices.

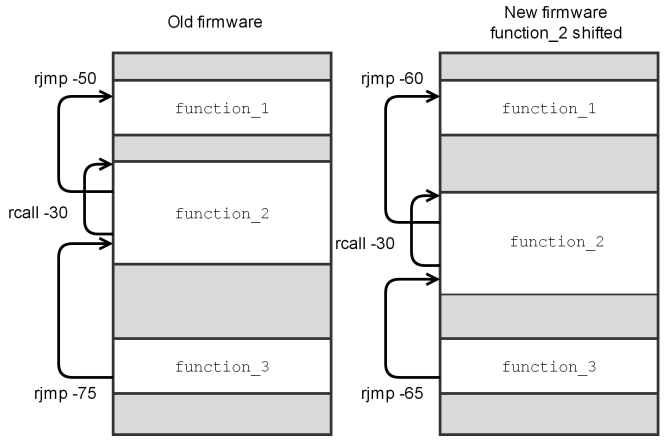


Fig. 1. Changes to some relocatable entries after a function shift

IV. Generating delta files

The proposed technique will grant the programmers full control over the layout of a firmware. It will also be possible to store the table with relocations on the device. Providing single functions or whole modules with slop regions aims to generate small deltas for frequent, incremental updates. Differencing algorithms compare firmware images and encode data shifts as COPY operations and new data as ADD operations.

We developed a differencing algorithm that can update a device without use of an external memory, does not require reboot and does not use RAM memory, it only requires 4 generic purpose registers [9]. The algorithm is called Delta Generator and performed better than R3diff [6] (up to 19%) in 6 out of 7 firmware change cases.

Table 2 shows the sizes of the delta files generated by two differencing algorithms – R3diff and Delta Generator. R3diff sets all relocations to zero, Delta Generator uses slop regions. Column 'Changed' shows the amount of bytes that changed between the old and the new firmware. Once we improve our scheme to fully handle relocations and function shifts, delta files of Delta Generator should be even smaller.

TABLE II. DELTA FILE SIZES GENERATED BY THE DIFFERENCING ALGORITHMS

Firmware change case	Changed (bytes)	R3diff [6] Delta (bytes)	Delta Generator [9] Delta (bytes)
1	2	15	12
2	2668	966	954
3	1222	100	106
4	3054	1522	1448
5	3150	2051	1986
6	648	1193	970
7	3136	2057	1990

V. Aims of the dissertation thesis, conclusion

This chapter lists aims of the dissertation thesis and provides short commentary on how they will be accomplished. The aims are listed in the following subchapters.

A. *Definition of parameters that influence performance and energy consumption of over-the-air firmware updates*

The thesis will provide in-depth analysis of the chosen problem. It will list all known challenges in this area along with their solutions. Some of these problems have been mentioned throughout this paper.

B. *Proposal of an energy consumption estimation model for over-the-air updates of low-power devices*

Energy consumption estimation models can help evaluate any reprogramming scheme. With the more possible configurations of the firmware, these models can help choose the most effective update strategy. We published paper that describes how these models can help evaluate the energy efficiency of the reprogramming schemes – [10].

C. *Design of a reprogramming scheme for fast and energy effective over-the-air updates of low-power devices*

We developed the differencing algorithm that generates small delta files. We also developed an update agent for target devices that does not use external memory, RAM memory and does not require reboot. We are currently working with relocatable code to give developers more control over the firmware layout, thus make proposed reprogramming scheme configurable.

D. *Implementation of a proposed scheme and its evaluation on a chosen hardware platforms*

Once the reprogramming scheme is complete, we will evaluate it on the ATmega microcontroller family, the MSP430 microcontroller family, and one other platform that has not been chosen yet. We aim at the low-power devices often used as sensors or actuators.

E. *Conclusion*

Our work is showing promise, but the real challenge is making our firmware reprogramming scheme more configurable. We published 2 papers from the completed work. Solution proposed in this paper is yet to be implemented on the target platforms and evaluated. This work has been supported by Slovak national project VEGA 2/0192/15.

VI. References

[1] F.-J. Wu, Y.-F. Kao and Y.-C. Tseng, "From wireless sensor networks towards cyber-physical systems," in *Pervasive and Mobile Computing*, vol. 7, Elsevier B.V., 2011, pp. 397-413.

[2] J. Shi, J. Wan, H. Yan and H. Suo, "A Survey of Cyber-Physical Systems," in *International Conference on Wireless Communications and Signal Processing (WCSP)*, Nanjing, 2011.

[3] J. Koshy and R. Pandey, "Remote Incremental Linking for Energy-Efficient Reprogramming fo Sensor Networks," in *Proceeedings of the Second European Workshop on Wireless Sensor Networks*, 2005.

[4] R. K. Panta, S. Bagchi and S. P. Midkiff, "Efficient incremental code update for sensor networks," *ACM Transactions on Sensor Networks (TOSN),* vol. 7, no. 4, February 2011.

[5] R. K. Panta and S. Bagchi, "Hermes: Fast and Energy Efficient Incremental Code Updates for Wireless Sensor Networks," in *IEEE INFOCOM 2009*, Rio de Janeiro, 2009.

[6] W. Dong, B. Mo, C. Huang, Y. Liu and C. Chen, "R3: Optimizing relocatable code for efficient reprogramming in networked embedded systems," in *IEEE INFOCOM Proceedings*, Turin, IEEE, 2013, pp. 315 - 319.

[7] N. B. Shafi, K. Ali and H. S. Hassanein, "No-reboot and Zero-Flash Over-the-air Programming for Wireless Sensor Networks," in *9th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, Seoul, 2012.

[8] J. Pallister, K. Eder, S. J. Hollis and J. Bennett, "A high-level model of embedded flash energy consumption," in *International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES)*, Jaypee Greens, 2014.

[9] O. Kachman and M. Balaz, "Optimized Differencing Algorithm for Firmware Updates of Low-Power Devices," in *19th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems*, Kosice, 2016.

[10] O. Kachman and M. Balaz, "Effective Over-the-Air Reprogramming for Low-Power Devices in Cyber-Physical Systems," in *Technological Innovation for Cyber-Physical Systems*, Lisbon, 2016.

High Performance Computing on Low Power Devices

Vojtech Nikl
2nd year, full-time study
Supervisor Jiri Jaros

Brno University of Technology, Faculty of Information Technology
Bozetechova 2, Brno, Czech Republic
inikl@fit.vutbr.cz

***Abstract*—Nowadays, the power efficiency of modern processors is becoming more and more important apart from the overall performance itself. Many programming tasks and problems do not scale very well with higher number of cores due to being memory or communication bound, therefore, it is often not beneficial to use faster chips to achieve better runtimes. In this case, employing slower low-power processors or accelerators may be more efficient, and possibly get the same results using much less energy. The dynamic runtime adjustments applied to the system based on the properties of a given algorithm, such as frequency and voltage scaling or switching off unneeded parts, may further enhance power efficiency. This paper describes the benefits of using low power chips for building an HPC cluster, the group of algorithms where this approach can be useful, possible system adjustments towards better power efficiency, results achieved so far, and future plans.**

***Keywords*—HPC, parallelism, low power, processor architecture, supercomputers, k-Wave, MPI, OpenMP, performance evaluation, numerical methods, clocking**

I. Introduction and Motivation

Even though computer processors have come a long way in terms of performance, there are still many tasks and problems which require large amounts of computing power to be successfully solved. For some time, hardware engineers haven't been purely focusing on raw performance, but the energy consumption has also become a very important factor. The use of low power processors can be much more efficient for certain kinds of algorithms, mainly for memory and communication bound problems. Unfortunately, this is where algorithms' scalability come into play. Underclocking processsor cores or switching them off during these computationally non-intensive phases may bring further energy benefits.

Today's supercomputers are usually based on the x86 architecture, specifically the Intel Sandy Bridge or newer. One of many examples is the Anselm cluster[1], located in Ostrava, Czech Republic. It consists of 209 2×8-core Intel Xeon E5-2665 2.6 GHz nodes, each with atleast 64 GB of RAM. Each node requires approximately 230 W of power under full load, while providing about 400 GFlop/s of theoretical performance in 64-bit double precision. Most of that energy is dissipated into heat and therefore requires a very intensive and expensive cooling system. Some systems chose a little bit different approach towards higher power effectiveness. One of them is the Fermi cluster[2], located at the CINECA consortium, Bologna, Italy. This cluster, on the other hand, consists of 10,240 nodes, each integrating 16-core IBM PowerA2 1.6GHz processor. While the overall performance per node is about half of the Anselm one's, the peak power consumption is 4 times lower. This results in almost twice as good performance per Watt ratio. *The Green 500 list*[3] provides a ranking of the most energy-efficient supercomputers in the world and Fermi is close to the top at the 59th place, having over 2 GFlop/s per Watt. The most efficient supercomputer has about 7 GFlop/s per Watt (January 2016). Current estimates indicate that processor efficiencies will have to evolve from the current 5 GFlop/s per Watt to 50 GFlop/s per Watt for exascale machines to be viable [1], mainly because a realistic power budget for an exascale system is 20 MW.

The Mont-Blanc project[4] [2][3] is aiming to design a new type of computer architecture capable of setting future global HPC standards, built from energy-efficient solutions used in embedded and mobile devices. The project is run by the Barcelona Supercomputing Center[5] and is funded mainly by the European Commision. The main focuses of development are the *OmpSs* parallel programming model to automatically exploit multiple cluster nodes, transparent application check pointing for fault tolerance, support for ARMv8 64-bit processors, and the initial design of the Mont-Blanc Exascale architecture. The main goal is to design a new high-end HPC platform that is able to deliver a new level of performance/energy ratio when executing real applications that should provide exascale performance using 15 to 30 times less energy.

Unfortunately, the approach of using ARM based kits has a few downsides. Since the low power processors are generally less powerful, it is necessary to employ much more of them

[1]https://docs.it4i.cz/anselm-cluster-documentation/hardware-overview

[2]CINECA consortium, IT, http://www.hpc.cineca.it/content/ibm-fermi-user-guide

[3]http://www.green500.org/

[4]https://www.montblanc-project.eu/

[5]https://www.bsc.es/

to reach the same level of the overall system performance, however algorithms may have troubles scaling that high.

Another problem may be the amount of system memory where ARM based kits only offer 1–4 GB, which can quickly become limiting for extensive simulations. For communication-intensive algorithms, the mostly equiped 1 Gbit/s network cards are also going to be quite limiting, although not as much as it might seem due to lower performance of the chips. Another important reason to focus more on power effectiveness is the resource allocation policy of supercomputing centers. Currently, resources are distributed among users based on *core-hours*, however in the future, users will most likely be billed based on consumed kWhs, which is going to put much more pressure on algorithm efficiency.

II. Power Consumption Analysis

In this section, frequency scaling benchmarks on x86 Intel Xeon Haswell architecture are presented, as the first step of understanding algorithm's power consumption. The purpose of these tests is to show the theoretical performance impact on both compute and memory-bound problems, the scaling of the power consumption of both the processor and memory modules in relationship to performance, define the power efficiency of different benchmarks in relation to the frequency scaling and make a conclusion about suitability of these tests in regards to low power architectures.

The tested system configuration is summarized in Table I.

TABLE I: System hardware overview.

Server	Supermicro 7048GR-TR
Motherboard	Supermicro X10DRG-Q
Processor	2×6-core Intel Xeon E5-2620v3
RAM	DDR4-2133 64GB (4 channels)
SSD	Crucial 250GB

The operating system is *Ubuntu 14.04* with *3.19.0-51-generic* kernel version.

The energy measurements were taken using the PAPI library[6] and its RAPL framework [4], which can directly access the hardware counters of the CPU. PAPI measures the energy consumption of three main components of each CPU - *package*, *powerplane* and *dram*. Package measures the whole socket including the memory controller, powerplane measures only the cores themselves and dram measures the corresponding dram module. The powerplane measurements were not supported on our system (always returned 0 Joules), so only packages and drams were taken into account.

Our Haswell CPU supports frequencies ranging from 1.2 to 2.4 GHz, excluding Turbo mode. To be able to manually set a chosen frequency, the Intel driver had to be replaced with the ACPI driver and the governor was set from *balanced* to *userspace* using the system's `cpupower` utility.

All the benchmarks were compiled using the `GNU GCC 5.3.1` compiler. The flags used were

```
gcc -std=c99 -O3 -mavx -ffast-math
```

[6]http://icl.cs.utk.edu/papi/

Three main frequencies for all the cores were chosen to be benchmarked, 1.2, 1.8 and 2.4 GHz. The turbo was turned off. Voltages for all frequencies were set automatically based on the default CPU stepping provided by Intel[7].

The total energy consumed by each benchmark was calculated by adding package0, package1, dram0 and dram1 power consumptions up.

The first set of tests focuses on memory subsystem performance, using the `lmbench` [5] tool. The lmbench memory bandwidth (see Fig. 1), running one thread per core, shows that while all levels of CPU cache scale almost linearly with the cpu frequency, the main memory bandwidth is affected very little, not more than 5%.

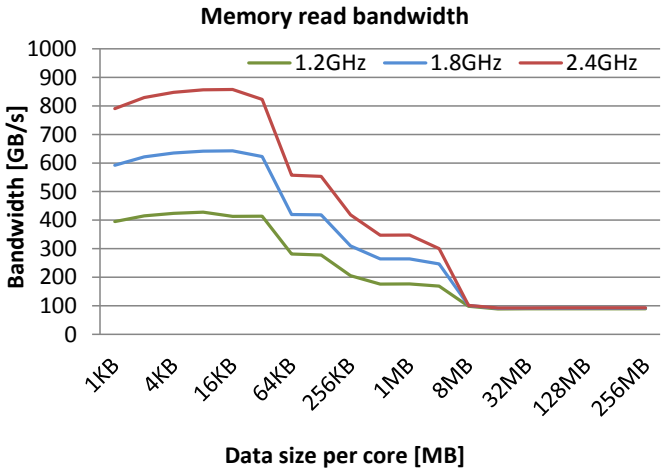


Fig. 1: Memory read bandwidth benchmark using lmbench.

When combined with the energy measurements, Fig. 2 presents the amount of GBs transfered per Watt during the memory read bandwidth benchmark. When the data fits into caches, 25% frequency drop results in 10–30% increased GBs per Watt ratio, in the main memory the increase is only 4–5%.

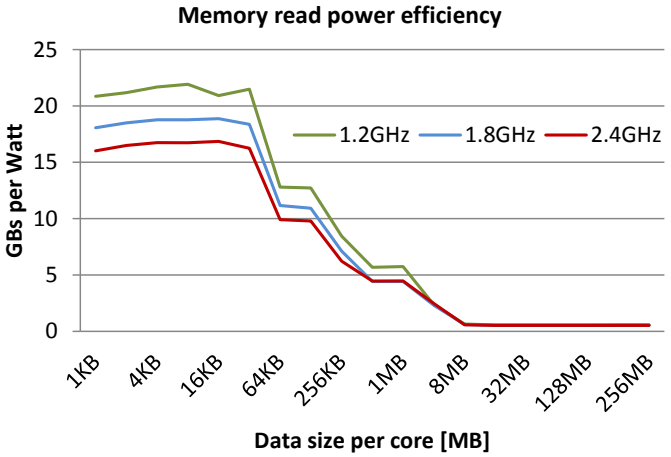


Fig. 2: Amount of data transfered from memory per one Watt.

[7]http://ark.intel.com/products/83352/Intel-Xeon-Processor-E5-2620-v3-15M-Cache-2_40-GHz

TABLE II: Linpack (size 25 000, leading dimension 25 000, 1 KB alingment).

	Package0 [W]	Package1 [W]	Dram0 [W]	Dram1 [W]	Gflop/s	GFlops/W
1.2GHz idle	16.1	16.4	14.0	9.7		
1.2GHz burn	37.2	37.0	35.1	29.2	201.2	2.44
1.8GHz idle	17.0	17.0	14.0	9.7		
1.8GHz burn	48.6	46.7	36.1	30.1	250.6	2.39
2.4GHz idle	17.1	17.1	14.2	9.8		
2.4GHz burn	73.5	57.0	41.6	31.2	299.7	2.07

TABLE III: Power consumption of one k-Wave simulation, 576^3, 10 timesteps (12 threads).

	Total time [s]	Simulation time [s]	Total energy [J]	Simulation energy [J]
1.2GHz	96.21	44.08	10135	5636
1.8GHz	82.56	38.88	9435	5023
2.4GHz	67.97	32.1	9636	5113

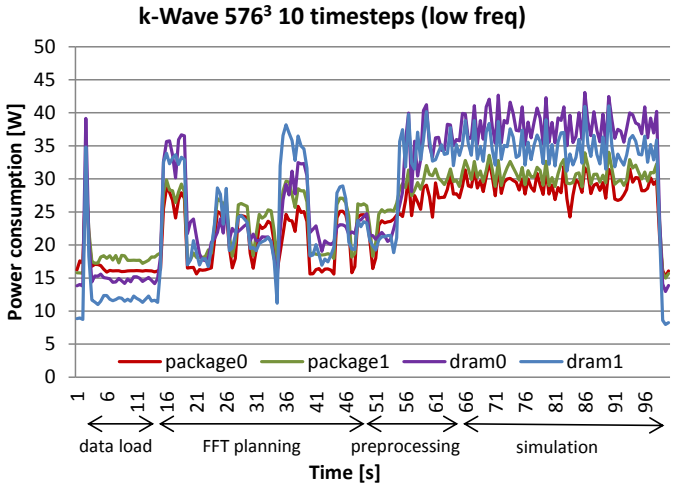


Fig. 3: k-Wave simulation, 576^3, 10 timesteps (1.2 GHz, 12 threads)

While previous benchmarks focused on the memory, the Linpack benchmark [6] focuses on the raw cpu performance. Linpack solves a set of equations based on factorization with $O(n^3)$ operation complexity.

Table II shows the average power consumption during a single run (problem size is 25 000, leading dimension is 25 000 and memory alingment is set to 1 KB), overall performance in GFlop/s and power efficiency in GFlop/s per Watt. The best efficiency is achieved running on low and middle frequency, while the top frequency is slightly behind.

As a final practical application, the OpenMP version of k-Wave [7] was benchmarked. In [8] and [9], the scalability of Fast Fourier transforms (FFT) and whole k-Wave simulations, which implement these FFTs, was shown to reach 16 384 and 8 192 cores, respectively, with over 50% efficiency. In table III, the total time and consumed energy of one k-Wave simulation of size 576^3 with 10 timesteps is measured. Fig. 3 shows the energy consumption evolution over time of different parts of the system during the whole simulation. While the frequency drop from 2.4 GHz to 1.8 GHz slightly improved the overall power-efficiency, further drop to 1.2 GHz worsened it. This is mainly because the dram consumption starts to dominate and the cpu runs too slowly and stalls the simulation. While previous benchmarks showed higher effectivity with lower frequencies, k-Wave's scaling ends when the cpu consumption drops below the dram consumption.

Overall these tests showed, that lowering the frequency of the cpu can bring significant improvements in performance-to-power ratio, mainly for tasks that are memory-bound and cannot benefit as much from fast CPUs, and are therefore, more suitable for low power architectures.

III. Goals of the PHD thesis

Goals

Show that certain classes of extreme-scaling algorithms used in HPC, mainly

- memory-bound,
- interconnect network-bound and
- I/O-bound

can objectively benefit from the use of low power architectures and **dynamic power consumption optimization techniques** in terms of

- total power consumed during the computation,
- lower cooling requirements, less expensive infrastructure at the massive deployment,
- total financial expenses,

while other important factors, such as performance, programming complexity or reliability are affected very little or not at all, compared to the current HPC clusters.

Achieving the goals

- Runtime algorithm analysis and profiling
  Analyze given algorithms and their runtime behaviour (power consumption, performance, level of utilization of different parts of the system, network communication,...) using hardware counters (either available from the OS or provided by JTAG hardware debuggers for development kits), profiling tools (Allinea, perf, ARMv7-A profiler,...), etc. Based on the data obtained, a model describing the behaviour of a given algorithm and hardware setup is automatically created. The goal is to locate

performance bottle-necks of specific parts of the system and find the optimal hardware adjustment. This step will be performed manually for now, but is planned to be almost completelly automated based on the tools used for analysis.

