Evolutionary Design of Digital Circuits: Where Are Current Limits?

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

Abstract

The objective of this paper is to classify the approaches proposed to the evolutionary digital circuit design in the recent years and to identify the levels of complexity and innovation that can be obtained by means of these approaches. In particular, gate-level evolution, circuit evolution in PLAs, functional-level evolution, incremental evolution, evolution utilizing developmental schemes and some application-specific schemes are analyzed. It is shown that we are able to effectively explore the search spaces not much larger than 2^{1000} points and that the innovative solutions can be produced independently of the utilized method.

1 Introduction

Evolutionary algorithms (EAs) have been utilized to design digital circuits in the recent decade. Papers [3, 31, 27] and others survey this field from various points of view.

In the area of evolutionary circuit design, researchers are primarily looking for *innovative* solutions. The word "innovative" has various meanings in this context: In some cases it means that EA is capable of evolving large/complex circuits. Therefore, it is natural to ask: What is the largest multiplier that we are able to evolve using computing resources available nowadays? Is any innovation visible in the evolved circuits in comparison with the conventional designs? Typically, the search space we have to deal with is very large, rugged and so difficult to search. Researchers have proposed new problem representations, operators and fitness functions to make the evolution more efficient.

On the other hand, researchers are also interested in novel applications. For example, Kasai et al have proposed an adaptive waveform control circuit for a data transceiver communication (IEEE 1394/USB) [12] allowing devices to communicate faster than the current standard or with a longer USB cable. There is no doubt that this approach is very useful. However, the problem is relatively simple from the point of view of EA as only 100bit-long chromosomes are utilized. In this category, the main contribution lies in a novel approach proposed to solving a given problem.

This paper deals with the evolutionary digital circuit design from the perspective of complexity and innovation that can be obtained using an evolutionary approach. The objective of this paper is to classify the approaches proposed to the evolutionary digital circuit design in the recent years and to identify the levels of complexity and innovation that can be obtained by means of these approaches. The following methods will be analyzed: gate-level evolution, circuit evolution in PLAs, functional-level evolution, incremental evolution, evolution utilizing developmental schemes and some application-specific schemes.

2 Evolutionary Digital Circuit Design

2.1 The Method

The evolutionary algorithm is utilized to search for a suitable configuration of a reconfigurable device in order to achieve the behavior required by a specification [7]. A chromosome represents either the configuration bitstream directly or a prescription determining how to create the configuration bitstream. Candidate circuits are evaluated in the following way: First, a configuration bitstream is extracted from a chromosome. Then, the bitstream is uploaded into a (simulator of) reconfigurable device and the circuit created is evaluated using the fitness function. In case of digital circuits, some training vectors are applied at the primary inputs, corresponding output responses are collected and compared against the desired vectors. The objectives are various, typically to minimize the difference between the output signal values and target signal values, delay, area on the chip etc.

Extrinsic evolution means that candidate circuits are evaluated using a circuit simulator, i.e. in software. *Intrinsic evolution* means that every candidate circuit in every population is evaluated in a physical reconfigurable circuit. In the second approach, the evolution can also exploit

Proceedings of the First NASA/ESA Conference on Adaptive Hardware and Systems (AHS'06)
0-7695-2614-4/06 $20.00 © 2006 IEEE

physical properties of a chip and other environmental characteristics (such as temperature, electromagnetic field etc.). The evolved circuits operate differently in different environments [25].

2.2 Evolutionary Circuit Design vs Evolvable Hardware

It is important to distinguish between two approaches: evolutionary circuit design and evolvable hardware. In case of the *evolutionary circuit design*, the objective is to evolve (i.e. to design) a single circuit. The aim is typically to design novel implementations that are better (in terms of area, speed, power consumption) than conventional designs and/or to design circuits with additional features such as fault-tolerance, testability, polymorphic behavior etc. that are difficult to design by conventional techniques.

In order to "measure" the level of innovation in evolved designs, J. Koza has proposed criteria to classify evolved designs. According to his classification, an automatically created result is "human-competitive" if it satisfies at least one of the eight criteria defined in [10]: Currently, there are 60 human-competitive results. Most results were evaluated as human-competitive in the area of analog circuit design (25 results), quantum circuit design (8 results) and digital circuit design (5 results).

In case of *evolvable hardware*, the evolutionary algorithm is responsible for continual adaptation. Evolvable hardware was applied to high-performance and adaptive systems in which the problem specification is unknown beforehand and can vary in time [6, 23, 20, 8]. Its main objective is the development of a new generation of hardware, self-configurable and evolvable, environment-aware, which can adaptively reconfigure to achieve optimal signal processing, survive and recover from faults and degradation, and improve its performance over lifetime of operation [23].

