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

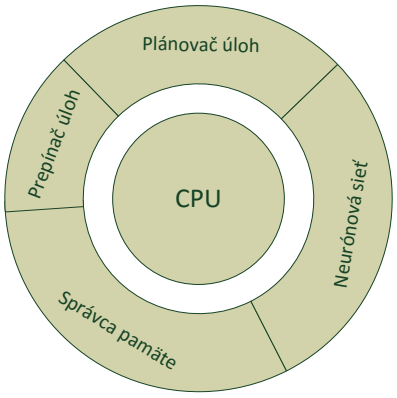
dosiahnutá konštantnou časovou zložitosťou operácií, ktoré má platforma poskytovať. Ďalšími parametrami optimalizácie sú štandardné parametre, na ktoré sa prihliada pri vývoji číslicových obvodov. Medzi tie patrí napríklad pracovná frekvencia systému, počet hodinových cyklov potrebný na vykonanie operácie, 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:

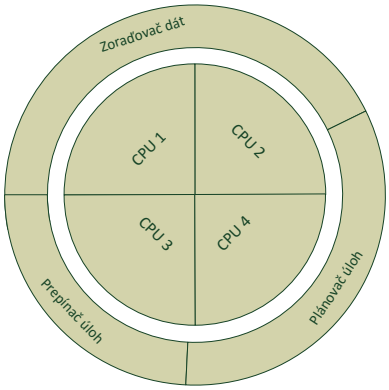
- CPU
- Prepínač úloh
- Plánovač úloh
- Správca dynamickej pamäte
- Zoraďovač aplikačných dát
- Neurónová sieť

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.