- Dynamic on-the-fly optimizations
  Runtime automated measurements of performance (memory bandwidth, stall cycles, utilization of cores,...) and power consumption, making power optimizations based on a decision model, such as dynamic on/off thread switching, over or underclocking of specific cores, etc. These events will trigger the optimal hardware adjustments found in the previous step. This can be described as an optimization problem and appropriate techniques (fuzzy logic and neural networks) will be used to find the optimal settings.
- Network communication optimizations
  Automated runtime decisions whether it is beneficial to switch off or underclock parts of the system during intensive MPI communications, based on message sizes and quantities, considering the latencies of such operations. This step is can be understood as a part of the first step, but is highlighted separately, because MPI communications are by far considered the slowest way of exchanging data in HPC clusters and low power systems are more likely to suffer from slow network cards, which for example on the ARM technology achieve 1–10 Gbps.

Future steps

Algorithm analysis briefly described in this paper runs on the Intel Haswell architecture. The frequency scaling showed only small improvements in terms of energy demands. Next step is to move to ARM based kits, namely nVidia Tegra or Samsung Odroid. A small cluster, consisting of 4 of these kits, will be built and run under a linux operating system and will serve as the main experimental platform for power consumption analysis.

IV. Conclusion

This paper described the motivation behind the suitability and the use of low power architectures for solving specific tasks, mainly the ones that are memory and/or communication bound, instead of the common architectures used today, mainly for overall power-efficiency and the use as basic building blocks for future exascale clusters. Power consumption analysis was presented on the Haswell architecture, which showed roughly 5% energy savings for 30% reduced frequency. Next step is to move to ARM-based kits, namely Samsung Odroid and nVidia Tegra.

Acknowledgment

This work was supported by the FIT-S-14-2297 Architectures of Parallel and Embedded Computer Systems project.

References

[1] K. Bergman, S. Borkar, D. Campbell, W. Carlson, W. Dally, M. Denneau, P. Franzon, W. Harrod, J. Hiller, S. Karp, S. Keckler, D. Klein, R. Lucas, M. Richards, A. Scarpelli, S. Scott, A. Snavely, T. Sterling, R. S. Williams, K. Yelick, K. Bergman, S. Borkar, D. Campbell, W. Carlson, W. Dally, M. Denneau, P. Franzon, W. Harrod, J. Hiller, S. Keckler, D. Klein, P. Kogge, R. S. Williams, and K. Yelick, "Exascale computing study: Technology challenges in achieving exascale systems peter kogge, editor & study lead," 2008.

[2] N. Rajovic, P. M. Carpenter, I. Gelado, N. Puzovic, A. Ramirez, and M. Valero, "Supercomputing with commodity cpus: are mobile socs ready for hpc?" in *High Performance Computing, Networking, Storage and Analysis (SC), 2013 International Conference for*. IEEE, 2013, pp. 1–12.

[3] N. Rajovic, N. Puzovic, L. Vilanova, C. Villavieja, and A. Ramirez, "The low-power architecture approach towards exascale computing," in *Proceedings of the Second Workshop on Scalable Algorithms for Large-scale Systems*, ser. ScalA '11. New York, NY, USA: ACM, 2011, pp. 1–2. [Online]. Available: http://doi.acm.org/10.1145/2133173.2133175

[4] V. Weaver, M. Johnson, K. Kasichayanula, J. Ralph, P. Luszczek, D. Terpstra, and S. Moore, "Measuring energy and power with papi," in *Parallel Processing Workshops (ICPPW), 2012 41st International Conference on*, Sept 2012, pp. 262–268.

[5] L. McVoy and C. Staelin, "Lmbench: Portable tools for performance analysis," in *Proceedings of the 1996 Annual Conference on USENIX Annual Technical Conference*, ser. ATEC '96. Berkeley, CA, USA: USENIX Association, 1996, pp. 23–23. [Online]. Available: http://dl.acm.org/citation.cfm?id=1268299.1268322

[6] J. J. Dongarra, P. Luszczek, and A. Petitet, "The linpack benchmark: past, present and future," *Concurrency and Computation: Practice and Experience*, vol. 15, no. 9, pp. 803–820, 2003. [Online]. Available: http://dx.doi.org/10.1002/cpe.728

[7] B. E. Treeby and B. T. Cox, "k-wave: Matlab toolbox for the simulation and reconstruction of photoacoustic wave-fields," *J. Biomed. Opt.*, vol. 15, no. 2, p. 021314, 2010.

[8] V. Nikl and J. Jaros, "Parallelisation of the 3d fast fourier transform using the hybrid openmp/mpi decomposition," in *Mathematical and Engineering Methods in Computer Science*, ser. LNCS 8934. Springer International Publishing, 2014, pp. 100–112.

[9] J. Jaros, V. Nikl, and E. B. Treeby, "Large-scale ultrasound simulations using the hybrid openmp/mpi decomposition," in *Proceedings of the 3rd International Conference on Exascale Applications and Software*. Association for Computing Machinery, 2015, pp. 115–119.

Využití rychlého offline testu v systému se schopností maskování jedné chyby

Jan Bělohoubek
2. ročník, prezenční studium
Školitel: Petr Fišer, Specialista: Jan Schmidt

Fakulta informačních technologií ČVUT
Thákurova 9, 160 00, Praha 6
jan.belohoubek@fit.cvut.cz

***Abstrakt*—V článku je představena nová metoda pro návrh systémů maskujících jednu chybu, která kombinuje redundanci v ploše a v čase. Schopnost maskování chyb je srovnatelná s TMR. Navržená metoda je porovnána s TMR z hlediska plochy čipu a je identifikována skupina obvodů, pro které je její použití vhodné.**

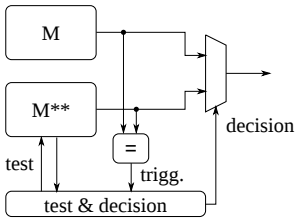
***Klíčová slova*—Maskování jedné chyby, offline test, hradlo, redundance, řiditelnost, pozorovatelnost, porucha, stuck-open/stuck-on**

I. Motivace

Spolehlivost je jedním z nejdůležitějších aspektů vývoje číslicových systémů. V centru pozornosti návrhářů jsou po léta metody pro *detekci a maskování chyb* (*error-detection and error-masking*) založené na multiplikaci funkčních bloků (*N-modular redundancy*). Výhodou použití těchto metod je přímočarost implementace, stejně jako dlouholeté zkušenosti s návrhem i provozem takových systémů [1].

Robustnost zmíněných metod lze dobře ukázat na nejjednodušším systému pro maskování jedné chyby na výstupu, který je složen ze tří totožných bloků – TMR (*Triple Modular Redundancy*). Systém TMR dokáže tolerovat jeden vadný modul ze tří za předpokladu, že oba zbývající moduly jsou bezchybné. TMR dovoluje maskovat chyby na výstupu způsobené jakýmkoli typem poruch – ať už se jedná o poruchy přechodné (*transient fault*, *soft-fault*) nebo trvalé (*permanent fault*) [1]. Výhodou TMR (i příbuzných metod) je, že k maskování chyb dochází bez významného vlivu na zpoždění v celém systému.

Nevýhodou metod založených na několikanásobném použití týž částí je značný negativní vliv na využitou plochu čipu. Příbuzná metoda používaná pouze pro detekci poruchy potřebuje přibližně o třetinu menší plochu čipu – *duplex*. Uvážíme-li, že i v případě duplexu je při výskytu jedné poruchy (jedna část duplexu je vadná) k dispozici správný výsledek, může se přístup využívaný v TMR jevit nehospodárně. Třetí modul v případě TMR slouží výhradně k výběru správného výsledku. Nový přístup k maskování chyb na výstupu systému prezentovaný v tomto článku umožňuje, za cenu malého zpoždění, vybrat správný (poruchou neovlivněný) výstup duplexu. Vlastností celého systému je, že přidané zpoždění se projeví pouze v případě detekce chyby. Schopnost maskovat chyby na výstupu je u navrženého řešení stejná jako u TMR, přičemž využitá plocha čipu může být v některých případech znatelně menší. Konceptuální schéma navrženého systému je na obrázku 1.



Obrázek 1. Konceptuální schéma duplexu umožňujícího maskování chyb. Modul M^{**} lze otestovat velmi rychlým testem. Na základě výsledku provedeného testu dojde k výběru správného výstupu. Po provedení testu je proveden přepočet (*recomputation*), aby byl vyloučen vliv přechodných poruch. Takový systém nazýváme *Time-Extended Duplex* (TED)

V oblasti testování logických obvodů je značným problémem délka testu. Není výjimečné, že délka testu se počítá v tisících testovacích vektorech [1]. Prezentovaná metoda z principu vyžaduje velmi krátký offline test se 100% pokrytím poruch (vzhledem k velmi přesnému poruchovému modelu). V případě, že by byl test příliš dlouhý, byla by doba nutná pro jeho vykonání (a tedy doba pozastavení výpočtu) neakceptovatelná. V případě neúplného pokrytí poruch by nebylo možno garantovat správnost rozhodnutí založeného na výsledku testu. Uvedené požadavky jsou dle současných poznatků v oblasti testování v přímém protikladu. Z těchto důvodů byly navrženy nové struktury (CMOS) umožňující provést extrémně krátký test se 100% pokrytím poruch vzhledem k tranzistorovému poruchovému modelu *stuck-open/stuck-on* (rozpojení/zkrat).

Zbytek článku je členěn takto: V sekci II jsou představeny základní principy krátkého offline testu a hardwarové struktury umožňující takový test provést. V sekci III jsou stručně popsány vlastnosti krátkého testu, který byl podrobně prezentován v [2] a [3]. V sekci IV je detailně popsán Time-Extended Duplex a v sekci V jsou prezentovány výsledky srovnání TED a TMR.

II. Princip offline testu

Testování logických obvodů je založeno na tom, že od místa výskytu poruchy je na výstup obvodu propagován *příznak poruchy* (*fault symptom*). Poruchové příznaky mohou být během propagace maskovány. Při použití základních hradel AND a OR pozorujeme, že příznak hodnoty 0 je dominantní vzhledem k hradlu AND (hradlo AND vždy propaguje hodnotu příznaku 0) a příznak hodnoty 1 je dominantní vzhledem k hradlu OR. Hodnoty příznaku poruchy, které nejsou dominantní, mohou být během propagace maskovány.

V případě, že výstup každého z hradel je připojen k alespoň jednomu hradlu AND a alespoň jednomu hradlu OR, dojde vždy k propagaci příznaku poruchy (*indication princpile*) za předpokladu, že na vstupu takového obvodu je konstantní vektor (tj. vektor samých jedniček resp. vektor nul). Pokud obvod neobsahuje žádné invertory, je v případě bezporuchového stavu jeho výstup roven jeho vstupu (až na délku) – vektor jedniček, resp. vektor nul.

Pro kombinační obvod s výše uvedenými vlastnostmi existuje vzhledem k modelu *trvalá jednička/trvalá nula* (*stuck-at-fault model*) test, který se skládá pouze ze dvou testovacích vektorů – samé jedničky a samé nuly.

A. Konverze kombinačních obvodů

Aby byly splněny požadavky pro krátký test, je třeba provést konverzi obecného kombinačního obvodu.

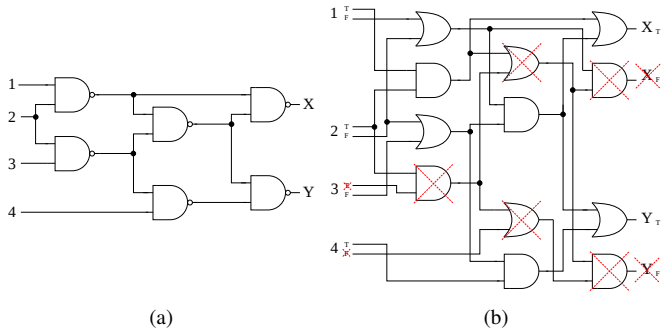
Libovolný kombinační obvod je možné snadno převést na obvod, který neobsahuje invertory. Toho lze dosáhnout použitím dvoudrátové logiky (*dual-rail logic*). Takový obvod obsahuje pouze hradla AND a OR [4].

Dále je potřeba zajistit, aby výstup každého hradla byl připojen jak k hradlu OR, tak k hradlu AND. Splnění této podmínky lze dosáhnout použitím rekonfigurovatelného hradla (hradlo se chová jako AND nebo OR) – podrobnosti viz [5], [2] a [3]. Použití rekonfigurace dovoluje propagovat příznak poruchy na výstupy, které mohou být poruchou ovlivněny (tj. umožňuje lokalizaci poruchy vzhledem k výstupu). Na druhou stranu, rekonfigurace sama komplikuje test (navíc je nutné provést test funkce rekonfigurace).

Vlivem použití dvoudrátové logiky dochází v modifikovaném obvodu k nárůstu počtu hradel – na obrázku 2a je obvod v klasické jednodrátové variantě (*single-rail*) složený z hradel NAND a na obrázku 2b jeho obraz v dvoudrátové variantě.

V případě, že se spokojíme pouze s jednodrátovými výstupy, je možné počet přidaných hradel výrazně zmenšit. To je vidět na obrázku 2b, kde byla odebrána přesně polovina výstupů a také všechna hradla spojená pouze s odebranými výstupy. Výsledný obvod může mít po odebrání nadbytečných hradel v nejlepším případě dokonce stejný počet hradel, jako obvod před konverzí, a má minimálně stejný počet vstupů.

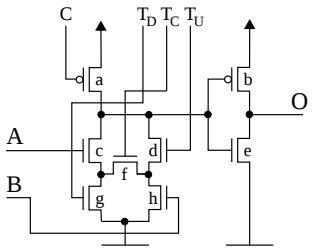
Jsou-li takto redukované obvody zřetězeny, je samozřejmě nutné použít převodníky z jednodrátové na dvoudrátovou logiku. Ty jsou realizovány pouze invertory. Výsledný obvod se tedy skládá z pole invertorů na vstupu logiky, která invertory neobsahuje. Logiku bez invertorů lze otestovat velmi krátkým testem, avšak invertory je nutno testovat odděleně.



Obrázek 2. Příklad obvodu z hradel NAND (a) a jeho implementace ve dvoudrátové logice (b)

B. Rekonfigurovatelné hradlo

Rekonfigurovatelné hradlo je struktura navržená tak, aby přesně emulovala chování hradel AND a OR v *domino logice* (*domino logic*) [6] – viz obrázek 3. Výhoda domino logiky spočívá v tom, že většina struktury je tvořena tranzistory typu N. To platí jak pro klasickou domino logiku, tak pro navržené rekonfigurovatelné hradlo. Výsledná struktura je tudíž úsporná z hlediska plochy a zároveň zachovává zpoždění klasických domino hradel [6].



Obrázek 3. Navržené rekonfigurovatelné hradlo

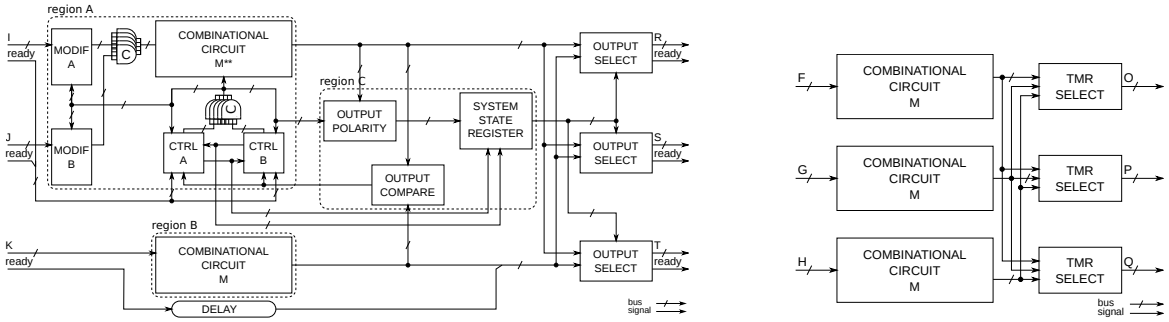
Tabulka I
Nastavení signálů pro funkční mód OR a AND

step	C	T_U	T_C	T_D	O
precharge	0	0	0	0	↓
evaluation OR	1	1	0	1	↕
evaluation AND	1	0	1	0	↕

Rekonfigurovatelné hradlo je vybaveno třemi řídícími signály: T_U, T_C a T_D. Jako ostatní domino hradla, i toto pracuje ve dvou fázích: *precharge*, během které dochází k přednastavení hodnoty 0 na výstup hradla a *evaluation*, během které jsou vyhodnoceny vstupy a nastaven výstup hradla. Nastavení signálů hradla pro funkční mód OR a AND jsou v tabulce I.

III. Test stuck-open/stuck-on poruch

Je-li kombinační obvod realizován s použitím navržených rekonfigurovatelných hradel, je možné tento obvod otestovat velmi krátkým testem. Test popsaný v [2] a [3] je rozšířením testu popsaného v sekci II.



(a) (b)

Obrázek 4. Detailní schéma Time-Extended Duplexu (a) a TMR (b)

Test kombinační logiky využívá schopnosti rekonfigurace hradel a v několika po sobě následujících fázích testuje různé druhy poruch. Jednotlivé operace jsou prováděny vždy zároveň se všemi hradly, která mají stejnou hloubku. Postupuje se od primárních vstupů směrem k výstupu. Během testu hradla pomocí definované sekvence přechodů na řídících signálech a primárních vstupech procházejí množinou stavů, přičemž výstup hradla se v konečném chybovém a bezchybném stavu liší. Test je navržen tak, že produkuje-li hradlo chybový výstup, jeho následovníci skončí též v chybovém stavu. Tímto způsobem dochází k propagaci poruch na primární výstupy kombinační logiky. Podrobný popis testu je uveden v [2] a [3].

Uvedený test má pokrytí všech poruch vzhledem k modelu *stuck-open/stuck-on* s výjimkou poruch *stuck-on* (zkrat) na tranzistorech označených `b` a `e`. Tyto poruchy lze však otestovat pomocí *měření poruchového proudu* (*fault-current measurement*).

V předchozích letech byly představeny zabudované senzory poruchového proudu – BICS (*Built-In Current Sensors*) [7], [8]. Výhodou použití BICS spolu s navrženým offline testem je, že proudový senzor musí monitorovat pouze jednu napájecí větev – tím dosáhneme snížení jeho velikosti.

Test kombinační logiky spolu s BICS má 100% pokrytí poruch vzhledem k modelu *stuck-open/stuck-on* a jeho délka závisí pouze na hloubce obvodu [2], [3]:

$$t \leq (3d + 4) \cdot t_e \ [+ \ 2 \cdot t_{BICS}], \tag{1}$$

kde d je hloubka obvodu, t_{BICS} je doba měření chybového proudu a t_e je rovno třem cyklům (hodinovým taktům). Měření poruchového proudu se může plně překrývat s prováděním testu, avšak použití BICS může, v závislosti na implementaci, znamenat další zpoždění.

IV. Navržená architektura

Díky krátkému offline testu kombinační logiky je možné zkonstruovat systém schopný maskovat jednu chybu na výstupu. Protože je offline test velmi krátký a je aktivován pouze v případě výskytu chyby, je vliv na propustnost systému malý.

Podrobné schéma navrženého systému (Time-Extended Duplex – TED) je na obrázku 4a. TED obsahuje moduly `M` a `M**`. Modul `M` realizuje kombinační funkci a je implementován stejným způsobem jako kombinační modul v TMR. Modul `M**` je funkčně ekvivalentní modulu `M`, je však realizován z navržených rekonfigurovatelných hradel. Modul `M**` je možno otestovat navrženým krátkým testem. V *regionu A* na obrázku 4a se dále nachází moduly `MODIF`, jež slouží jako generátor testu a zároveň jako konvertory jednodrátových signálů na signály dvoudrátové. Díky duplikaci modulů `MODIF` je možno detekovat poruchy v těchto modulech a případné rozdíly ve vstupních vektorech `I` a `J`. V *regionu A* se dále nachází řadič testu, který je navržen jako *self-checking* obvod – případná chyba v řadiči je detekována.

Region C obsahuje komparátor výstupů `M` a `M**` – `OUTPUT_COMPARE`, který slouží ke spuštění testu, dále modul `OUTPUT_POLARITY`, který slouží ke kontrole výstupu obvodu v testovacím módu a `SYSTEM_STATE_REGISTER`, jež uchovává informaci o poruchách detekovaných offline testem nebo o chybách detekovaných self-checking řadičem.