2.3 Issues in the Evolutionary Circuit Design

2.3.1 Bias in the Design Method

The resulting circuits are built according to the genetic information coming from EA and by means of the bias included into the design method by designer. The complexity of evolved circuits cannot be evaluated without considering this bias. We can illustrate this issue on a 4bit adder design problem. In case that this circuit is evolved at the gate level, the chromosome defines which gates are used and how they are interconnected. The bias is seen in terms of the initial set of gates that can be utilized and in the options in connectivity of these gates (e.g. only two-input gates allowed, no feedback allowed, etc.). Typically, the chromosome consists of a few hundred of bits in this case. On the other hand, when the evolutionary design is carried out at the functional level, the 4bit adder could be composed of 1bit full adders. In comparison with the gate-level, the chromosome is much shorter (les than 100 bits) because the resulting circuits consist of fewer components and interconnections. It is much easier to evolve the 4bit adder at functional level than at gate level because the search space is much smaller. While the bias allowed the designer to reduce the search space and to make evolution easier its disadvantage could be that the evolved solutions do not exhibit any innovation in their structure.

2.3.2 Chromosome Size vs Complexity of Circuits

It is important to explore the relation between the size of chromosome and the complexity of evolved circuits. Usually, a complex solution (circuit) is represented using a long chromosome. Long chromosomes imply large search spaces that are typically difficult to search. Computing resources that are currently available determine the size of the search space that can be explored. Consequently, a limit in the complexity of evolved circuits exits which designers are not able to overcome using available computing resources. This is known as the problem of *scalability of representation*. If a non-trivial genotype-phenotype mapping is introduced, the search space can sometimes be compressed and so the limit in the complexity of evolved circuits can be overcome.

2.3.3 Fitness Calculation as a Bottleneck

In case of combinational circuits evolution the evaluation time of a candidate circuit grows exponentially with the increasing number of inputs (assuming that all possible input combinations are tested in the fitness function). Hence, the evaluation time becomes the main bottleneck of the evolutionary approach even if the circuit consists of a few components. This problem is known as the problem of *scalability of evaluation*. In order to reduce the time of evaluation, only a subset of all possible input vectors can be utilized. In these applications (for example, in signal processing [26]), the evolutionary design is performed using a training set and the solution is validated using a test set. In some other cases it is sufficient to evaluate only some structural properties of candidate circuits which can be done with linear or quadratic time complexity (with respect to the number of circuits elements) even for complex circuits [18]. Then, relatively complex circuits can be evolved.

2.3.4 Level of Innovation

It is impossible to measure the level of innovation in an evolved circuit. Only a qualified expert in the particular

field is able to recognize the result of evolution as an innovative solution. The Hummies competition organized by J. Koza shows a way to do that in the area of evolutionary design. As this paper will clearly illustrate, it is reasonable to believe that the level of innovation does not depend on the approach utilized to evolve the circuit and on the complexity of the evolved circuits.

3 A Brief Survey of Approaches for the Evolutionary Circuit Design

As it is impossible to include all existing techniques (and applications), we have chosen those that have attracted most attention in the recent years.

3.1 Gate-Level Evolution

Thompson has pioneered the intrinsic gate-level evolution in FPGAs [25]. For extrinsic evolution, Miller and Thomson introduced Cartesian Genetic Programming (CGP) that has recently been applied by several researchers especially for the evolutionary design of combinational circuits [16]. In CGP, the reconfigurable circuit is modeled as an array of u (columns) $\times$ v (rows) of programmable elements (PEs) whose functionality is task-dependent. EA is used to find interconnections of PEs and functions performed by PEs.

A number of combinational circuits were evolved at the gate level, including small multipliers, parity circuits, majority (i.e. median) circuits, sorting networks and adders [16, 30, 19]. Novel implementations, unknown in the area of conventional electronics, were proposed for small multipliers (the best conventional solutions were improved in 20% in terms of area [30]) and edge-triggered D-latches equipped with full on-line built-in self test [2].

3.1.1 Evolution in PLAs

When a PLA is utilized then the chromosome determines status of switches in AND and OR planes defining thus a logic function in the disjunctive normal form. An LSI evolvable hardware chip was developed for real-world applications of evolvable hardware [6]. The chip consists of GA hardware, two reconfigurable PLA circuits, chromosome memory, training data memory and a 16-bit processor core. The maximum input width is 28 bits and the maximum output width is 8 bits. The maximum length of chromosome is 2048 bits. Among others, the chip was utilized to implement an EMG signal classificatory unit for a prosthetic hand controller and a robot controller unit. In these applications, the evolution is performed using a training set and the evolved solutions are verified using a test set.