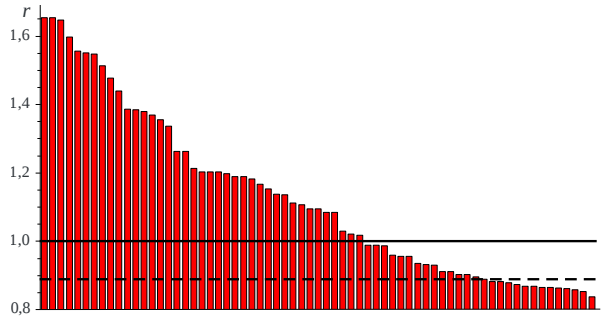
Moduly `OUTPUT_SELECT` slouží, podobně jako v případě TMR, k výběru správného výstupu. Rozhodnutí je založeno nejen na porovnání výstupů `M` a `M**`, ale navíc na informaci uložené v `SYSTEM_STATE_REGISTER`. Tento registr zároveň slouží pro redukci frekvence provádění testu v případě výskytu trvalé poruchy – pro výběr správného výstupu lze využít informace uložené v tomto registru a není potřeba opakovaně spouštět test. Vzhledem k možnému výskytu přechodných poruch je vhodné obsah `SYSTEM_STATE_REGISTER` (a řadič testu) periodicky resetovat. Tímto způsobem lze řídit propustnost TED.

V. Experimentální ověření

Pro ověření použitelnosti metody a pro zhodnocení jejích vlastností bylo provedeno množství experimentů s použitím benchmarkových obvodů z těchto sad: *LGSynth'91* [9], *LGSynth'93* [10], *ISCAS'85* [11], *ISCAS'89* [12], a *IWLS 2005* [13] – viz [5], [2] a [3].

Pro srovnání různých přístupů byl vytvořen model reflektující vlastnosti hradel (vstupní kapacity, výstupní proud, velikost tranzistorů), vycházející z popisu hradla na tranzistorové úrovni [2], [3]. Zpoždění rekonfigurovatelných hradel pro funkci AND a funkci OR odpovídá přesně zpoždění hradel domino logiky. Velikost rekonfigurovatelných hradel je při témže zpoždění vyšší než u hradel domino logiky.

Na základě vlastností TED byla ze sady benchmarků vybrána skupina obvodů, jež umožňuje smysluplnou realizaci TED. Jsou to kombinační obvody, které mají větší velikost než logika TED a zároveň mají menší počet vstupů/výstupů. Počet vstupů/výstupů má totiž vliv na velikost komparátorů (*region C*) a tím významně ovlivňuje velikost a zpoždění TED. Pro podrobnější experimentální vyhodnocení byly vybrány obvody s alespoň 3k hradly a maximálně 15k vstupů/výstupů.



Obrázek 5. Srovnání TED a TMR pro 67 vybraných kombinačních obvodů. Na svislé ose je uveden poměr plochy implementací TED a TMR pro tytéž kombinační obvody: $r = \frac{|\mathtt{TED}|}{|\mathtt{TMR}|}$. Obvody jsou řazeny sestupně dle r

S použitím modelu hradel bylo provedeno srovnání velikosti TMR a TED. TMR bylo pro všechny benchmarkové obvody realizováno dle obrázku 4b a TED dle obrázku 4a. TMR bylo implementováno v domino logice s použitím dvouvstupových hradel. TED byl realizován stejným způsobem, pouze klasická domino hradla byla v modulu `M**` nahrazena hradly rekonfigurovatelnými.

Graf na obrázku 5 ukazuje výsledky srovnání pro 67 vybraných kombinačních obvodů zabezpečených pomocí TED a TMR. Výsledky ukazují, že pro přibližně 1/3 obvodů (plná čára na obrázku 5) je realizace TED z hlediska plochy výhodnější než realizace TMR [2], [3].

Na základě vyhodnocení vztahů mezi velikostmi různých bloků v TED a TMR byl empiricky stanoven výraz [2], [3]:

$$18 \cdot |\mathtt{TMR\ SELECT}| < |\mathtt{M}| \qquad (2)$$

Výraz (2) určuje vztah mezi velikostí modulů `TMR_SELECT` a `M` v TMR. Je-li tento vztah splněn, pak je z hlediska plochy čipu výhodnější realizace TED. Za předpokladu, že známe parametry implementace TMR v domino logice, umožňuje výraz (2) určit, zda je realizace TED výhodnější z hlediska plochy či nikoli. Na obrázku 5 se obvody, pro něž platí vztah (2) nacházejí pod přerušovanou čárou a tvoří přibližně 1/6 všech obvodů – vztah (2) je pesimistický.

VI. Závěr

V článku byly stručně představeny základní myšlenky nové metody pro návrh systémů schopných maskování jedné chyby na výstupu. Navržená metoda byla srovnána s TMR a byla určena skupina obvodů, pro něž je navrhované řešení výhodnější než TMR.

Nevýhodou navržené metody proti TMR je, že zpoždění a plocha komparátorů na výstupu kombinačních částí jsou přímo úměrné počtu výstupů – z toho důvodu je použití TED výhodné zejména pro obvody s menším počtem výstupů.

Z experimentů provedených s použitím velkého množství obvodů z několika benchmarkových sad vyplývá, že TED je vhodné použít zejména pro zabezpečení kombinačních obvodů s větší hloubkou – jejich velikost je výrazně větší než velikost přidané logiky TED. Empiricky byl odvozen jednoduchý vztah, který umožňuje určit, zda je (z hlediska plochy na čipu) pro daný kombinační obvod zabezpečení pomocí TED vhodnější než zabezpečení pomocí TMR.

Další výzkum se zaměří zejména na popis vlastností obvodů nutných pro realizaci velmi krátkého offline testu a na obvodové transformace umožňující významné zkrácení délky testu s minimálním vlivem na plochu a zpozdění obvodu. Pokusím se formulovat nutné a postačující podmínky krátkého testu.

V rámci řešení projektu "GA16-05179S" se budu zabývat implementací mikroarchitektury umožňující maskování chyb s minimálním vyzařováním postranními kanály. Cílem je navrhnout řešení, které bude odolné proti poruchám a omezí vyzařování postranními kanály v bezchybném provozu i v případě výskytu poruchy. Předpokladem je, že oprava chybné hodnoty se odehraje co nejblíže zdroji poruchy.

Poděkování

Výzkum byl částečně podpořen grantem GA16-05179S GA ČR a grantem ČVUT SGS16/121/OHK3/1T/18.

Reference

[1] I. Koren and C. M. Krishna, *Fault-Tolerant Systems*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007.

[2] J. Bělohoubek, "Error correction method based on the efficient offline test," FIT CTU in Prague, Tech. Rep., 2016.

[3] J. Bělohoubek, P. Fišer, and J. Schmidt, "Error Correction Method Based On The Short-Duration Offline Test," in *Euromicro Conference on Digital System Design (DSD), 2016*, Aug 2016.

[4] J. Sparsø and S. Furber, *Principles of Asynchronous Circuit Design: A Systems Perspective*, 1st ed. Kluwer Academic Publishers, Boston, 2001.

[5] J. Bělohoubek, P. Fišer, and J. Schmidt, "Novel C-Element Based Error Detection and Correction Method Combining Time and Area Redundancy," in *Euromicro Conference on Digital System Design (DSD), 2015*, Aug 2015, pp. 280–283.

[6] N. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, 4th ed. USA: Addison-Wesley Publishing Company, 2010.

[7] S. Athan, D. Landis, and S. Al-Arian, "A novel built-in current sensor for IDDQ testing of deep submicron CMOS ICs," in *Proceedings of 14th VLSI Test Symposium, 1996.*, Apr 1996, pp. 118–123.

[8] R. Possamai Bastos, J.-M. Dutertre, and F. Sill Torres, "Comparison of bulk built-in current sensors in terms of transient-fault detection sensitivity," in *5th European Workshop on CMOS Variability (VARI), 2014*, Sept 2014, pp. 1–6.

[9] S. Yang, "Logic synthesis and optimization benchmarks user guide: Version 3.0," MCNC Technical Report, Tech. Rep., Jan. 1991.

[10] K. McElvain, "IWLS'93 Benchmark Set: Version 4.0," Tech. Rep., May 1993.

[11] F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran," in *Proceedings of IEEE International Symposium Circuits and Systems (ISCAS 85)*. IEEE Press, Piscataway, N.J., 1985, pp. 677–692.

[12] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *IEEE International Symposium on Circuits and Systems, 1989.*, May 1989, pp. 1929–1934 vol.3.

[13] C. Albrecht, "IWLS 2005 Benchmarks," Tech. Rep., Jun. 2005.

Nová a efektivní metoda pro zajištení platnosti dat ve vestavných pamětech FPGA se zaměřením na kompresi IP packetů v reálném čase

Matěj Bartík
2. ročník prezenčního studia
Školitel: Sven Ubik, školitel specialista: Pavel Kubalík

České vysoké učení technické v Praze, Fakulta informačních technologií
Thákurova 9, Praha 6 - Dejvice, 160 00, Česká republika
bartimat@fit.cvut.cz

***Abstrakt*—Tento článek se zabývá novým a efektivním způsobem zajištěná platnosti dat ve vestavných pamětech uvnitř FPGA, který je vhodný pro realizaci slovníků v bezztrátových kompresních algoritmech realizovaných v hardwaru (FPGA). Klíčem k této inovaci je chytré využití vlastností LUT (Look-Up Table), které umožňuje dosáhnout menšího počtu využitých zdrojů a vyšší frekvence celého systému oproti běžně používaným způsobům realizace. Tato metoda je navržena pro vysokou propustnost a nízkou latenci, což ji činí vhodnou pro kompresi jumbo IP packetů obsahující multimediální data v reálném čase. Použitou metodu je možné aplikovat na další datové struktury, které jsou mapovány do vestavných bloků RAM v FPGA.**

***Klíčová slova*—FPGA, LUT, slovník, bezztrátová, komprese, LZ4, LZ77, efektivní, jumbo, IP, packety**

I. Úvod a motivace

Na základě motivace shrnuté v [1] jsem se rozhodl za cíl dizertace zvolit „Tvorba nových optimalizací těžících z FPGA architektury“ i když některé tyto optimalizace mohou být inspirovány druhým tématem „Metody pro přenos procesorových optimalizací na FPGA“. V následujících kapitolách popíši především nově zjištěné poznatky vůči předchozímu (prvnímu) ročníku [1] doktorandského studia.

II. Analýza

Moderní implementace bezztrátových kompresních algoritmů, které jsou implementovány v hardwaru (FPGA), mají některé společné vlastnosti [1]. Jednou z nich je zvětšení šířky zpracovávaného slova z původně běžné hodnoty 8 bitů na 32, 64 nebo 128 bitů [2]–[4]. Původní 8-bitový přístup vychází z triviální implementace kompresních algoritmů, kdy takový přístup značně zjednodušuje výsledný hardware, protože se obejde bez pokročilé práce s pamětí (zarovnání paměti) [5].

A. Problémy HW implementací

Typickým přístupem k paralelizaci 8-bitových implementací byla instanciace více stejných kompresních bloků do FPGA, mezi které je rozdělena zátěž. V současné době se kromě použití více stejných bloků dostává do popředí i paralelizace uvnitř každého kompresního bloku právě zvětšením šířky zpracovávaného slova. S tím souvisí nutnost paralelizace celého mechanismu vyhledávání shody.

Prvním typickým příkladem řešeného problému je paralelizovatelnost přístupu ke slovníku, ve kterém je vyhledávána shoda [3]. Druhým typickým příkladem je použití větších slovníků pro zvýšení kompresního poměru, kdy je potřeba tento slovník před každou sadou vstupních dat (re)inicializovat [1] [5]. Právě tento problém bych rád podrobněji rozebral v tomto článku.

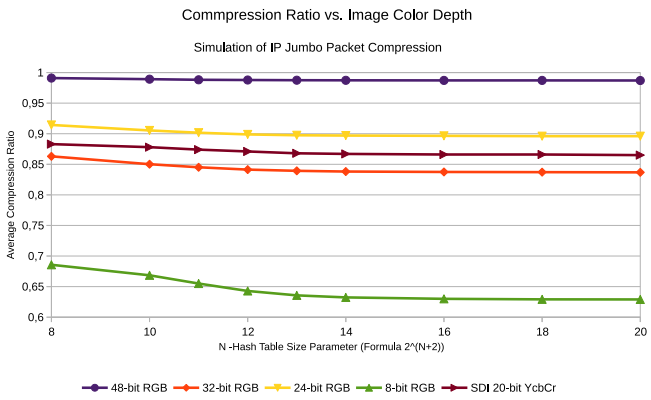
III. Analýza

A. Kompresní algoritmus LZ4

Bezztrátový kompresní algoritmus LZ4 [6] je derivátem LZ77 [7] a příkladem moderního „rychlého“ („fast“) kompresního algoritmu. Tato třída bezztrátových kompresních algoritmů se vyznačuje využitím velkého množství procesorových optimalizací (typicky je velikost slovníku rovna nebo menší než L1 cache) [8]. Tím se dosahuje velké rychlosti komprese, často ale na úkor kompresního poměru. LZ4 jsem zvolil jako výchozí algoritmus pro HW implementaci, jehož optimalizace se pokusím portovat do hardwaru. Pokud identifikuji některé neoptimální části LZ4, můžu se pokusit navrhnout další optimalizace speciálně navržené pro FPGA. Mezi klíčovou optimalizaci LZ4 patří mechanismus vyhledávání shody.

1) Mechanismus vyhledávání shody LZ4: Vyhledání shody je u algoritmu LZ4 založeno na hashovací tabulce [4], která reprezentuje slovník. Každý záznam v hashovací tabulce obsahuje odkaz do paměti vstupních dat (pointer) s potencionálním kandidátem na shodu. Adresu záznamu je vypočtena ze 32-bitových vstupních dat pomocí multiplikativního hashování s prvočíslem 2654435761. Toto prvočíslo je nejblíže hodnotě $\frac{2^{32}}{\phi} = 2654435769$, kde ϕ je hodnota Zlatého řezu. V případě FPGA může být implementace provedena pomocí DSP bloků.

Hashovací tabulka musí být před každým během algoritmu LZ4 (re)inicializována. V případě softwarové varianty LZ4 (referenční zdrojový kód je v jazyce C) se každá inicializace



Obrázek 1. Vliv bitové hloubky, kódovacího schématu a velikost hashovací tabulky na kompresní poměr. [1]

provádí alokováním paměti (malloc) a následně je lineárním průchodem každá buňka pole (záznam tabulky) inicializován na výchozí hodnotu 0 (null pointer). V případě hardwarové implementace je nutné všechny buňky hashovací tabulky inicializovat do výchozí hodnoty nebo udržovat informaci, zda je záznam v dané buňce platný či neplatný.

B. Experimentální měření

V současné době je prováděn výzkum [9], zda lze „rychlé“ bezztrátové kompresní algoritmy použít pro „lehkou“ kompresi multimediálních dat, přestože jsou k tomu určeny jiné algoritmy pro specifické typy dat, převážně ztrátové. Tyto algoritmy jsou však mnohem složitější (je potřeba více logických hradel). Dále komprese trvá delší dobu, tedy nepříznivě ovlivňují zpoždění. Prvním krokem je tedy analýza vhodnosti kompresního algoritmu LZ4 pro přenos IP packetů obsahující multimediální data. Na základě experimentálních měření jsem se rozhodl právě pro algoritmus LZ4, který dosahoval průměrné úspory 23%.

Tyto experimenty však neuvádějí, jak velký byl použit slovník a LZ4 byla implementována ve formě softwaru. Pro realističtější odhad jsem simuloval posílání multimediálních dat v „Jumbo“ packetech s datovou částí o velikosti 9000 bytů a pro velikosti slovníků odpovídající možnostem použitého FPGA. V našem specifickém případě (SDI 20-bit) jde o průměrnou úsporu 11%–13% podle velikosti slovníku. Nejlepší výsledky v poměru velikost slovníku vůči kompresnímu poměru (na obrázku 1) vykazují slovníky o velikosti 1024–4096 záznamů [1].

C. Řešený problém a současná řešení

Jako řešený problém jsem si vybral navržení nového způsobu (re)inicializace slovníku tak, aby se zkrátila latence reinicializace a/nebo se zmenšila spotřeba zdrojů FPGA. Na základě výše uvedených informací jsem si stanovil tyto předpoklady:

- velikost slovníku minimálně 1024 záznamů,
- velikost datové části IP packetu 9000 bytů,
- MVTP používá 64-bitovou datovou cestu na 156,25 MHz (požadavky 10G Ethernetu).

Z předpokladů je patrné, že packet o maximální velikosti 9000 bytů je spodní mez pro kompresi 1125 hodinových taktů. Pro první běh kompresního algoritmu může být hashovací tabulka inicializována pomocí výchozích hodnot v bitstreamu. Další běhy algoritmu je nutné prosvést (re)inicializaci pomocí logiky. Za předpokladu, že (re)inicializace slovníku se dá provádět po dobu kopírování vstupních dat do paměti a výstupních dat z paměti zpět do MVTP, máme k dispozici čas v řádu stovek hodinových taktů.

V současnosti se používají tyto přístupy [10]:

- Nahrání výchozí hodnoty do každé buňky paměti lineárním průchodem paměti (za pomocí čítače) [11]
- Použitím příznakového registru z klopných obvodů, který obsahuje příznak platný/neplatný pro každou buňku paměti a lze všechny klopné obvody resetovat paralelně.

Linární průchod paměti se zápisem výchozí hodnoty je možné považovat za triviální řešení a při HW implementaci se použije čítač, který generuje adresy pro zápis. Toto řešení optimální z hlediska logických hradel, ale dlouhý carry-chain sčítačky (pro velké šířky adresy do paměti) výrazně ovlivňuje dosažitelnou frekvenci. Největší nevýhodou je dlouhá doba potřebná pro (re)inicializaci (mocnina 2 na počet adresních bitů), v našem případě minimálně 1024 taktů nebo více.

Příznakový registr je optimální z pohledu doby potřebné pro (re)inicializaci, která trvá pouze jeden hodinový takt. Jeho nevýhodou jsou exponenciální nároky na logická hradla a při větší šířce adresy nad 8 bitů je takto realizovaný systém příznaků větší, než samotný design LZ4 na FPGA [1].

IV. Návrh možného alternativního řešení

Obě metody (lineární průchod a registr příznaků) mají problémy s optimalitou z hlediska času nebo logických hradel. Pro náš případ by bylo mnohem vhodnější použít řešení, které by sice nemuselo být optimální, ale mohlo být mnohem vhodnější než obě dnes používané metody, případně i rychlejší. Typické moderní FPGA má obvykle tyto možnosti uchování informace (logické hodnoty):

- Klopné obvody,
- Vestavná paměť (BlockRAM),
- Distribuovaná paměť.

Z těchto možností jsem se rozhodl detailněji prozkoumat možnosti distribuované paměti na FPGA. Distribuovaná paměť je jedním ze 3 režimů LUTu, kdy LUT místo realizace logické kombinační funkce pracuje jako malá RAM, v případě Virtex-6 o velikosti $2^6 = 64$ bitů (Virtex-6 používá šestivstupé LUTy). Vzhledem k paralelní povaze FPGA je možné současně pracovat s více bloky distribuované paměti.

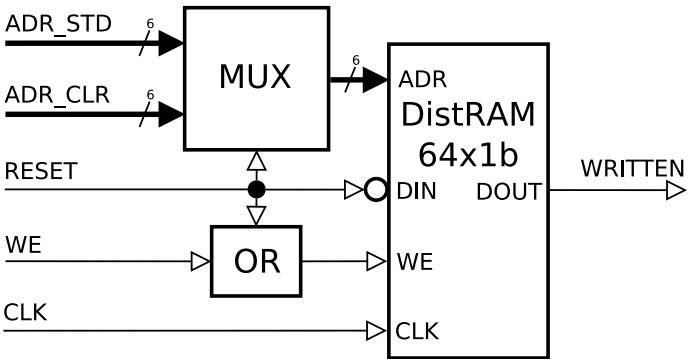
Možným řešením je použít distribuovanou paměť pro realizaci systému příznaků na místo jednotlivých klopných obvodů. Jednotlivý blok distribuované paměti je nutné inicializovat stejným způsobem jako samotnou hashovací tabulku (lineárním průchodem). Díky možnosti paralelního přístupu k jednotlivým blokům distribuované paměti je možné inicializovat všechny bloky naráz a tím je zároveň omezena doba (re)inicializace celého systému příznaků na 64 hodinových taktů (počet bitů bloku distribuované paměti).

V. Příklad možné implementace

Systém příznaků realizovaný pomocí distribuované paměti lze rozdělit na několik hlavních částí (Obr. 3).

A. Elementární blok

Elementární blok (EB) obsahující jednu buňku distribuované paměti, adresní multiplexor (Obr. 2), který přepíná mezi adresami standardního režimu a režimu (re)inicializace. Signál resetu zároveň slouží jako datový vstup. V našem specifickém případě se při přístupu ve standarním režimu provádí zápis a čtení příznaku současně (dochází automaticky k zápisu logické jedničky), zatímco v režimu (re)inicializace je automaticky zapisována logická nula.



Obrázek 2. Vnitřní architektura elementárního bloku.

B. Adresní dekodér

Adresní dekodér se sestává z bloků SPLIT a EB SEL. SPLIT rozděluje adresu na vyšší (která jde do EB SEL) a nižší část o velikosti 6 bitů, kterou se přímo adresují jednotlivé bity uvnitř elementárního bloku ve standardním režimu. EB SEL na základě vyšší části adresy a signálu write-enable (WE) generuje signály chip-enable (CE), kterým vybírá EB s požadovanou hodnotou. Zbylé signály CE vymaskují výstupní hodnoty z nevybraných EB.

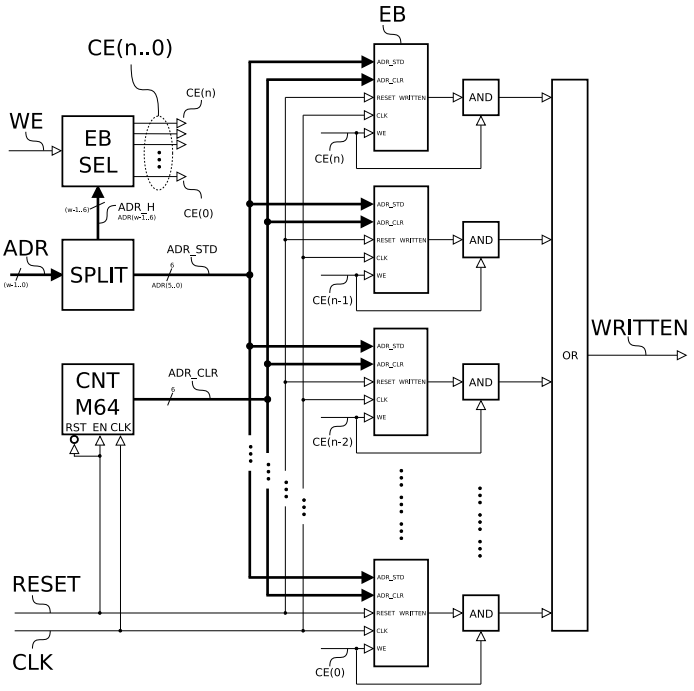
C. Čítač modulo 64

Čítač M64 slouží jako generátor adres po dobu (re)inicializace. 6-bitová sčítačka nebude s největší pravděpodobností omezujícím faktorem pro maximální dosažitelnou frekvenci.

VI. Experimentální měření a výsledky

Pro snažší měření jsem sjednotil rozhraní (Program 1) pro všechny typy přístupů (lineární průchod, příznakový registr pomocí klopných obvodů, pomocí distribuované paměti), kde velikost hashovací tabulky je nastavena parametrem „W“. Velikost datového vstupu a výstupu je 36 bitů, abych simuloval případný 32-bitový pointer, tak jak je to v SW implementaci LZ4 (36 bitů je jedna z nativních šířek BlockRAM bloků v FPGA).

Pro měření jsem použil FPGA Virtex-7 690T (XC7V690T-2FFG1158) a vývojové prostředí Xilinx ISE 14.7. Měření



Obrázek 3. Architektura celého systému příznaků.

Program 1 VHDL rozhraní pro všechny typy implementací

```
entity top is port (
  clk, reset, we : in std_logic;
  adri : in std_logic_vector(W-1 downto 0);
  din : in std_logic_vector(35 downto 0);
  dout : out std_logic_vector(35 downto 0);
  writen : out std_logic);
end top;
```

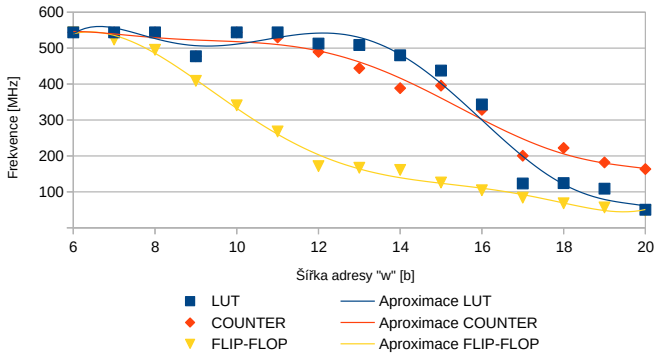
probíhala na počítači s 6-ti jádrovým procesorem Intel Xeon E5-1650 v3 (15M Cache, 3.50 GHz) a 32 GB DDR4 RAM. Nastavená syntézní a implementační strategie preferovala maximální dosažitelnou frekvenci. Následně jsem měnil požadované časování celého designu, než jsem nalezl jeho maximum. Naměřené výsledky jsou v Tabulce I a vizualizovány jako Obr. 4. Z výsledků je patrné, že nová metoda předčí oba dnes používané přístupy z hlediska dosažitelné frekvence, tak má i výborné výsledky co se týče nároků na logická hradla.

Pro hodnoty 17–20 parametru „W“ nastává výrazné zhoršení, které však přičítáme neschopnosti ISE efektivně pracovat s velkými návrhy (jak je patrné z výrazného úbytku klopných obvodů – patrně přestává fungovat duplikace čítače M64 a tím se poruší prostorová lokalita). Dále se nepodařilo provést syntézu (hodnoty 17–20 parametru „W“) pro příznakový registr využívající klopné obvody z důvody nestability Xilinx ISE 14.7 s největší pravděpodobností ze stejného důvodu, přestože by mělo být využito méně než 100% zdrojů použitého FPGA (XC7V690T-2FFG1158).

Tabulka I
VLASTNOSTI (LOGICKÁ HRADLA, FREKVENCE) JEDNOTLIVÝCH IMPLEMENTACÍ V ZÁVISLOSTI NA PARAMETRU „W“ PRO ČIP XC7V690T-2FFG1158.

COUNTER (Lineární průchod)															
Šířka adresy „W“	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Slice	96	85	93	100	111	110	112	118	122	249	287	393	194	631	347
LUTy	179	185	191	196	197	203	206	211	215	386	457	533	299	805	568
Klopné obvody	50	52	54	56	59	61	63	65	67	75	76	95	79	90	91
Frekvence [MHz]	543,8	543,8	543,8	477,3	543,8	529,7	489,5	444,0	388,7	395,6	328,3	200,5	222,3	182,0	163,5
FLIP-FLOP (Příznakový registr pomocí klopných obvodů)															
Šířka adresy „W“	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Slice	89	137	155	268	453	1157	2205	3776	10108	15774	*30335*	*55482*	*118295*	*236468*	*463594*
LUTy	221	309	478	837	1542	3417	6562	13850	31750	57582	*153323*	*242202*	*525419*	*1119292*	*2349075*
Klopné obvody	111	176	303	594	1148	2223	4157	8270	16607	32928	*75852*	*128087*	*258144*	*512953*	*1023707*
Frekvence [MHz]	543,8	523,8	495,3	408,8	340,8	268,2	171,5	167,1	161,1	126,0	*104,4*	*83,8*	*68,1*	*56,7*	*47,5*
LUT (Příznakový registr pomocí distribuované paměti)															
Šířka adresy „W“	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Slice	73	79	77	75	98	123	206	337	488	984	2364	3686	7033	13104	24987
LUTy	146	151	159	171	209	235	377	471	912	1889	3657	6737	13341	25227	55607
Klopné obvody	54	57	63	73	91	129	195	239	619	1118	2176	71	72	190	74
Frekvence [MHz]	543,8	543,8	543,8	477,3	543,8	543,8	512,6	508,6	480,1	437,4	343,2	123,4	124,3	108,9	50,2

*šedé výsledky jsou extrapolovány, protože syntézní nástroj nezvádl tak velký návrh.



Obrázek 4. Závislost frekvence na parametru „W“ pro různé přístupy.

VII. POKRAČOVÁNÍ PRÁCE A CÍL DIZERTAČNÍ PRÁCE

Na základě rešerší a současných výsledků:

- Detailní analýza vlastností kompresního algoritmu LZ4,
- Vyhodnocení (ne)vhodnosti LZ4 pro přenos multimediálních dat v softwaru (hardwaru),
- Návrh nové rychlé a na zdroje nenáročné metody pro realizaci systémů příznaků,

jsem se rozhodl pokračovat v současné práci a za cíl dizertační práce jsem si stanovil tvorbu nových optimalizací pro hardwarové implementace bezztrátových kompresních algoritmů založených na algoritmu LZ77 a které jsou určené pro vysokou propustnost nebo nízkou latenci.

VIII. ZÁVĚR

Navrhl a implementoval jsem novou metodu, kterou lze (re)inicializovat hashovací tabulku nebo jinou RAM orientovanou datovou strukturu na FPGA, která vykazuje lepší výsledky než dosud používané metody lineárního průchodu paměti a příznakového registru z klopných obvodů. Výsledná metoda založená na použití distribuované paměti dosahuje řádově konstatního času nutného pro (re)inicializaci při velmi nízkých nárocích na logická hradla.

PODĚKOVÁNÍ

Tento výzkum byl částečně podpořen z grantu SGS16/121/OHK3/1T/18.

REFERENCE

[1] M. Bartík, S. Ubik, and P. Kubalik, “Rychlé bezztrátové kompresní algoritmy,” in *Sborník příspěvků PAD 2015*, Zlín, CZ, 2015, pp. 31–36.

[2] J. L. Nunez and S. Jones, “Gbit/s lossless data compression hardware,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 3, pp. 499–510, June 2003.

[3] B. Sukhwani, B. Abali, B. Brezzo, and S. Asaad, “High-throughput, lossless data compresion on fpgas,” in *Field-Programmable Custom Computing Machines (FCCM), 2011 IEEE 19th Annual International Symposium on*, May 2011, pp. 113–116.

[4] J. Fowers, J. Y. Kim, D. Burger, and S. Hauck, “A scalable high-bandwidth architecture for lossless compression on fpgas,” in *Field-Programmable Custom Computing Machines (FCCM), 2015 IEEE 23rd Annual International Symposium on*, May 2015, pp. 52–59.

[5] M. Bartík, S. Ubik, and P. Kubalik, “Lz4 compression algorithm on fpga,” in *2015 IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*, Dec 2015, pp. 179–182.

[6] Y. Collet, “Realtime data compression: Development blog on compression algorithms.” tinyurl.com/qc9yve4, [Online; accessed 15-May-2016].

[7] J. Ziv and A. Lempel, “A universal algorithm for sequential data compression,” *IEEE Transactions on Information Theory*, vol. 23, no. 3, pp. 337–343, May 1977.

[8] J. Kane and Q. Yang, “Compression speed enhancements to lzo for multi-core systems,” in *Computer Architecture and High Performance Computing (SBAC-PAD), 2012 IEEE 24th International Symposium on*, Oct 2012, pp. 108–115.

[9] R. D. Gomes, Y. G. G. d. Costa, L. L. Aquino Júnior, M. G. d. Silva Neto, A. N. Duarte, and G. L. d. Souza Filho, “A solution for transmitting and displaying uhd 3d raw videos using lossless compression,” in *Proceedings of the 19th Brazilian Symposium on Multimedia and the Web*, ser. WebMedia ’13. New York, NY, USA: ACM, 2013, pp. 173–176.

[10] M. Stohanzl and Z. Fedra, “The fpga implementation of dictionary; hw consumption versus latency,” in *Telecommunications and Signal Processing (TSP), 2013 36th International Conference on*, July 2013, pp. 82–85.

[11] S. Rigler, W. Bishop, and A. Kennings, “Fpga-based lossless data compression using huffman and lz77 algorithms,” in *2007 Canadian Conference on Electrical and Computer Engineering*, April 2007, pp. 1235–1238.

Polymorfní elektronika a návrhové metody

Adam Crha
3.ročník, prezenční studium
Školitel: Richard Růžička

Fakulta informačních technologií, VUT v Brně
Božetěchova 2, Brno, 612 66, Česká republika
icrha@fit.vutbr.cz

***Abstrakt*—Práce popisuje výzkum týkající se nekonvenční elektroniky a návrhových metod. V úvodu jsou diskutovány principy, výhody a nevýhody nekonvenční elektroniky. Dále je pak představena polymorfní elektronika a její vlastnosti. Následuje popis metodiky, která navrhuje polymorfní obvody hledáním společných částí v termech. Je diskutován princip metody a její výsledky. Poslední kapitola lehce popisuje novou metodiku, která využívá kerneling a AIG.**

***Klíčová slova*—Polymorfismus, logický obvod, logická hradla, syntéza, metodika, AIG, kerneling.**

I. Úvod a motivace

Rozvoj číslicové techniky v šedesátých letech minulého století otevíral vědě nový, neprozkoumaný prostor. Nová technologie nabízela nové otázky, na které nebyly známé odpovědi. Vědci museli novou technologii poznávat a pozorovat chování nových materiálů, ze kterých byla vyrobena logická hradla. Jejich úkolem také bylo vymyslet vhodné nástroje, pomocí kterých se z hradel navrhovaly složitější číslicové obvody. Nyní, po více jak 60 letech existence číslicových obvodů, lze předpokládat, že jsou k dispozici ty nejlepší návrhové nástroje a chování číslicových je prozkoumáno do posledního detailu.

Ano, toto tvrzení je zcela pravdivé, hovoříme-li o běžných číslicových obvodech. Avšak v roce 2001 představil A. Stoica moderní pojem „polymorfní elektronika“, čímž otevřel další, nepříliš prozkoumanou, vědeckou oblast [1]. Jde o vícefunkční číslicové obvody, u kterých změna funkce není vyvolána přepínačem nebo rekonfigurací, jak je známo u konvenční elektroniky. Namísto toho je změna funkce vyvolána uvnitř číslicového obvodu v závislosti na externím prostředí (teplota, světlo, ...) [2]. Objev polymorfní elektroniky s sebou přinesl nové materiály, technologie a otázky týkající se efektivního návrhu polymorfních obvodů. Materiály, které dříve byly považovány za nestabilní a tudíž nepoužitelné, nacházejí uplatnění právě v polymorfní elektronice. Je možné sledovat značný pokrok ve vývoji grafenu, silikonových nanotrubiček a organických materiálů [3] [4]. Jedná se tak o velmi mladou vědeckou disciplínu nabízející mnoho disertabilních témat.

Bohužel, konvenční návrhové metody a algoritmy nejsou dobře použitelné pro návrh polymorfních obvodů. Metody syntézy pro návrh polymorfních obvodů jsou mnohem složitější oproti metodám syntézy konvenční elektroniky. Touto problematikou se již zabývá několik vědců, avšak dosud vynalezené syntézní metody nejsou natolik efektivní, jako metody pro návrh konvenční elektroniky. Tato situace vyžaduje výzkum a vývoj nových, lepších a efektivnějších návrhových metod pro polymorfní obvody. Největší přínos polymorfní elektroniky je sledován ve sdílení prostředků realizovaných funkcí v co největší možné míře. Je snahou objevovat metody, které budou generovat polymorfní obvody splňující tento předpoklad. Syntézní algoritmy pracují s obvody, nejčastěji reprezentované pravdivostní tabulkou, logickým výrazem, či binárním rozhodovacím diagramem. Výstupem by měla být co nejjednodušší reprezentace obvodu [5].

Cílem této práce je obecně představit polymorfní elektroniku a její otevřené problémy (kap. II), je zmíněna návrhová technika polymorfních obvodů (kap. IV) a jsou diskutovány její výsledky. Nakonec je představena idea nové návrhové techniky pro polymorfní obvody (kap. V).

II. Polymorfní elektronika

Tato kapitola popisuje kritické shrnutí poznání v oblasti polymorfní elektroniky a specifikuje otevřené problémy k řešení.

Polymorfní elektronika je nová disciplína v oblasti elektronických systémů. Jak již bylo zmíněno v úvodu, objevil ji A. Stoica se svou výzkumnou skupinou v NASA (Jet Propulsion Laboratory), která se primárně zabývá výzkumem elektronických technologií pro dlouhé nepilotované kosmické mise, kde je nezbytná odolnost proti poruchám, jejich kompenzace, automatické přizpůsobení se tvrdým a extrémním podmínkám [1].

V oblasti počítačových systémů se polymorfní elektronikou rozumí elektronické číslicové obvody, které dokážou vykonávat více než jednu funkci, zatímco zapojení elektronického obvodu je stále stejné. Volba funkce, kterou obvod vykonává je závislá na stavu okolního prostředí (teplota, tlak, vlhkost, polarita napětí, ...). Všechny požadované funkce obvodu jsou navrženy úmyslně. Jedná se tak o požadované funkce obvodu, nikoliv například o poruchový stav vyvolaný překročením provozních parametrů obvodu. Stav okolního prostředí je možné přesně popsat, typicky nějakou fyzikální veličinou. Pak je možné pro konkrétní hodnotu této veličiny určit, jakou funkci bude polymorfní obvod realizovat [6] [1].

Typické vlastnosti polymorfních obvodů [7]:

1) Obecně se od polymorfních obvodů očekávají menší nároky na využitou plochu na čipu. Předpokládá se, že dvě různé funkce mohou sdílet některé společné části a tím uspořit značné množství prostředků.
2) Přepínání mezi funkcemi je závislé na stavu okolního prostředí, přepnutí je okamžité a není nutné čekat například na dokončení rekonfigurace.
3) Reakce na stav okolního prostředí je přirozenou vlastností obvodu. S tímto chováním je ve fázi návrhu nutno počítat.
4) Obvod reaguje na stav okolního prostředí globálně. Všechny prvky obvodu jsou o stavu prostředí informovány stejným způsobem. Jedná se tak o distribuovaný systém, nikoliv centralizovaný.

Vzhledem k výše uvedeným vlastnostem polymorfních obvodů lze předpokládat, že tato vědecká disciplína, polymorfní elektronika, má jistý potenciál.

V následujících řádcích jsou definovány základní pojmy použité v této práci:

Definice 1. ***Digitální obvod*** *[7]*
Digitální obvod je možné reprezentovat acyklickým grafem $G = (V, E, \varphi)$, kde V je množina vrcholů (bran komponent obvodu), $E = \{(a, b)|a, b \in V\}$ je množina hran (spojů obvodu) a φ je zobrazení, které každému vrcholu z V přiřazuje komponentu z množiny K, $\varphi : V \to K$.

Definice 2. ***Polymorfní obvod*** *[7] [6]*
Polymorfní obvod je reprezentovaný acyklickým grafem $G = (V, E, \phi)$, kde V je množina uzlů (V/V hradel), $E = \{(a, b)|a, b \in V\}$ je množina hran (spojů) a $\phi = \{\varphi_1, ..., \varphi_n\}$ je množina zobrazení a platí $|\phi| > 1$. Každé zobrazení $\varphi_i \in \phi$, přiřazuje každému uzlu z V hradlo z množiny K, $\varphi_i : V \to K$ pro $\forall i = 0..n$.

Z uvedené definice plyne, že zapojení polymorfního obvodu (jednotlivých hradel / komponent), tedy graf G je stejný. Vlivem prostředí však může být ovlivněna výsledná funkce obvodu. Pak se musí měnit význam jednotlivých komponent uvnitř obvodu.

Tato definice zavádí zajímavý problém, a to: Jak nalézt ideální acyklický graf G?

III. Otevřené problémy

Na základě předchozích úvah vyplývají některé ne zcela vyřešené problémy:

1) Vyvinout vhodná hradla (na materiálové úrovni), které by byla schopna exaktně a předvídatelně reagovat na stav prostředí.
   Těmito problémy se zabývají zejména vědci z chemických a elektrotechnických univerzit. K výzkumu materiálů a nových technologií mají vhodné laboratorní podmínky, díky čemuž je jejich výzkum podpořen. Poměrně zajímavými materiály se v současnosti jeví grafen, organické polovodiče, silikonové nanodrátky a mnoho dalších zajímavých polovodivých struktur ze kterých mohou být vytvořena polymorfní hradla [3] [4].
2) Vyvinout efektivní metodu pro návrh polymorfní obvodů. Tato metoda musí umět z polymorfních hradel sestrojit graf G, který reprezentuje zapojení obvodu. Obvod pak musí vykonávat požadovanou funkci v požadovaném okolním stavu prostředí.
   Této problematice se intenzivně věnoval výzkumný tým prof. Sekaniny na Fakultě informačních technologií v Brně a učinili mnoho zajímavých objevů, které budou podrobně popsány např. v literatuře [8]

IV. Metodika hledáním společných částí v termech

Hlavní idea této návrhové metodiky spočívá v hledání společných částí obvodů tak, aby se společné části hledaly jako společné dělitele dvou různých termů popisující část obvodu [9] [10] [11].