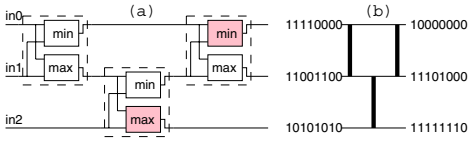

Figure 1. A 3-input sorting network encoded as (0,1)(1,2)(0,1) and its alternative symbol

3.2 Special Representations

Specialized representations are utilized to include more domain knowledge to EA and so to evolve more complex circuits than by using, for example, CGP.

Sorting networks (SN) and median networks can be considered as N-input digital circuits composed of AND and OR gates. However, a very efficient representation based on *compare&swap* components was developed to improve the efficiency of the search algorithm. A *compare&swap* of two elements (a, b) compares and exchanges a and b so that $a \leq b$ is obtained after the operation (see Fig. 1). A sorting network is defined as a sequence of compare&swap operations.

A specialized architecture was developed to evolve sorting networks (with $N \leq 28$) in Xilinx FPGA Virtex II [13]. The interconnection of *compare&swap* components were determined by EA implemented on the same chip. Sorting networks were optimized neither for the speed nor size. For example, the evaluation of a candidate network requires 1.3 ms for $N = 16$ and 5.4 s for $N = 28$ (at 50 MHz). The evaluation time of a single candidate sorting network for $N = 28$ was compared against a highly optimized SW implementation running in Xeon 3 GHz. The FPGA evaluation running at 100 MHz is $40\times$ faster than the software approach. In this application, outputs for all possible input combinations are evaluated in the fitness function.

Also median circuits were represented as sequences of *compare&swap* components. For example, the median network shown in Fig. 1 can be encoded as (0, 1)(1, 2)(0, 1). The results reported in [19] show that median circuits were evolved up to $N = 25$; however, they are area-optimal only for $N = 3 - 11$.

3.3 Functional Level Evolution

The idea of functional level evolution was introduced in [14]. In contrast to the gate level evolution, circuits are composed of high-level functional blocks such as adders and multipliers. An important property of this approach is that the size of chromosome remains similar to CGP while the complexity of circuits can grow arbitrarily. Functional-level evolution was applied to many real-world problems,

for example, to the design of multiplier-less filters [8], image filters [15], multipliers [1] etc. In order to illustrate the method, this section describes evolution of image filters and benchmark circuits.

3.3.1 Evolutionary Design of Image Filters

A special filter is evolved for a given type of noise. Every image operator is considered as a digital circuit of nine 8bit inputs and a single 8bit output, which processes grayscaled (8bits/pixel) images. As Fig. 2 shows, every pixel value of the filtered image is calculated using a corresponding pixel and its eight neighbors in the processed image. The design objective is to minimize the mean difference per pixel (mdpp) between the filtered image and the original image. The evolvable system was simulated [20] and then completely implemented in an FPGA [15].

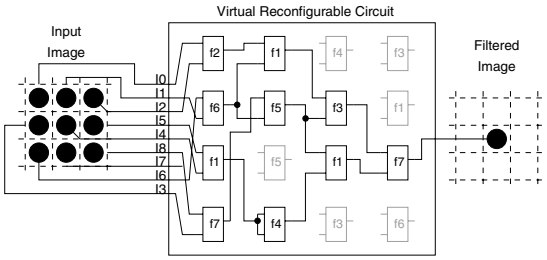

Figure 2. Functional-level evolution of 3×3 image operators

From [15] it can be derived that 30k generations (i.e. 20 seconds in FPGA operating at 100 MHz) are needed in average to find a filter. The design time is very reasonable if the proposed system should operate "instead" of a designer in the image filter design task. Note that the speedup is 50 against the software approach (Pentium III/800MHz) if the FPGA operates at 100 MHz. The system can be used either to adapt image filters in situ or to assist the designer in the design process of a particular filter. In the second case, the resulting filter is also available at the level of synthesizable VHDL code. When implemented in an FPGA, the filters consist of 1000–5000 equivalent gates [20].

3.3.2 Evolution of Benchmark Circuits

If we were able to completely evaluate a candidate solution in a linear or quadratic time (with respect to the number of circuit inputs/components) the evolutionary design process will be more effective and scalable. Fortunately, some methods exist to predict testability of a digital circuit in a quadratic time [18]. Hence, EA was utilized to design benchmark circuits with the required testability properties [18].

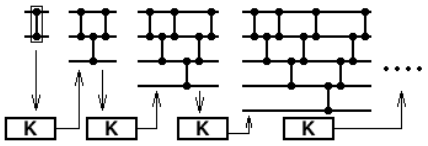

Figure 3. Designing larger sorting networks from smaller sorting networks by means of a constructor K

The objective was to produce register transfer level benchmark circuits automatically. The user is supposed to specify the number of primary inputs and outputs of the circuit, the number and type of components, the requirements on testability (average controllability and observability) and parameters of the evolutionary algorithm. The fitness function analyzes three features of candidate circuits: structure, component interconnections and (3) circuit testability.