Definice 3. ***Značení polymorfního multiplexoru:***
Nechť „(A|B)“ je výraz popisující část logického obvodu. „|“ je značka operátoru, značící dvouvstupý polymorfní multiplexor. „A“ je vstupní signál polymorfního multiplexoru, propagován na výstup v režimu 1, „B“ je vstupní signál polymorfního multiplexoru, propagován na výstup v režimu 2.

Princip metody v bodech:

1) Vstupem jsou již minimalizované výrazy v DNF popisující obě funkce.
2) Vytvoříme tabulku průniku o velikosti $m * n$, kde m je počet skupin termů F_1, n je počet skupin termů F_2. První sloupec bude obsahovat skupiny termů F_1, první řádek skupiny termů F_2.
3) Jednotlivá políčka tabulky vyplníme v tomto tvaru:
   průnik skupiny termů (zbylé termy F_1 | zbylé termy F_2).
4) Provádíme první průchod tabulkou. V tabulce hledáme políčka, která mají maximální průnik, tj. *minterm(1 | 1).* První nalezený minterm zapíšeme do výsledného výrazu. Nalezené mintermy mají obě funkce společný a proto není třeba, aby byl řešen polymorfně. Po zapsání mintermu do výsledného výrazu celý řádek a celý sloupec z tabulky vyškrtneme.
5) Provádíme další průchod tabulkou. V tabulce hledáme největší průnik. Políčko s největším průnikem opíšeme do výsledného výrazu a opět celý řádek a celý sloupec z tabulky vyškrtneme. *Poznámka: Pokud se v tabulce vyskytuje více políček s největším průnikem, zvolíme si*

heuristiku, která bude políčka vybírat - např. první, které je nalezeno.

6) Pokračujeme bodem 5, dokud jsou v tabulce nepokrytá políčka s alespoň nějakým průnikem. Pokud jsou všechny průniky pokryty, pokračuj dále.
7) Nyní se v tabulce nachází skupiny termů, které nemají společný dělitel - tedy nejsou nijak společné. Je nutné použít polymorfní multiplexor (ve výrazu značíme „|“), kterým se izoluje nesdílná část funkce F_1 a nesdílná část funkce F_2.

A. Experimenty s metodikou

Metodika pro návrh polymorfních obvodů, využívající tři typy polymorfních hradel, byla testována na 380 náhodně vygenerovaných obvodech seřazených do kategorií dle jejich hlavních parametrů: počet vstupních signálů, počet termů v disjunktivní normální formě a velikosti množiny ON-set vyjádřenou v procentech. Kategorie a specifikace testů jsou znázorněny v literatuře [11] v tabulce 1.

Průměrné výsledky metodiky je možné nalézt v literatuře [11] v tabulce 2, kde zeleně je zvýrazněn nejlepší výsledek a červeně nejhorší. Výsledek je znázorněn jako procentuální zlepšení ve srovnání s konvenční metodikou Espresso pro dvouvýstupové funkce. Po vynesení výsledků do grafu se ukázalo, že metodika je při zvětšujícím se počtu termů, či velikosti množiny ON-set málo škálovatelná [9] [10] [11].

V. Metodika návrhu využívající kerneling a AIG

Předchozí prezentovaná metoda disponuje dvěma nedostatky, kterými jsou špatná škálovatelnost a práce s dvouúrovňovou reprezentací obvodu. Proto je vhodné nadále pokračovat ve výzkumu návrhových technik a použít se z předchozích problémů, na které jsme narazili dříve.

Obecně můžeme návrh polymorfních obvodů můžeme rozdělit na dvě části a to:

1) Nalezení společných částí, které budou řešeny konvenčně.
2) Spojení rozdílných částí v jednu pomocí polymorfních prvků.

V nově představené metodice budeme využívat tento model k návrhu polymorfních obvodů. K nalezení společných částí bude využita technika *kerneling*, zatímco k návrhu rozdílných částí využijeme vlastnosti And-Inverter Grafů (AIG).

A. Kerneling

Nyní je možné představit koncept takzvaného „kernelu“ algebraického výrazu, který je velice dobře dokumentován v této literatuře. [12] [13] [14]:

K nalezení společných částí využijeme techniku kerneling v neupravené formě. Společná část obvodu pak bude reprezentována jednotlivými, nalezenými kernely.

Definice 4. *Nechť krychle (angl. cube) je součin podmnožiny literálů funkce f.*

Nechť $D(f)$ je množina primárních dělitelů získaná podílem výrazu f všemi možnými krychlemi.

Nechť $K(f) \subseteq D(f)$ je množina kernelů výrazu f, kde všechny prvky jsou „cube-free“ (výraz bez krychle, tj. prvky již není možné dělit krychlí).

Kernel algebraického výrazu f je „cube-free“ výsledek, který vzniká po dělení výrazu f krychlí a výraz není možno dále dělit beze zbytku.

K tomu, abychom byli schopni nalézt kernely funkce, potřebujeme takzvaný co-kernel. Co-kernel je krychle, tedy součin podmnožiny literálů funkce f. Co-kernely je možné nalézt sestrojením cube-literální matice. Po získání co-kernelů již snadno získáme kernely vydělením výrazu co-kernelem. Více informací o principech nalezení kernelu je možné nalézt v literatuře [12].

Výraz, reprezentující logickou funkci, však může mít více kernelů. Největší problém vyplývá na povrch. Nalezení nejlepšího kernelu je klasifikováno jako NP těžký problém a proto se využívá heuristických metod. K nalezení nejlepších kernelů více funkcí se používá tabulka průniků kernelů (kernel-intersection table), která je taktéž dokumentována v [12].

Technika zvaná „kerneling“ nachází uplatnění v logické syntéze za účelem nalezení nejlepších dělitelů a proto se využívá k efektivní dekompozici/faktorizaci.

Příklad 1. *Mějme funkci $F = af + bf + ag + cg + ade + bde + cde$ a $G = af + bf + ace + bce + ade + cde$. S pomocí techniky kerneling dokážeme nalézt největší společné části:*

Společné části:

$X = a + b$
$Y = a + c$
$Z = de$

Rozdílné části:

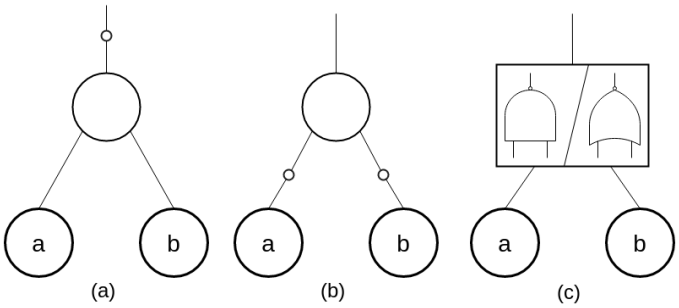
$F = ZX + fX + gY + cZ$
$G = fX + ceX + ZY$

Jak je možné spatřit na příkladu, kerneling je opravdu silným nástrojem k faktorizaci algebraických výrazů.

Algoritmus a příklady je možné nalézt v této publikaci [12].

B. AIG

V dnešní době jsou velmi populární takzvané And-Inverter grafy. Jedná se o orientované acyklické grafy využívající pouze dva typy uzlů, AND a invertor. kde invertor definován jako vlastnost hrany propojující dva uzly. Primární vstupy jsou terminály, primární výstup je reprezentován kořenem AIG. Jsou oblíbené zejména kvůli jejich jednoduchosti, což zajišťuje právě jeden typ uzlů a dva typy hran. Jsou univerzální, škálovatelné a flexibilní. AIG se také vyznačují tím, že jsou uniformní - mají pouze jeden typ uzlů. Mezi další výhody patří jednoduché operace nad grafem (balancování). AIG, stejně jako binární rozhodovací diagramy nabízejí hashování, tedy stejné podgrafy mohou být reprezentovány jedním podgrafem.



Obrázek 1. Princip mapování dvou AIG na polymorfní hradla. (a) NAND (b) NOR (c) namapované polymorfní hradlo.

K návrhu polymorfních obvodů nespolečných částí budeme využívat uniformitu AIG a sestrojíme tabulku vzorů polymorfních hradel, které máme k dispozici.

AIG vzor f_1	**AIG vzor f_2**	**Polymorfní hradlo**
AND	OR	AND/OR
XOR	NAND	XOR/NAND
a	b	poly mux
konstanta	identita	konstanta/identita
...	...	...

Tabulka I
TABULKA AIG VZORŮ MAPUJÍCÍ POLYMORFNÍ HRADLO, POUZE PŘÍKLAD.

Mějme funkce F a G z příkladu 1. Pro obě tyto funkce sestrojíme AIG. Poté, budeme procházet oba grafy od terminálních uzlů současně - tedy od stejných proměnných. Při procházení oběma sestrojenými AIG od jejich terminálů budeme vyhledávat shodu podgrafů v tabulce AIG vzorů, které odpovídají námi definovaným polymorfním hradlům.

Takto namapujeme dva různé AIG na jeden graf sestavený z polymorfních hradel. Vlastnosti AIG budou stále zachovány.

Tímto byl tedy představen koncept metodiky, jenž by měla být v dohledné době implementována do syntézního nástroje ABC.

VI. ZÁVĚR

Tato práce chronologicky popisuje průběh výzkumu týkající se polymorfní elektroniky. Nejdříve byly studovány principy polymorfní elektroniky na technologické úrovni, tedy elementárních prvků, ze kterých jsou tvořena hradla. Dále byly zmíněny techniky pro návrh polymorfních obvodů. Metodika, popsaná v kap. IV byla otestována na 380 náhodně vygenerovaných obvodech s různými parametry. Další metoda (kap. V), která je prozatím ve fázi návrhu a brzy se dočká implementace. Proto tato práce neobsahuje výsledky a srovnání. Informace získané o stávajících metodách napověděly, že syntéza polymorfních obvodů není stále ideální a je vyžadován další výzkum v oblasti návrhových technik polymorfních obvodů.

PODĚKOVÁNÍ

Tato práce vznikla za podpory národního grantu COST Nekonvenční návrhové techniky pro číslicové obvody s vlastní rekonfigurací: od materiálů k implementaci (LD14055).

LITERATURA

[1] A. Stoica, R. Zebulum, and D. Keymeulen, "Polymorphic electronics. Proc. of Evolvable Systems: From Biology to Hardware Conference," vol. 2210 of LNCS, pp. 291–302, 2001.
[2] A. Stoica and R. S. Zebulum, "Multifunctional logic gate controlled by temperature. ," p. 18, 2005.
[3] H. Raza, "Graphene Nanoelectronics: Metrology, Synthesis, Properties and Applications. ," 2012.
[4] J. Morris and K. Iniewski, "Nanoelectronic Device Applications Handbook ," 2013.
[5] P. Fišer, "Randomized Iterative Logic Synthesis Algorithms ," p. 84, 2012.
[6] A. Crha, "Polymorfní elektronika a metody syntézy." *Sborník příspěvků PAD2014*, vol. 2014, no. 1, pp. 57–62, 2014.
[7] R. Růžička, "Polymorfní elektronika." p. 118, 2011.
[8] Z. Gajda, "Evolutionary Approach to Synthesis and Optimization of Ordinary and Polymorphic Circuits ," 2011.
[9] A. Crha, R. Růžička, and V. Šimek, "Novel Approach to Synthesis of Logic Circuits Based on Multifunctional Components. ," *Journal of Electrical Engineering*, vol. 1, pp. 29–35, 2016.
[10] A. Crha, R. Růžička, and Šimek, "On the Synthesis of Multifunctional Logic Circuits." pp. 52–53, 2015.
[11] A. Crha, R. Růžička, and V. Šimek, " Toward Efficient Synthesis Method of Multifunctional Logic Circuits." pp. 21–24, 2015.
[12] G. D. Hachtel and F. Somenzi, " Logic Synthesis and Verification Algorithms ," p. 564, 1996.
[13] R. Brayton and C. McMullen, " The Decomposition and Factorization of Boolean Expressions," 1982.
[14] S. Hassoun and T. Sasao, "Logic Synthesis and Verification ," p. 454, 2002.

Generování testovacích stimulů

Ondřej Čekan
3. ročník, prezenční studium
Školitel: Zdeněk Kotásek

Fakulta informačních technologií, Vysoké učení technické v Brně
Božetěchova 2, 612 66 Brno, ČR
icekan@fit.vutbr.cz

***Abstrakt* — Tento článek popisuje jednotlivé kroky k dosažení stanovených cílů disertační práce a přehled výsledků dosažených za dobu studia. Hlavní část práce je věnována generátoru testovacích stimulů, který je založen na dvou vstupních strukturách popisujících formát a omezení generovaných stimulů. V rámci práce je ukázáno generování assemblerovských programů pro procesory, generování softwarově implementované odolnosti proti poruchám a generování bludiště. Tyto případy představují různé oblasti v použití tohoto generátoru a v práci jsou řádně zadokumentovány včetně ukázek. Závěrem práce je představena budoucí práce.**

Klíčová slova* — *verifikace; generování; stimul; program; odolnost proti poruchám; bludiště.

I. Úvod

Elektronické obvody se dostávají stále více do popředí našeho každodenního života a dalo by se říci, že od 21. století si bez nich svět již nedokážeme představit. Mikrokontroléry, které mají na starost řízení určitých zařízení, se nacházejí dnes i tam, kde bychom to ani nečekali. Jedná se o hračky, kuchyňské spotřebiče, chytré přívěsky nebo dokonce šperky. Takovéto použití představuje spíše high-end současné doby a obvody, které takovéto zařízení řídí, nejsou abnormálně spolehlivé. Na druhou stranu ale existují aplikace, u nichž jakýkoli nedostatek v návrhu nebo ve funkci systému může znamenat vážné riziko a v ohrožení se nacházejí především lidské životy. Takovéto aplikace se označují jako kritické z hlediska bezpečnosti (safety-critical) a představují je oblasti automobilového průmyslu, letectví, kosmonautiky či medicíny.

V případě, že cílíme na systémy, které neobsahují návrhové ani implementační chyby, které by mohly způsobit nekorektní chování, musí být tyto systémy důkladně otestovány. V úvahu se berou obvyklé, ale i neobvyklé kombinace vstupních hodnot, které mohou v daném systému nastat. Jelikož neustále roste složitost systému, tak roste i složitost spojená s důkladným ověřením jeho správné funkce [1]. Jednoduché systémy není složité otestovat manuálně. U komplexnějších systémů je manuální testování velmi časově náročné. Rovněž doposud vyvinuté formální techniky pro ověření rozsáhlých systémů selhávají. Z tohoto důvodu byla vytvořena technika zvaná funkční verifikace, která na základě vstupních a výstupních hodnot ze systémů ověřuje jejich korektnost. Hodnoty, které vstupují do systému, se typicky získávají pomocí generátoru stimulů (testů), který je předmětem této práce.

V současnosti existuje řada nástrojů, které jsou schopny generovat vstupní stimuly [2,3]. Tyto nástroje se ale zaměřují na jeden konkrétní systém, typicky procesor typu RISC. Systém pro generování stimulů, který by byl schopný uplatnit se v různorodých oblastech, neexistuje, proto je naším cílem takovýto systém vytvořit. Pomocí tohoto systému chceme definovat závěry v podobě obecných principů konstrukce testovacích stimulů pro různé systémy.

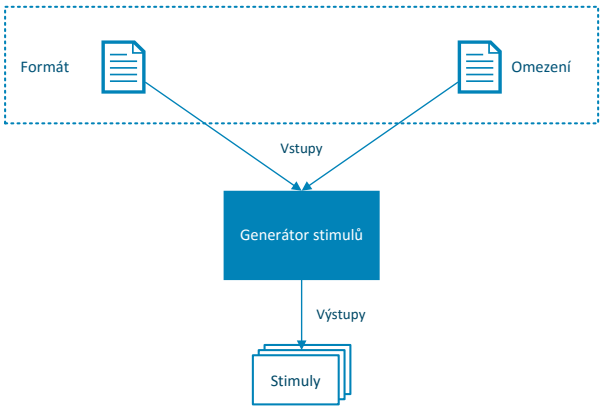
II. Cíle Práce

Hlavními cíli této práce jsou:

1. Navrhnout a vytvořit univerzální generátor testovacích stimulů založený na řešení problému s omezujícími podmínkami, který bude vhodný především pro použití ve funkční verifikaci.
2. Navrhnout obecný a jednotný popis stimulů různých systémů, pomocí něhož lze popsat veškeré podmínky a zákonitosti nutné k vygenerování platného testovacího stimulu. Výsledkem této činnosti bude vytvoření obecných principů konstrukce testovacích stimulů pro různé systémy.

III. Generátor Stimulů

Princip generování stimulů je založen na námi navržené architektuře [4], která je znázorněna na Obrázku 1. Tato architektura má za cíl zjednodušit a urychlit vytváření stimulů pro různé systémy.



Obr. 1. Architektura námi navrženého generátoru stimulů.

Základní idea je založena na dvou specifických vstupních strukturách, které definují formát generovaných dat (*Formát*) a omezující podmínky (*Omezení*) určující, jak má být s těmito formáty nakládáno při jejich generování. Tyto dva popisy představují vstup do *Generátoru stimulů*, který na základě jejich obsahu generuje validní stimuly na jeho výstup. Tyto stimuly pak již mohou být předány konkrétnímu systému jako vstupní data. Uživatel při vytváření vlastních stimulů neprogramuje žádný kód, ale pouze specifikuje požadovaný formát stimulů a omezení, které generátor na základě řešení problému s omezujícími podmínkami vyřeší. Důležitým předpokladem pro tento popsaný princip generování je široká množina vstupní abecedy, která musí obecně popisovat problémy a omezení různých stimulů. Tento princip generování stimulů je parametrizovatelný. Dokáže za běhu zpracovávat a měnit omezující podmínky změnou vstupních struktur, a tudíž je vhodný pro použití v procesu funkční verifikace.

A. *Formát stimulu*

Formát stimulu popisujeme pomocí tří základních části: *Nahrazení*, *Proměnné* a *Syntaxe*.

Část *Nahrazení* definuje identifikátor a všechny možné substituce, za které je možno daný identifikátor nahradit. Jedná se o obdobu výčtového datového typu. Nahrazuje se za konkrétní řetězec z definované množiny hodnot. V každém novém cyklu generování dochází k náhodnému zvolení určité substituce pro daný identifikátor.

Část *Proměnné* definuje proměnné v obecném slova smyslu. Pro každou proměnnou je přiřazena náhodná hodnota v závislosti na jejím datovém typu a to v každém cyklu generování.

Část *Syntaxe* syntakticky popisuje řetězce, jeden po druhém, které mají být náhodně generovány na výstup generátoru jako součást stimulu. V jednotlivých definovaných řetězcích se mohou nacházet identifikátory z částí *Nahrazení* a *Proměnné*. Tyto identifikátory jsou nahrazeny za konkrétní nebo náhodnou hodnotu v závislosti na typu identifikátoru. Část *Syntaxe* představuje statické hodnoty v generovaném řetězci, zatímco zbylé dvě části představují dynamické (měnící se) hodnoty v generovaném řetězci.

B. *Omezení*

Omezeními povolujeme generování pouze těch stimulů, které jsou platné pro zvolený systém. Jedná se především o omezení pro datové hodnoty (proměnná může nabývat pouze hodnot z určitého rozsahu) nebo závislostní omezení (některá kombinace hodnot nemůže nastat po aktuálně generované kombinaci). Omezení jsou unikátní pro každý systém, a tedy různá omezení jsou aplikována na různé systémy.

C. *Generátor stimulů*

Generátor stimulů kombinuje *Syntaxe*, *Nahrazení* a *Proměnné* tak, že všechna omezení jsou splněna - žádné není porušeno. Výstupem generátoru je posloupnost řádků, která odpovídá definovanému problému a která tvoří výsledný stimul.

IV. Generování Programů

Generování assemblerovského programu pro procesory je prvním příkladem použití tohoto generátoru. Vstupní struktury byly vytvořeny speciálně pro procesor Codix RISC [5], ale úpravou lze popsat program pro jiný procesor obdobného typu.

A. *Formát*

Část *Syntaxe* obsahuje instrukční sadu zvoleného procesoru. Každá definovaná instrukce v této části se skládá z identifikátoru a své assemblerovské reprezentace. Identifikátor slouží jako odkaz mezi instrukcí a omezeními. Připravené assemblerovské reprezentace instrukcí v sobě obsahují další identifikátory, které jsou definovány a nahrazovány z částí *Nahrazení* (ty představují především registry) a *Proměnné* (ty představují především náhodná čísla a řetězce). Příklad definice instrukce OR:

```
ori { "dst = or src1, imm" }
```

kde *ori* je identifikátor instrukce, řetězec ve složených závorkách je assemblerovská reprezentace instrukce, *dst* a *src1* jsou identifikátory z části *Nahrazení* a *imm* je jméno proměnné. Pro procesor Codix RISC bylo potřeba vytvořit 61 takovýchto definic instrukcí.

Část *Nahrazení* definuje množinu řetězců, za které může být nahrazena určitá část assemblerovské reprezentace instrukce. Tato část je použita především pro definici dostupných registrů procesoru. Příklad definice jednoho nahrazení:

```
dst { r0|r1|r2 }
```

kde *dst* je identifikátor nahrazení a *r0*, *r1*, a *r2* jsou možné řetězce, kterými je identifikátor nahrazován.

Část *Proměnné* definuje proměnné, do kterých je přiřazena náhodná hodnota v závislosti na jejich datovém typu. Tato část je použita pro definici a přiřazení přímých operandů anebo řetězců jako návěští skokových instrukcí. Příklad definice jedné proměnné:

```
VAR16 imm
```

kde datový typ *VAR16* určuje 16-bitové celé číslo a *imm* je jméno proměnné.

B. *Omezení*

Pro běžné instrukce, které nevyžadují zvláštní předzpracování, omezení hodnot či kooperaci s jinými instrukcemi, není zapotřebí definovat žádná omezení. V případě, že není žádné omezení pro danou instrukci definováno, znamená to, že se tato instrukce může nacházet kdekoliv v programu a to po jakékoliv instrukci. Některé instrukce ale omezení vyžadují, aby byla zajištěna jejich platná sekvence a jejich platné hodnoty operandů. Proto bylo vytvořeno několik omezení, které řeší typické problémy při generování assemblerovského kódu. Těchto omezení pro procesor Codix RISC je celkem 18. Příklad definice několika omezení:

```
end("nop//halt")     # pro ukončení programu
nouse(add(dst),1)    # pro latence
```

```
contain(label(name),jump(name)) # pro skoky
unique(label(name)) # pro skoky
mdiv(imm,4)         # pro zarovnání paměti
```

C. *Shrnutí*

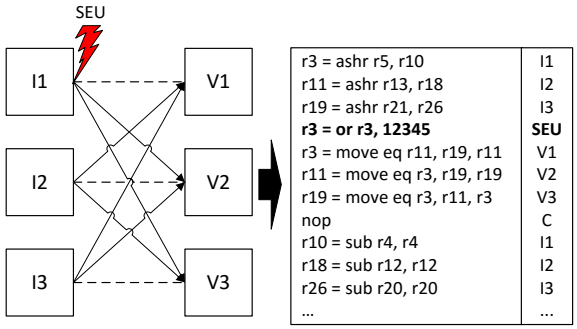
Princip popsaný v této kapitole byl využit v praxi v komerční společnosti a publikován v časopise [6].

V. Generování Softwarově Implementované Odolnosti Proti Poruchám

Základní idea je založena na informační redundanci, která je přidána do assemblerovského programu. Využili jsme princip *Triple Modular Redundancy* (TMR) v softwaru a definovali jsme proto obdobu *Triple Instructional Redundancy* (TIR). TIR pracuje na principu ztrojení jednotlivých instrukcí programu a vyhodnocení majority. Tím dojde k detekci a opravě chyby v registrech procesoru nebo paměti. Korektní funkčnost a odolnost samotného programu jsme ověřili pomocí softwarové injekce poruchy.

A. *Softwarová injekce poruchy*

Simulace poruchy v softwaru je provedena vložením nezabezpečené instrukce do zabezpečeného programu. Tímto jsme schopni přepsat hodnotu jednoho paměťového prvku a následně jednonásobnou poruchu opravit se 100% úspěšností. Tato operace je ekvivalentní se *single event upset* (SEU) poruchou. Pomocí této injekce jsme schopni rovněž vygenerovat více poruch na různá místa programu a tak simulovat více poruch. Příklad injekce do TIR programu je ukázán na Obrázku 2.

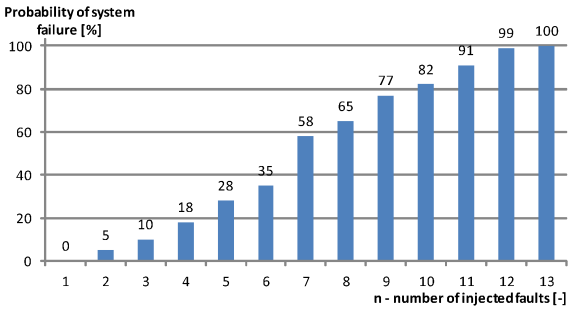


Obr. 2. Příklad softwarové injekce poruchy do zabezpečeného programu.

B. *Generování TIR programu*

Každá instrukce v části Syntaxe je ztrojena ve vstupní struktuře definující formát generovaného programu a je následována trojicí porovnávacích instrukcí *move*, které představují voliče pro vyhodnocení majority. Omezení obsahují speciální definice zajišťující správné rozdělení a používání registrového a paměťového prostoru, aby si jednotlivé TIR instrukce nepřepisovaly své výsledky. SEU porucha je generována instrukcí bez ztrojení. Pro ověření správnosti TIR programu jsme generovali jak zabezpečený program, tak i nezabezpečený.

V rámci experimentů jsme generovali až 13 poruch v rámci jednoho programu o 100 instrukcích. Poruchy byly umísťovány náhodně mezi TIR instrukce a voliče. Výsledek určující procentuální selhání programu při vícenásobné poruše ukazuje Obrázek 3.



Obr. 3. Pravděpodobnost selhání systému pokud je injektováno *n* poruch.

C. *Shrnutí*

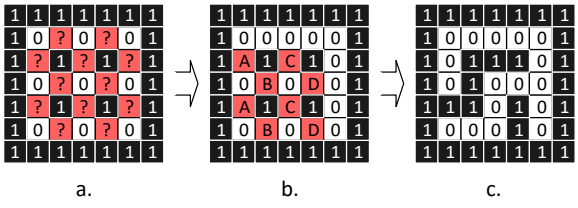
Tento přístup byl ověřen pro procesor Codix RISC a byl publikován na konferenci [7].

VI. Generování Bludiště

Generování bludiště je velmi známá a prozkoumaná oblast, pro kterou existuje značné množství algoritmů, jak generovat jednoduchá nebo sofistikovanější bludiště. Drtivá většina algoritmů pracuje ve dvourozměrném prostoru, uchovávají si aktuální stav a dokáží neustále měnit hodnoty buněk bludiště v čase. Takovéto algoritmy jsou pro námi navrženou architekturu univerzálního generátoru značně nevhodné, jelikož výstup generátoru (v našem případě se jedná o řádek bludiště) není možno určit v jednom kroku, ale je určen mnoha faktory a závislostmi mezi různými buňkami bludiště. Existuje ovšem algoritmus, který je založen na binárním stromu a konkrétní řádek bludiště lze určit jen a pouze z předcházejícího řádku. Takovýto princip je pro náš generátor zcela vyhovující a výstupní bludiště zcela dostačuje pro naše potřeby.

A. *Základní princip*

Základní princip algoritmu je ukázán na Obrázku 4. Vychází se ze základní matice bludiště (a), u které jsou některá pole pevně zadaná – buď chodba, nebo stěna. Chodby reprezentujeme pomocí nul, kdežto stěny pomocí jedniček. Pole označená otazníkem reprezentují oblasti, které mohou nabývat hodnotu 0 nebo 1, tedy chodba nebo stěna. Aby byla zachována průchodnost bludiště z jakékoliv chodby do jiné, je potřeba provést modifikaci základního bludiště tak, že vždy dvě přilehlé strany bludiště musí obsahovat chodbu přes celou jeho šíři (b). V našem případě jsme si zvolili tuto chodbu na severní a východní straně bludiště. Posledním nejkritičtějším úkolem je určit buňky A,B,C,D tak, aby bludiště bylo spojité (c).



Obr. 4. Ukázka převodu základního bludiště pro potřeby generátoru.

Originální popis algoritmu rozdělí buňky bludiště na daném řádku do skupin chodeb ohraničených zdmi. Pro každou takovouto skupinu algoritmus určí jeden vchod buď v severní, nebo východní části ohraničení skupiny. Tímto způsobem je zajištěn průchod ze severní části bludiště do jižní a to samé platí pro průchod ze západu na východ. Tento princip jsme převedli do podoby závislostí jednoho řádku bludiště na druhém a výsledkem je následující závislost. Pokud na Obrázku 4.b buňka A respektive C byla náhodně vybrána za chodbu, pak buňka B respektive D bude zdí a naopak.

B. *Aplikování na generátor*

Struktura popisující formát generování v tomto případě definuje množinu hodnot a požadovaný výstup – nuly a jedničky. Struktura s omezeními zahrnuje podmínky vycházející z předchozího odstavce, které jsou nutné pro spojité generování bludiště. Ukázky jednoduchosti jednotlivých popisů bez dalšího vysvětlení, které dostačují k vygenerování bludiště o rozměrech 7x7 buněk jsou níže:

```
----- Formát -----

substitute {
  A,C { "0"|"1" }
  B,D { "0" }
}

syntax {
  odd { "1A1C101" }
  even { "10B0D01" }
}
```

```
----- Omezení -----

constraints {
  nlines(7,7)
  ifthen(A("0"),B("1"))
  ifthen(C("0"),D("1"))
  start("1111111")
  start("1000001")
  useonly(odd)
  afterinsert(odd,even)
  end("1111111")
}
```

Drobnou modifikací se lze dostat na jakýkoliv požadovaný rozměr bludiště. Aby mohl být použit předpoklad základní matice bludiště, je nutné uvažovat liché rozměry bludiště.

C. *Shrnutí*

Generováním bludiště jsme ukázali další možnou oblast v generování pomocí námi navržené architektury generátoru. Vygenerovaná bludiště jsme použili jako vstup pro verifikaci řídicí jednotky robota, která hledá cestu v bludišti. V rámci práce byla řídicí jednotka ověřována na správnou funkcionalitu pro různé tvary a rozměry bludišť. Tento přístup byl opublikován na konferenci [8].

VII. Budoucí Práce

V rámci budoucí práce pracujeme na zobecnění v samotném generování stimulů, které bude možné aplikovat pro různé systémy. Tato práce zahrnuje modifikaci architektury a popsání stimulů pomocí existujícího aparátu. Jako vhodný aparát se jeví použití gramatických systémů, které principiálně již částečně využíváme v definici vstupní struktury generátoru. Určité omezující podmínky ale budou nadále nutné, aby byla zajištěna validnost vygenerovaného stimulu.

V současné funkcionalitě generátor podporuje několik omezení pro generování programů pro procesory typu RISC, VLIW, softwarově implementovanou odolnost proti poruchám a bludiště. Zobecněním vstupních popisů by proto práce nabyla jiného rozměru a našla by uplatnění v dalších systémech v řadě oblastí.

VIII. Závěr

V práci jsme ukázali architekturu námi navrženého generátoru stimulů, která je založena na dvou vstupních strukturách. Pomocí těchto struktur jsme byli schopni nadefinovat popisy pro generování assemblerovských programů pro procesor typu RISC. Takovýto program jsme byli rovněž schopni zabezpečit pomocí softwarově implementované odolnosti ztrojením instrukcí a přidáním voličů pro vyhodnocení majority v paměťových prvcích procesoru. V poslední části práce jsme se věnovali generováním bludišť pro řídicí jednotku robota.

Ukázali jsme široké použití generátoru stimulů v různých oblastech použití a nastínili naše kroky v další výzkumné činnosti. Naše výsledky jsou řádně zdokumentovány díky řadě publikací na konferencích ve světě. V další činnosti pracujeme na zobecnění popisu a samotného generování stimulů.

Poděkování

Tato práce byla podpořena Ministerstvem školství, mládeže a tělovýchovy z Národního programu udržitelnosti (NPU II); projekt IT4Innovations excellence in science (IT4I XS - LQ1602), ARTEMIS JU na základě grantové dohody č. 641439 (ALMARVI) a projektu Vysokého učení technického v Brně FIT-S-14-2297.

Reference

[1] Roy, S.; Ramesh, S.: Functional verification of system on chips - practices, issues and challenges. In Proceedings of ASP-DAC 2002, 2002, s. 11-13, doi:10.1109/ASPDAC.2002.994873.

[2] Belkin, V.; Sharshunov, S.: ISA Based Functional Test Generation with Application to Self-Test of RISC Processors. In Design and Diagnostics of Electronic Circuits and systems, 2006 IEEE, duben 2006, s. 73-74, doi:10.1109/DDECS.2006.1649575.

[3] Hudec, J.: An effient technique for processor automatic functional test generation based on evolutionary strategies. In Proceedings of the ITI 2011, 33rd International Conference on Information Technology Interfaces, červen 2011, ISSN 1330-1012, s. 527-532.

[4] Podivínský, J.; Čekan, O.; Šimková, M.; Kotásek, Z.: The Evaluation Platform for Testing Fault-Tolerance Methodologies in Electro-mechanical Applications. In: 17th Euromicro Conference on Digital Systems Design. Verona: IEEE Computer Society, 2014, s. 312-319. ISBN 978-1-4799-5793-4.

[5] Codasip. (2013) Codasip - codix-risc. [Online]. Dostupné z: www.codasip.com/products/codix-risc/

[6] Podivínský, J.; Čekan, O.; Šimková, M.; Kotásek, Z.: The Evaluation Platform for Testing Fault-Tolerance Methodologies in Electro-mechanical Applications. Microprocessors and Microsystems. Amsterdam: Elsevier Science, 2015, roč. 39, č. 8, s. 1215-1230. ISSN 0141-9331.

[7] Čekan, O.; Podivínský, J.; Kotásek, Z.: Software Fault Tolerance: the Evaluation by Functional Verification. In: Proceedings of the 18th Euromicro Conference on Digital Systems Design. Funchal: IEEE Computer Society, 2015, s. 284-287. ISBN 978-1-4673-8035-5.

[8] Podivínský, J.; Čekan, O.; Lojda, J.; Kotásek, Z.: Verification of Robot Controller for Evaluating Impacts of Faults in Electro-mechanical Systems. In: Proceedings of the 19th Euromicro Conference on Digital Systems Design. IEEE Computer Society, 2016, přijato k publikaci.

Funkční verifikace jako nástroj pro sledování vlivu poruch na elektro-mechanický systém

Jakub Podivínský
3. ročník, prezenční studium
Školitel: Zdeněk Kotásek

Fakulta informačních technologií, Vysoké učení technické v Brně
Božetěchova 2, 612 66 Brno
ipodivinsky@fit.vutbr.cz

***Abstrakt*—Náplní tohoto článku je představení práce zabývající se využitím techniky funkční verifikace jako nástroje pro ověřování metodik pro zajištění odolnosti proti poruchám v systémech založených na FPGA. V tomto článku jsou představeny cíle disertační práce vycházející z aktuálního stavu poznání v řešené oblasti. Představen je také návrh řešení, jehož součástí je rozdělení procesu ověřování do tří fází: (1) klasická funkční verifikace, (2) funkční verifikace využívající FPGA a injektor poruch a (3) sledování vlivu poruch na mechanickou část. V této práci bude věnována pozornost zejména první a druhé fázi, které byly předmětem zkoumání v uplynulém roce. Jak již název článku napovídá, budou zde prezentovány experimenty využívající verifikační prostředí z první a druhé fáze pro sledování vlivu poruch na elektronickou část experimentálního elektro-mechanického systému.**

***Klíčová slova*—Funkční verifikace, odolnost proti poruchám, elektro-mechanický systém, řídicí jednotka robota.**

I. Úvod a motivace

Číslicové systémy hrají důležitou roli v našem každodenním životě, setkáváme se s nimi téměř v každé situaci a jejich výskyt se stále rozšiřuje. Řada z těchto číslicových systémů je využívána v prostředích, kde může docházet ke zvýšenému výskytu poruch a tyto poruchy mohou mít nedozírné následky ve formě finančních a materiálních ztrát, ale mohou ohrozit i lidské zdraví a životy. Může se jednat o různé aplikace v leteckém či vesmírném průmyslu, nebo o systémy využívané v lékařství, jejichž selhání může ohrozit zdraví pacientů.

Naše práce je zaměřená na FPGA (Programovatelná hradlová pole) využívající SRAM paměť, protože tyto obvody jsou stále populárnější a to především díky jejich vysokému výkonu a rekonfigurovatelnosti, která zajišťuje jejich vysokou flexibilitu. Obvody FPGA jsou složeny z konfigurovatelných logických bloků (CLB), které jsou propojeny konfigurovatelnou propojovací sítí. Konfigurace těchto prvků je uložena v SRAM paměti ve formě tzv. *bitstreamu*. Nevýhodou z hlediska spolehlivosti je vyšší náchylnost FPGA na poruchy způsobené nabitými částicemi. Tyto částice mohou způsobit inverzi bitu v *bitstreamu* a tím způsobit změnu chování celého obvodu. Tyto poruchy jsou nazývány *Single Event Upset* (SEU) [1]. Pro eliminování vlivu poruch se používají techniky jako *předcházení poruchám* nebo *odolnost proti poruchám* [2]. Odolnost proti poruchám (FT) je vlastnost systému, která říká, že tento systém je schopen vykonávat svou činnost i v přítomnosti poruchy. Existuje řada metodik pro zajištění odolnosti proti poruchám u systémů založených na FPGA a další jsou předmětem výzkumu na různých pracovištích [3].

Důležitou činností je testování, ověřování a porovnávání vlastností těchto metodik pro zajištění odolnosti proti poruchám. Existují nejrůznější přístupy, některé jsou spíše na teoretické úrovni, jako například simulační metoda pro emulaci SEU představená v [4]. Jiné přístupy naopak používají injekci poruch přímo do FPGA. Jako příklad poslouží speciální deska vyvinutá pro tyto účely prezentovaná autory v [5]. Ve všech případech se jedná o vyhodnocování odolnosti proti poruchám se zaměřením pouze na elektronickou část, v naší práci se chceme zaměřit také na vyhodnocování vlivu poruch na mechanickou část, kterou elektronické systémy v řadě případů řídí.

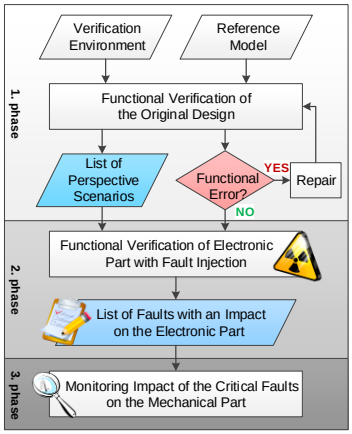
II. Cíle disertační práce a návrh řešení

V rámci své disertační práce bych rád nalezl odpověď na několik otázek: *Jaké budou výsledky FT metodik na reálných systémech? Lze spoléhat na to, že ne všechny poruchy v elektronické části systému se projeví na chování řízené mechanické aplikace? Lze využít funkční verifikaci pro ověřování vlivu poruch na FT systémy?* Na základě těchto otázek byly formulovány cíle disertační práce:

- Navrhnout a vytvořit platformu, která bude založená na technologii FPGA a bude sloužit k testování FT metodik a k sledování vlivu poruch nejen na výstup elektronické části, ale také na řízenou mechanickou aplikaci. Bude využita technika funkční verifikace a injektor poruch [6] vyvinutý týmem doc. Kotáska. Jádrem platformy bude experimentální elektromechanická aplikace.
- V návaznosti na vytvořenou testovací platformu navrhnout proces ověřování FT metodik s přihlédnutím ke specifikům elektromechanických systémů. Tyto postupy budou využívat navrženou a vytvořenou testovací platformu. Proces bude navržen na základě experimentování s vytvořenou platformou. Součástí bude popis činností před zahájením procesu ověřování. Proces bude ověřen a demonstrován na dalším experimentálním systému. Tím dojde k zobecnění získaných výsledků.

V rámci dosavadní práce byl navržen proces ověřování metodik pro zajištění odolnosti proti poruchám, který je aktuálně rozdělen do tří fází znázorněných na Obrázku 1. V *první fázi*

proběhne klasická funkční verifikace experimentálního sytému tak, aby byl zajištěn jeho soulad se specifikací. Výstupem této fáze je odladěný experimentální systém a sada vygenerovaných verifikačních scénářů, pro které byla verifikace prováděna. Ve *druhé fázi* je prováděna funkční verifikace s využitím FPGA, tak aby řídicí elektronika běžela přímo na FPGA a bylo do ní možné injektovat poruchy. V této fázi jsou využity dříve získané verifikační scénáře, pro které je jisté, že se experimentální systém bez výskytu poruch chová správně. Můžeme si tak být jisti, že případné nekorektní chování způsobila injektovaná porucha a ne chyba v implementaci. V poslední *třetí fázi* dochází k detailnímu ověřování situací, kdy se injektovaná porucha projevila na výstupu řídicí elektroniky a je vyhodnocován projev této poruchy na mechanickou část. Pro každou fázi je potřeba specifické verifikační prostředí, v tomto článku bude pozornost věnována první a druhé fázi a budou představena použitá verifikační prostředí pro náš experimentální elektromechanický systém. Jako experimentální systém používáme robota pro hledání cesty v bludišti a jeho řídicí jednotku.

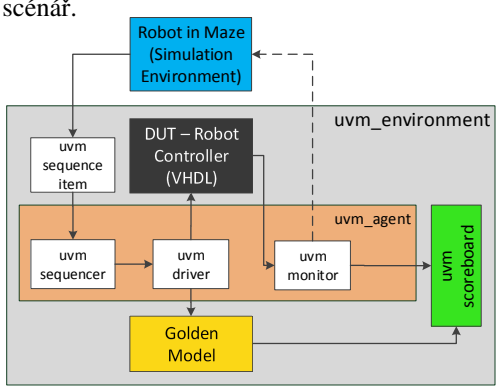


Obrázek 1. Tři fáze procesu ověřování odolnosti proti poruchám.

III. První fáze - verifikační prostředí pro řídicí jednotku robota

V první fázi navrženého procesu ověřování odolnosti proti poruchám je využívána klasická funkční verifikace založená na simulaci verifikovaného obvodu. Klasická funkční verifikace ověřuje, zda systém odpovídá specifikaci monitorováním jeho vstupů a výstupů v simulačním prostředí (např. *ModelSim*). V našem případě je verifikovaným obvodem (DUT) řídicí jednotka robota pro hledání cesty v bludišti. V první fázi využíváme verifikační prostředí, které je implementováno podle metodiky UVM (Universal Verification Methodology), což znamená, že odpovídá současným trendům a požadavkům. Toto verifikační prostředí je připraveno pro vyhodnocování jednoho verifikačního scénáře (jedno bludiště, počáteční a cílová pozice) a tvoří základ rozšiřujícího prostředí pro vyhodnocování množiny verifikačních scénářů.

Implementované verifikační prostředí je zobrazeno na Obrázku 2, kde jsou patrné základní komponenty odpovídající metodice UVM. Za pozornost stojí mechanická část, přesněji její simulace, která je řízena výstupy DUT a zároveň generuje nové vstupy pro DUT. V našem případě využíváme volně dostupné simulační prostředí pro simulaci robota Player/Stage [7]. Součástí verifikačního prostředí jsou, mimo DUT a simulaci mechanické části, také další komponenty. O produkování referenčních výstupů řídicí jednotky se stará referenční (*golden*) model, který byl implementován v rámci diplomové práce [8] v jazyce C/C++. Komponenta *sequence* získává data ze senzorů z robota v simulačním prostředí a transforumuje je na transakce pro DUT, které dále předává komponenta *driver* jako vstup DUT a referenčního modelu. Naopak *monitor* čte výstupní data z DUT (směr a rychlost pohybu robota) a předává je (1) zpět do simulačního prostředí robota, který provede zadaný pohyb, a (2) do komponenty *scoeboard* k porovnání s výstupem referenčního modelu. Výstupem tohoto verifikačního prostředí je informace o tom, zda DUT pro tento verifikační scénář respektuje specifikaci či nikoliv, a zároveň také informace o pokrytí kódu (*codecoverage*) pro daný verifikační scénář.



Obrázek 2. Verifikační prostředí pro jeden verifikační scénář.

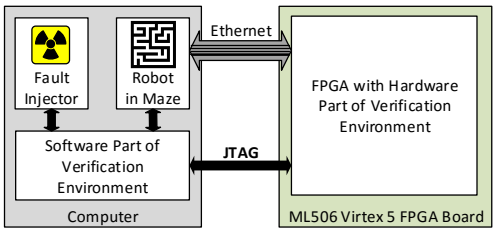
Představené verifikační prostředí není schopno automaticky vyhodnotit více verifikačních scénářů, tuto činnost zajišťuje rozšířený proces ověřování. Verifikační prostředí tvoří jednu z komponent toho procesu. Důležité je také generování bludiště, kdy pro každý verifikační běh je vygenerováno nové bludiště. Po každém běhu verifikačního prostředí je uložen ověřený verifikační scénář uložen spolu s informací o dosaženém pokrytí kódu. Po skončení celého procesu ověření množiny verifikačních scénářů jsou sloučeny jednotlivé dílčí informace o pokrytí kódu a výsledkem je jediný report.

IV. Druhá fáze - architektura testovací platformy

V druhé fázi je využíváno verifikační prostředí, které pro implementaci DUT využívá FPGA, což umožňuje injekci poruch přímo do reálného obvodu. Pro potřeby provozování verifikačního prostředí využívajícího FPGA jsme navrhli platformu složenou z několika komponent běžících na FPGA nebo na počítači:

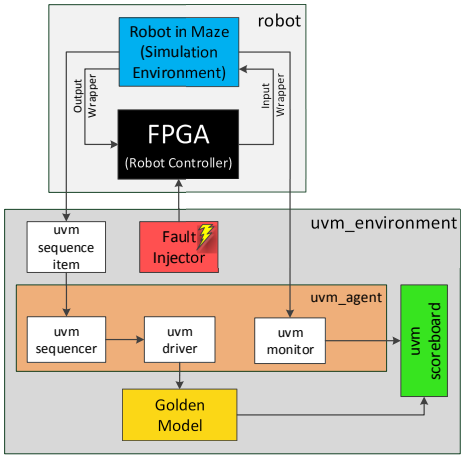
- SW část verifikačního prostředí běžící na počítači,
- SW simulační prostředí pro simulaci robota (Player/Stage) běžící na počítači,
- řídicí jednotka robota implementovaná na FPGA,
- externí injektor poruch [6] běžící na počítači, který umožňuje simulovat poruchy přímo v FPGA.

Celkovou architekturu této platformy zachycuje Obrázek 3. Mimo zmíněných komponent jsou zde patrné také komunikační kanály mezi FPGA a počítačem. Pro komunikaci mezi řídicí jednotkou robota a robotem je použito rozhraní Ethernet. Rozhraní JTAG je určeno pro programování FPGA a následně také pro injekci poruch do FPGA.



Obrázek 3. Struktura experimentální platformy.

Verifikační prostředí využívající FPGA (Obrázek 4) vychází z velké části z výše uvedeného klasického verifikačního prostředí. Zobrazené verifikační prostředí je rozdělené na dvě části, první částí je simulovaný robot komunikující s řídicí jednotkou na FPGA. Tato část pracuje zcela autonomně, mezi FPGA a simulačním prostředím proudí informace ze senzorů a pokyny k pohybu robota, v simulačním prostředí lze sledovat pohyb robota. Druhou částí je samotné verifikační prostředí, které převzalo většinu komponent z původního prostředí. Toto verifikační prostředí zde funguje jako pozorovatel, což znamená, že pouze pasivně sleduje komunikaci mezi FPGA a simulací robota a porovnává je s výstupy referenčního modelu a nijak do této komunikace nezasahuje.



Obrázek 4. Architektura verifikačního prostředí využívajícího FPGA.

Stejně jako v první fázi, i zde je třeba zajistit automatické vyhodnocování množiny verifikačních scénářů spolu s injekcí poruch. V tomto procesu již nedochází ke generování bludiště, jelikož ve druhé fázi se pracuje s množinou vygenerovaných a ověřených bludišť. Po naprogramování FPGA, spuštění simulace robota a verifikačního prostředí robot začíná procházet bludištěm. A právě zde nastává prostor pro injekci poruch do FPGA. Poruchy mohou být injektovány dle zvolené strategie, například jednonásobné poruchy do vybrané funkční jednotky, vícenásobné poruchy do jedné nebo více funkčních jednotek a podobně. Výstupem tohoto procesu je report shrnující provedené experimenty, pro každý běh je uložena také pozice injektované poruchy pro použití v další fázi.

Pro potřeby injekce poruch máme vyvinutý vlastní injektor poruch [6], který umožňuje injektovat poruchu do zadaného bitu bitstreamu. Tento injektor využívá rozhraní JTAG a částečnou dynamickou rekonfiguraci, kdy nejprve vyčte část používaného bitstreamu, modifikuje zadaný bit a uloží tento upravený bitsream zpět do FPGA. Pro zjištění relace mezi bitem bitstreamu a funkční jednotkou používáme nástroj RapidSmith , který umí analyzovat FPGA a najít seznam bitů bitstreamu odpovídající využitým LUT tabulkám v zadané oblasti FPGA. Díky rozmístění funkčních jednotek na FPGA pomocí nástroje PlanAhead víme, v které části FPGA se nachází jednotlivé funkční jednotky.

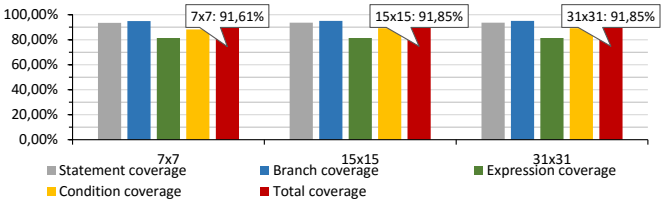
V. Experimentální výsledky

Doposud prováděné experimenty s představenými verifikačními prostředími korespondují s první a druhou fází navrženého procesu ověřování odolnosti proti poruchám. V první fázi jsme prováděli funkční verifikaci řídicí jednotky robota s využitím představeného verifikačního prostředí. Výstupem této fáze je (1) odladěná řídicí jednotka robota, (2) množina použitých verifikačních scénářů a (3) report informující o dosaženém pokrytí kódu. Pro experimenty v této fázi jsme použili tři rozměry bludiště: 7x7, 15x15 a 31x31 políček. Průměrný počet kroků, které musí robot vykonat, zachycuje Tabulka I.

Tabulka I
Průměrný počet kroků robota

Velikost bludiště	7x7	15x15	31x31
Průměrný počet kroků	16	93	433

V průběhu této fáze jsme sledovali zejména vliv počtu vygenerovaných bludišť a velikost bludiště na dosažené pokrytí kódu. Provedli jsme funkční verifikaci pro 10, 100, 200 a 500 verifikačních scénářů pro každý rozměr bludiště. Narůstající počet verifikačních scénářů neměl téměř žádný vliv na dosažené pokrytí . Sloupcový graf 5 zobrazuje dosažené pokrytí kódu (jednotlivé dílčí složky a celkové pokrytí) pro 100 verifikačních běhů pro různé rozměry bludiště. Z tohoto grafu je patrné, že ani rozměr bludiště neměl podstatný vliv na dosažené pokrytí, navzdory původnímu předpokladu.



Obrázek 5. Pokrytí kódu pro 100 verifikačních běhů s bludišti různých velikostí.

Ve druhé fázi byly provedeny experimenty využívající verifikační prostředí s FPGA a injektor poruch. Experimenty byly prováděny s řídicí jednotkou robota bez aplikace metodik pro zajištění odolnosti proti poruchám, jejich cílem byla (1) analýza řídicí jednotky robota z pohledu spolehlivosti a (2) demonstrace funkčnosti vytvořené testovací platformy.

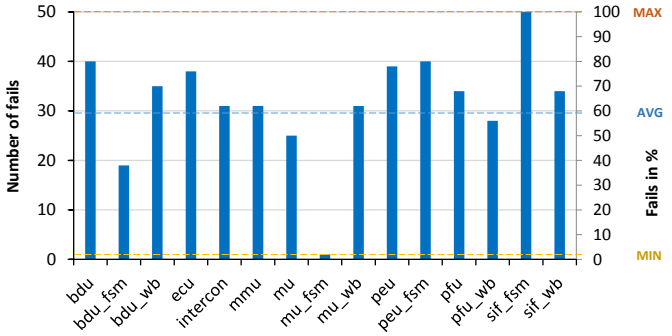
Bylo provedeno 50 verifikačních běhů pro každou funkční jednotku s verifikačními scénáři získanými během předchozí

fáze, přesněji se jednalo o bludiště o rozměrech 15x15 políček, která nám zajistila dostatek času pro injekci poruchy a následné sledování jejího vlivu před tím, než robot dorazil do cíle. Řídicí jednotka robota obsahuje 15 funkčních bloků, dohromady tedy bylo provedeno 750 verifikačních běhů. Tento počet byl zvolen především z časových důvodů, jelikož jeden krok robota trvá přibližně 5s, projití celého bludiště pak zabere cca 10minut. Pro injekci poruch jsme si zvolili strategii injekce jednonásobné poruchy do vybrané funkční jednotky během jednoho verifikačního scénáře. Tabulka II shrnuje získané výsledky, pro každý funkční blok je zde uveden počet poruch (z celkového počtu 50 injektovaných poruch), které způsobily neshodu mezi výstupem řídicí jednotky robota a referenčním modelem. Uvedeno je také procentuální vyjádření.

Tabulka II
EXPERIMENTÁLNÍ VÝSLEDKY: VLIV PORUCH NA VÝSTUP ELEKTRONIKY.

Funkční blok	Počet poruch [-]	Počet poruch [%]	Funkční blok	Počet poruch [-]	Počet poruch [%]
bdu	40	80	mu_wb	31	2
bdu_fsm	19	38	peu	39	78
bdu_wb	35	70	peu_fsm	40	80
ecu	38	76	pfu	34	68
intercon	31	62	pfu_wb	28	56
mmu	31	62	sif_fsm	50	100
mu	25	50	sif_wb	34	68
mu_fsm	1	2			

Výsledky experimentů shrnuje také Obrázek 6, ze kterého je jasně patrné, že některé funkční bloky jsou na poruchy citlivější než jiné. V grafu je zakreslena průměrná hodnota, která je přibližně 60%. Některé bloky se odchylují méně, jiné více. Největší odchylka od průměru je patrná u bloku *sif_fsm*, kde byla dosažena hodnota 100%, naopak u bloku *mu_fsm* se projevily pouze 2%. U těchto bloků jsme se rozhodli zopakovat experimenty s použitím většího počtu verifikačních scénářů. Pro každý tento blok jsme provedli dalších 225 verifikačních běhů. Získané výsledky se přiblížily průměrným hodnotám.



Obrázek 6. Experimentální výsledky: vliv poruch na výstup elektroniky.

Závěr z našich experimentů je jednak zjištění, které funkční bloky jsou více či méně náchylné k poruchám, ale také ke zjištění, že jsme schopni pomocí funkční verifikace doplněné injekcí poruch do FPGA ověřovat vliv poruch na elektromechanický systém, přesněji na jeho elektronickou část (řídicí jednotka robota).

VI. ZÁVĚR A DALŠÍ PRÁCE

V této práci byl představen posun v řešení disertační práce na téma ověřování metodik pro zajištění odolnosti proti poruchám pomocí kombinace funkční verifikace a injekce poruch do FPGA. Byl zde představen návrh procesu pro ověřování odolnosti proti poruchám rozděleného do tří fází, kdy každá fáze vyžaduje specifické verifikační prostředí. V tomto článku bylo představeno verifikační prostředí odpovídající první (klasická funkční verifikace) a druhé (verifikační prostředí využívající FPGA) fázi navrženého procesu. Tato verifikační prostředí byla také předmětem experimentů, jejich výsledky byly v článku také stručně představeny. Představené výsledky byly průběžně publikovány na mezinárodních konferencích a v odborném časopise, výčet publikací bude součástí prezentace.

V budoucí práci se plánujeme zaměřit jednak na třetí fázi navrženého procesu, tedy na automatizaci sledování vlivu poruch na mechanickou část. Musíme vytvořit další verifikační prostředí, které bude schopné provádět tuto kontrolu automaticky. Díky používanému simulačnímu prostředí Player/Stage jsme schopni sledovat nejen informace ze senzorů robota, ale také informace o chování a pozici robota v bludišti. Současně plánujeme také aplikaci různých metodik pro zajištění odolnosti proti poruchám a jejich ověřování pomocí naší platformy. Plánujeme využít klasické ztrojení (TMR) spolu s naším řadičem rekonfigurace, on-line hlídací obvody a podobně.

PODĚKOVÁNÍ

Tato práce byla podpořena Ministerstvem školství mládeže a tělovýchovy z Národního programu udržitelnosti II (NPU II) v rámci projektu IT4Innovations excellence in science - LQ1602, projektem ARTEMIS JU č. 641439 (ALMARVI) a projektem VUT FIT-S-14-2297.

REFERENCE

[1] M. Ceschia, M. Violante, M. Reorda, A. Paccagnella, P. Bernardi, M. Rebaudengo, D. Bortolato, M. Bellato, P. Zambolin, and A. Candelori, "Identification and Classification of Single-event Upsets in the Configuration Memory of SRAM-based FPGAs," vol. 50, no. 6, 2003, pp. 2088–2094.

[2] I. Koren and C. M. Krishna, *Fault-Tolerant Systems*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007.

[3] F. Siegle, T. Vladimirova, J. Ilstad, and O. Emam, "Mitigation of Radiation Effects in SRAM-Based FPGAs for Space Applications," *ACM Comput. Surv.*, vol. 47, no. 2, pp. 37:1–37:34, Jan. 2015.

[4] C. Bernardeschi, L. Cassano, A. Domenici, and L. Sterpone, "Accurate Simulation of SEUs in the Configuration Memory of SRAM-based FPGAs," in *Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2012 IEEE International Symposium on*. IEEE, 2012, pp. 115–120.

[5] M. Alderighi, F. Casini, S. d'Angelo, M. Mancini, S. Pastore, and G. R. Sechi, "Evaluation of Single Event Upset Mitigation Schemes for SRAM-based FPGAs Using the FLIPPER Fault Injection Platform," in *Defect and Fault-Tolerance in VLSI Systems, 2007. DFT'07. 22nd IEEE International Symposium on*. IEEE, 2007, pp. 105–113.

[6] M. Straka, J. Kastil, and Z. Kotasek, "SEU Simulation Framework for Xilinx FPGA: First Step Towards Testing Fault Tolerant Systems," in *14th EUROMICRO Conference on Digital System Design*. IEEE Computer Society, 2011, pp. 223–230.

[7] B. Gerkey, R. T. Vaughan, and A. Howard, "The Player/Stage Project: Tools for Multi-robot and Distributed Sensor Systems," in *Proceedings of the 11th international conference on advanced robotics*, vol. 1, 2003, pp. 317–323.

[8] S. Krajcir, "Functional Verification of Robotic System Using UVM," Tech. Rep., 2015. [Online]. Available: http://www/study/DP/DP.php?id=15154

Kopulové EDA algoritmy

Martin Hyrš
3. ročník, prezenční forma
Školitel: Josef Schwarz

Fakulta informačních technologií, Vysoké učení technické v Brně
Božetěchova 1/2, 612 66 Brno
ihyrs@fit.vutbr.cz

***Abstrakt*—EDA algoritmy (Estimation of Distribution Algorithm) patří mezi stochastické optimalizační techniky, založené na vytváření a vzorkování pravděpodobnostní modelu. V tomto článku prezentuji CEDA (Copula-EDA) algoritmus, ve kterém je použita kopule k vytvoření modelu a následné migraci parametrů pravděpodobnostního modelu. Experimentální výsledky potvrzují, že tento koncept vede ke zlepšení konvergence (ve srovnání se sekvenční variantou navrženého algoritmu nebo s výsledky CEDA algoritmů jiných autorů).**

***Klíčová slova*—EDA, kopule, migrace modelů.**

I. Úvod

EDA algoritmy (Estimation of Distribution Algorithms) patří do třídy evolučních optimalizačních metod, které prozkoumávají stavový prostor pomocí odhadu/aproximace a vzorkování explicitního pravděpodobnostního modelu nad množinou řešení.

Běžné EDA algoritmy používají grafický pravděpodobnostní model, např. řetězec, stromy nebo Bayesovskou síť, tyto algoritmy jsou známé pod názvy MIMIC [1], BMDA [2], BOA [3], UMDA [4]. Obsáhlý přehled těchto algoritmů je uveden v [5] nebo [6].

V posledních pěti letech začal být používán nový přístup ke konstrukci pravděpodobnostního modelu – postup založený na teorii kopulí [7], [8], [9]. Kopule jsou speciální funkce rozdělení pravděpodobnosti. Díky jejich vlastnostem je lze použít k modelování korelací ve vícerozměrných problémech – sdružená distribuční funkce je rozdělena na jednorozměrná marginální rozdělení a na korelaci, která je popsána kopulí.

II. Teorie kopulí

Definice. Kopule C je vícerozměrná distribuční funkce, jejíž marginální rozdělení jsou rovnoměrná na intervalu $\langle 0; 1\rangle$.

Teorém (Sklarův teorém [10])**.** *Nechť F je d-rozměrná sdružená distribuční funkce s marginálními rozděleními $F_1, \ldots, F_d$. Potom existuje d-rozměrná kopule C taková, že pro všechna $(x_1, \ldots, x_d) \in \mathbb{R}^d$ platí $F(x_1, \ldots, x_d) = C\left(F_1(x_1), \ldots, F_d(x_d)\right)$.*

Zabýváme se dvěma rodinami kopulí – Archimedovskými a eliptickými:

Archimedovské kopule jsou poměrně populární, protože popisují různé vzory závislostí, a mají relativně jednoduchý funkcionální zápis $C(u_1, \ldots, u_d) = \varphi_\theta\left(\varphi_\theta^{-1}(u_1) + \ldots + \varphi_\theta^{-1}(u_d)\right)$. Jejich definice je založena na generátoru φ, tato funkce má typicky jeden parametr θ, který vyjadřuje míru závislosti. To, že i vícerozměrné Archimedovské kopule mívají jen jeden parametr, je do určité míry nevýhoda, protože jejich schopnosti popsat vícerozměrné závislosti jsou výrazně omezené. Toto omezení lze kompenzovat např. pomocí hierarchických Archimedovských kopulí (HAC).

Eliptické kopule jsou odvozeny od příslušného eliptického rozdělení, příkladem je Gaussova kopule $C(u_1, \ldots, u_d) = \Phi_R\left(\Phi^{-1}(u_1), \ldots, \Phi^{-1}(u_d)\right)$, kde $\Phi_R(x_1, \ldots, x_d)$ je sdružené normální rozdělení s pozitivně-semidefinitní korelační maticí R, Φ je standardní normální rozdělení, Φ^{-1} je jeho kvantilová (inverzní) funkce. Obdobným příklad je t-kopule, odvozená od Studentova t-rozdělení. Na rozdíl od Archimedovských kopulí není počet dimenzí nijak limitován, eliptické kopule se typicky používají jako vícerozměrné kopule, parametrem je korelační matice R.

III. mCEDA: EDA, kopule a migrace

V této kapitole stručně popíšeme algoritmus mCEDA, který je podrobně popsán v našem článku [11]. Jedná se o EDA algoritmus, který používá několik typů kopulí, a který je paralelizován s využitím ostrovního komunikačního modelu.

A. EDA (Estimation of Distribution Algorithm)

EDA algoritmy jsou evoluční optimalizační technika podobná genetickým algoritmům. Potenciální řešení problému je reprezentováno pomocí jedince, množina jedinců pak tvoří populaci. Co nejlepší řešení je vyvíjeno v průběhu několika generací, vhodní jedinci vybraní z jedné generace jsou použiti k vytvoření jedinců pro novou generaci. Na rozdíl od genetických algoritmů (které používají křížení a mutaci) jsou noví jedinci v EDA vytvářeni pomocí náhodného vzorkování z pravděpodobnostního modelu. Tento pravděpodobnostní model je odvozován z vybraných jedinců předešlé generace.

EDA algoritmus tedy tvoří tyto tři základní kroky:

1) Vyber vhodné jedince.
2) Vytvoř pravděpodobnostní model.
3) Navzorkuj nové jedince.

Jednotlivé EDA algoritmy se odliší druhem použitého pravděpodobnostního modelu. V případě CEDA je model tvořen kopulí a marginálními rozděleními.

Algorithm 1 Vzorkování kopulového pravděpodobnostního modelu

1) Získej náhodný vzorek kopule $(u_1, \ldots, u_d) \sim C$, kde $u_i \in \langle 0; 1 \rangle$.
2) Odvoď vektor hledaného jedince $\mathbf{x}$ za použití inverzí marginálních rozdělení $x_i = F_i^{-1}(u_i)$.

Algorithm 2 CEDA s migrací modelů (mCEDA)

1) Vygeneruj počáteční populaci.
2) **FOR** každý ostrov **DO IN PARALLEL**:
3) **WHILE** (není konec):
4) Vyber výhodná řešení
5) Vytvoř pravděpodobnostní model
6) **IF** (podmínka migrace):
7) Pošli model
8) **WHILE** (imigrantský model přijat):
9) Zkombinuj modely
10) Navzorkuj novou populaci ze získaného pravděpodobnostního modelu

B. Kopulový pravděpodobnostní model

Pravděpodobnostní model v kopulovém EDA algoritmu je specifikován ve dvou krocích. Nejdříve jsou odvozeny parametry pro jednorozměrná marginální rozdělení – používáme normální rozdělení $No(\mu, \sigma)$ s parametry μ (střední hodnota) a σ (směrodatná odchylka). Za druhé jsou vypočítány parametry pro zvolenou kopuli (R nebo θ) pomocí korelačního koeficientu Spearmanova ρ_S.

Hlavní přínos tohoto postupu spočívá v tom, že vzorkování nových jedinců (krok 10 v Alg. 2) je realizováno ve dvou jednoduchých krocích, tak jak je popsáno v Alg. 1. Tento postup umožňuje oddělit marginální rozdělení od struktury závislostí mezi dimenzemi.

C. Migrace modelů

V případě ostrovní modelu evoluční algoritmu se celková populace skládá z několik částí – ostrovů. Evoluční proces na každém ostrově běží nezávisle (viz Alg. 2). Pouze když je splněna podmínka migrace, je aktivována komunikace (v našem případě přenos parametrů modelu mezi dvojicí sousedících ostrovů).

Celý proces migrace je rozložen na interakci sousedních ostrovů – příjemce (rezidentní model) a imigrant (příchozí model). Model z imigranta je přenesen k příjemci, kde je zkombinován s původním rezidentním modelem a tak vzniká nový rezidentní model. Migrace a kombinace modelů je v Alg. 2 pokryta řádky 6–9.

Ke kombinaci modelů jsme použili následující vztahy, v souladu s [12], [13]:

- Učení středních hodnot μ_i každého jednorozměrného marginálního rozdělení $F_i(x_i)$

$$\mu_i^{new} = (1-\beta)\mu_i^R + \beta\mu_i^I \tag{1}$$

- Učení směrodatných odchylek σ_i každého jednorozměrného marginálního rozdělení $F_i(x_i)$

$$\sigma_i^{new\,2} = (1-\beta)\left(\left(\mu_i^{new} - \mu_i^R\right)^2 + \left(\sigma_i^R\right)^2\right) + \\ + \beta\left(\left(\mu_i^{new} - \mu_i^I\right)^2 + \left(\sigma_i^I\right)^2\right) \tag{2}$$

- Učení prvků korelační matice R_{ij}

$$R_{ij}^{new} = (1-\beta)R_{ij}^R + \beta R_{ij}^I \tag{3}$$

Koeficient β je určen vztahem

$$\beta = \begin{cases} \frac{fit^R}{fit^R + fit^I} & fit^I \leq fit^R \\ 0.1 & \text{jinak} \end{cases} \tag{4}$$

kde fit^R nebo fit^I představuje střední hodnotu fitness rezidenta respektive imigranta.

IV. Experimenty

Ukázkové experimenty byly provedeny na sadě běžných testovacích úloh (Rastrigin, Ackley, Schwefel, Rosenbrock, Sphere) a na sadě CEC 2013 [14] pro 10 rozměrů. Jsou prezentovány výsledky dvou variant mCEDA, používající Frankovu (Archimedovská rodina) kopuli (mCEDA-F) a Gaussovu (eliptická rodina) kopuli (mCEDA-G), ve srovnání se dvěma variantami sCEDA-* (sekvenční verze, tj. pouze jeden ostrov). Výsledky výše popsaných algoritmů mCEDA-* a sCEDA-* jsou znázorněny v grafech na obrázku 1.

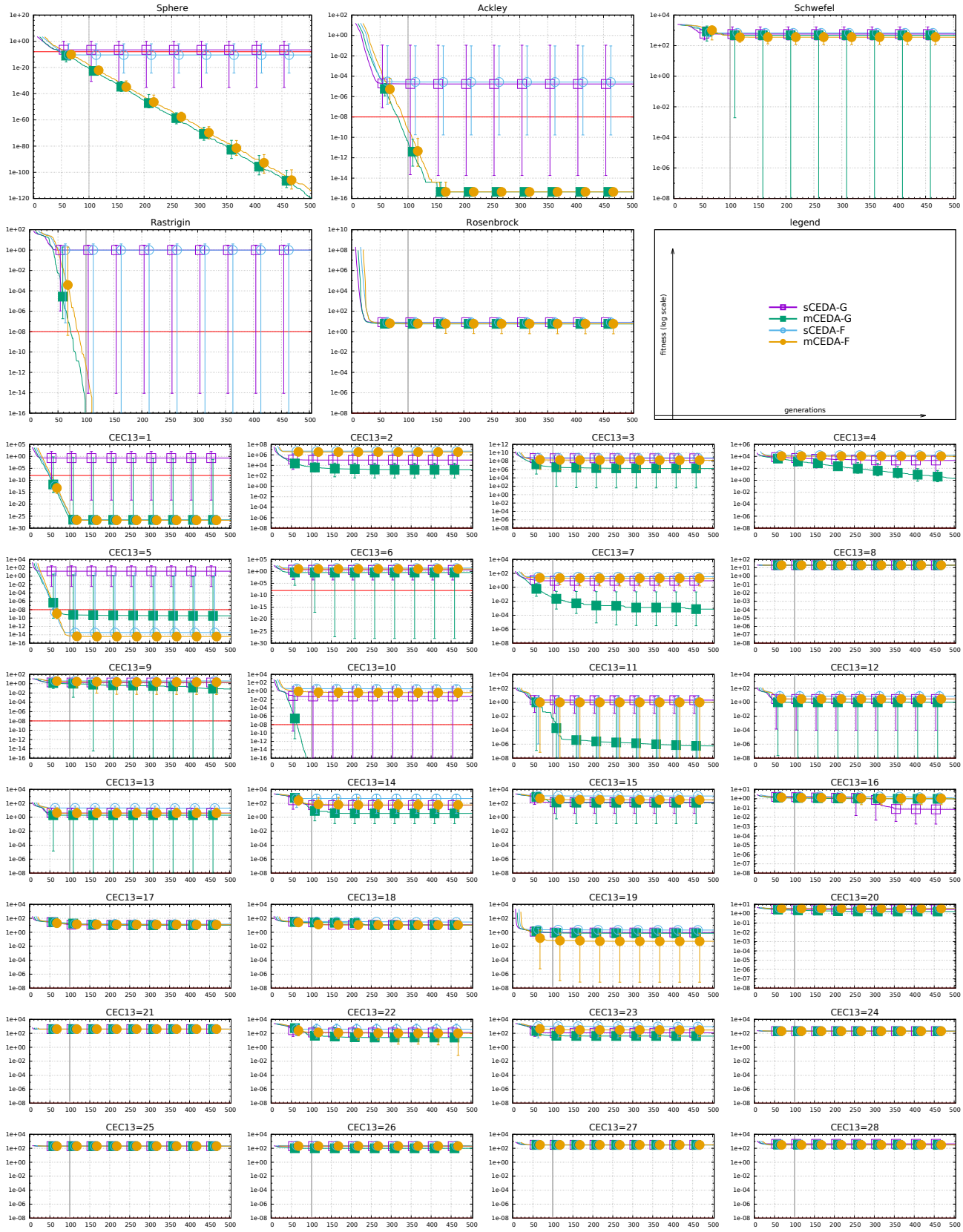
Z grafů je patrné, že verze s migrací modelů (na grafu jako plné body) je téměř vždy lepší (nebo stejně dobrá) jako sekvenční verze (prázdné body); výjimkou je funkce č. 16 ze sady CEC13. Varianta užívající Gaussovu kopuli (čtverečky) je obvykle lepší nebo stejně dobrá jako varianta s Frankovou kopulí (kolečka). Celkově nejlepší testovanou variantou je tedy mCEDA-G (kopulová EDA s Gaussovou kopulí a migrací modelů (plné zelené čtverečky). V deseti případech ze sady CEC13 je mCEDA-G zřetelně lepší než ostatní varianty. Ve dvou případech je lepší mCEDA-F, na zbylých úlohách není patrný žádný významný rozdíl.

V Tabulkách I a II je prezentováno srovnání (střední hodnoty fitness) navrženého algoritmu mCEDA-* s dalšími publikovanými algoritmy, které nějakým způsobem používají kopule. Srovnání jsme provedli vždy pro stejný počet vyhodnocení fitness. Ve většině případů dosahuje mCEDA o několik řádů lepších výsledků.

V našem algoritmu jsme použili toto nastavení experimentů: 4 ostrovy, 250 jedinců na každém ostrově, selekce 20 %, migrace každých 10 generací, obousměrná kruhová topologie, 20 nezávislých běhů.

V. Cíl dizertace

Teorie kopulí je relativně nový koncept používaný ve statistice, ale v oblasti evolučních algoritmů se teprve začíná více rozvíjet. Moje práce zkoumá nové možnosti EDA algoritmů na bázi kopulí. Dosavadní výsledky naznačují, že CEDA



Obrázek 1. Výsledky experimentů – medián a kvartily hodnoty fitness algoritmů s Gaussovou a Frankovou kopulí; sekvenční verze a verze s migrací modelů.

Tabulka I
mCEDA vs. Copula Bayesian Network (CBN) [15] – 100000 vyhodnocení fitness.

	Rastr.	Ack.	Schw. 1.2	Rosen.
CBN	2,39e+00	3,71e-02	2,23e+01	1,05e+01
mCEDA-F	1,99e-01	1,28e-11	3,41e-21	5,90e+00
mCEDA-G	1,99e-01	1,14e-11	1,08e-20	5,66e+00

Tabulka II
mCEDA vs. Clayton (Cl), Gumbel (Gu), Sn-EDA (Sn) [16] – 300000 vyhodnocení fitness.

	Sphere	Rastrigin's	Rosenbrock's
Cl	1,45e-07	7,00e-08	8,36e+00
Gu	3,59e-09	5,49e-09	6,62e+00
Sn	1,22e-09	9,52e-09	6,54e+00
mCEDA-F	4,05e-67	1,99e-01	5,64e+00
mCEDA-G	1,58e-67	1,99e-01	5,51e+00

umožňuje efektivní vytváření pravděpodobnostních EDA modelů, zejména v případě komplexních problémů s netriviálními vazbami mezi proměnnými.

Dalším aspektem, kterým se zabývám, je paralelizace algoritmu pomocí ostrovního modelu – tzn. použití několika samostatných populací (tzv. ostrovů), mezi kterými dochází k občasnému přenosu informací. Zejména se zaměřuji na migraci pravděpodobnostních modelů jednotlivých subpopulací mezi ostrovy.

Cílem dizertace je ukázat, že EDA algoritmy mohou významněji profitovat z použití teorie kopulí a migrace pravděpodobnostních modelů, tj. dokázat, že pro určitý typ úloh lze použitím vhodné kopule výrazně zvýšit efektivitu EDA algoritmů vzhledem ke standardní variantě EDA.

VI. Závěr

Představili jsme použití kopulí při návrhu pravděpodobnostního modelu v EDA algoritmu a užití paralelního ostrovního modelu algoritmu s použitím migrace modelů.

Ze vzájemného srovnání jedno-ostrovní verze (sCEDA) s více-ostrovní (mCEDA) je patrný velký přínos ostrovního modelu. Toto srovnání bylo provedeno jednak na běžných testovacích úlohách, jednak na sadě úloh CEC 2013.

Výkonnost našeho mCEDA algoritmu byla porovnána s jinými publikovanými kopulovými EDA. Naše mCEDA je ve většině případů výrazně lepší.

Poděkování

This work was supported by Brno University of Technology under number FIT-S-14-2297.

Reference

[1] J. S. De Bonet, C. L. Isbell, and P. A. Viola, "MIMIC: Finding optima by estimating probability densities," in *Advances in Neural Information Processing Systems*, vol. 9. The MIT Press, Cambridge, 1997, pp. 424–430.

[2] M. Pelikan and H. Mühlenbein, "The bivariate marginal distribution algorithm," in *Advances in Soft Computing*. Springer London, 1999, pp. 521–535.

[3] M. Pelikan, D. Goldberg, and E. Cantú-Paz, "BOA: The bayesian optimization algorithm," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99)*, vol. I, 1999, pp. 525–532 also IlliGAL Report no. 99003.

[4] M. Pelikan and H. Mühlenbein, "Marginal distributions in evolutionary algorithms," in *Proceedings of the International Conference on Genetic Algorithms Mendel 98*, 1999, pp. 90–95.

[5] P. Larrañaga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Norwell, MA, USA: Kluwer Academic Publishers, 2001.

[6] M. Hauschild and M. Pelikan, "An introduction and survey of estimation of distribution algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 3, pp. 111 – 128, 2011.

[7] J. Mai and M. Scherer, *Simulating Copulas: Stochastic Models, Sampling Algorithms, and Applications*, ser. Series in quantitative finance. Imperial College Press, 2012, vol. 4.

[8] R. B. Nelsen, *An Introduction to Copulas*, ser. Springer Series in Statistics. Springer New York, 2006.

[9] U. Cherubini, E. Luciano, and W. Vecchiato, *Copula Methods in Finance*. Hoboken, NJ: John Wiley & Sons, 2004.

[10] A. Sklar, "Fonctions de répartition à n dimensions et leurs marges," *Publications de l'Institut de Statistique de l'Université de Paris*, vol. 8, pp. 229–231, 1959.

[11] M. Hyrš and J. Schwarz, "Elliptical and archimedean copulas in estimation of distribution algorithm with model migration," in *Proceedings of the 7th International Joint Conference on Computational Intelligence (IJCCI 2015)*. SciTePress - Science and Technology Publications, 2015, pp. 212–219.

[12] S. Frühwirth-Schnatter, *Finite Mixture and Markov Switching Models*. New York: Springer, 2006.

[13] J. Schwarz and J. Jaroš, "Parallel bivariate marginal distribution algorithm with probability model migration," in *Linkage in Evolutionary Computation*, ser. Studies in Computational Intelligence. Springer Berlin Heidelberg, 2008, vol. 157, pp. 3–23.

[14] J. Liang, B. Qu, P. Suganthan, and A. G. Hernández-Díaz, "Problem definitions and evaluation criteria for the cec 2013 special session on real-parameter optimization," *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report*, vol. 201212, 2013.

[15] M. Méndez and R. Landa, "An EDA based on bayesian networks constructed with archimedean copulas," in *2012 Fourth World Congress on Nature and Biologically Inspired Computing (NaBIC)*, 2012, pp. 188–193.

[16] B. Jia, L. Wang, and Z. Cui, "Copula for estimation of distribution algorithm based on goodness-of-fit test," in *Journal of Theoretical and Applied Information Technology*, no. 3, 2013, pp. 1128–1132.

Počítačové architektury & diagnostika PAD 2016
Česko-slovenský seminář pro studenty doktorského studia

Sborník příspěvků

Název:	Počítačové architektury & diagnostika PAD 2016
Editoři:	Jiří Jaroš a Richard Růžička
Autoři:	Kolektiv autorů příspěvků
Vydavatel:	Vysoké učení technické v Brně Fakulta informačních technologií
Pořadí vydání:	První
Rok vydání:	2016

ISBN 978-80-214-5376-0