There are two interesting features of this approach: (1) Relatively complex circuits were evolved with the required testability properties. According to our knowledge, no similar approach exists to accomplish this task. (2) The evolved circuits are the largest circuits evolved so far. The website of the FITTest_BENCH06 Benchmarks project [17] shows that the most complicated benchmark circuit (s20) consists of 310,610 gates (700 components, 180 primary inputs).

3.4 Development in EAs

Various approaches have been proposed to introduce the development to EAs. In order to implement nontrivial genotype-phenotype mappings and so to allow the evolution of large (or fault-tolerant, polymorphic etc.) circuits they have utilized L-systems, cellular automata [5], interactions of genes and proteins [4] or specialized instruction sets [9].

Evolution of arbitrarily large sorting networks will be presented to illustrate the idea of development [21]. A genetic algorithm is used to design a program – constructor (consisting of application-specific instructions) – that is able to create a larger sorting network from a smaller one (the smallest one is called the embryo). Then the constructor is applied on its results in order to create a larger sorting network and so on (see Fig. 3).

EA have produced sorting networks with better implementation cost (the number of comparators) and delay than the conventional method for even-input as well as odd-input sorting networks [21]. The main feature of this developmental system is that a lot of problem-domain knowledge (such as the definition and use of copy and modify instructions) has been presented in its inductive bias. It seems that the design of an efficient developmental system is as dif-

ficult as the design of an efficient genetic algorithm for a given problem. This example represents the rare case in which a new scalable principle was discovered by an evolutionary algorithm.

3.5 Transistor Level

Small digital circuits were evolved at the transistor level using elementary components such as transistors, capacitors and resistors [22]. For purposes of our comparison, a two-input NAND gate evolved using two cells of JPL's FPTA and the Set-Reset a D-Latch circuits evolved using 4–5 cells of FPTA are considered.

3.6 Incremental Evolution

In order to evolve more complex circuits, Torresen has introduced a *divide-and-conquer* approach for the evolution of evolvable hardware systems (also named *increased complexity evolution*) [26]. The evolution is first applied individually to a set of basic units. The evolved units are building blocks for the evolution of a complex system.

The problem is how to define the fitness functions for the subcircuits. The approach can be twofold in the case of combinational circuits. Either training vectors (i.e. each separate output is evolved separately) or the training set (i.e. some subsets are identified and corresponding circuits evolved separately) are partitioned and corresponding circuits are evolved. The objective is usually to evolve larger systems rather than to optimize the amount of resources.

A specialized architecture has been proposed by Torresen to evolve prosthetic hand controller [26] and sign number recognizer [28]. The idea of incremental evolution has been applied to find configurations of subsystems of that architecture. In both cases very good results have been reported.

Kalganova applied the incremental evolution in two directions [11]. The principle is to semi-automatically divide a complex task into simpler subtasks in order to evolve each of these subtasks and then to incrementally merge the evolved subsystems, reassembling a new evolved complex system. The approach has been successfully evaluated using circuits from MCNC library (for example, the circuits of 16 inputs and 1 output, 10 inputs and 16 outputs etc.) and multiplier circuits of 6bit operands [24].

4 Analysis of Evolved Circuits

4.1 Size of Chromosome vs Complexity of Evolved Circuits

For purposes of this paper, we will measure the *complexity* of an evolved circuit as the number of gates utilized in the evolved circuit. This is not a perfect measure because the complexity of evolved circuits was not optimized in some applications (for example, for sorting networks evolved in FPGA). Furthermore, the gates were not used as building blocks in some experiments (e.g. for the evolution at the transistor and functional levels). Therefore, in order to express the complexity of circuits evolved at these levels some conversions have been performed. Nevertheless, this measure is accurate enough to distinguish between complexities produced by different classes of circuits that were evolved using different approaches.

Domain knowledge (bias in the method) is the second parameter considered to characterize the quality of presented methods. Most domain knowledge is included into the problem representation in which a designer defines the elementary blocks the circuits are built from and the possibilities for interconnecting these blocks. Note that the size of chromosome is not usually directly related to the amount of domain knowledge included. A short chromosome means that we assume a lot about the target system (i.e. we know how it should "look like"). A long chromosome means that we do not assume a lot about the target system and more freedom is given to the evolution to build the target system. Again, the proposed measure is not perfect; however, it is sufficient for our purposes.

Figure 4 shows the relation between the size of chromosome and the complexity of circuits presented in the previous section. The curves in Fig. 4 can easily be extrapolated toward circuits of lower complexity (and shorter chromosomes). In particular, the following classes are considered.

The **CGP** class contains the circuits evolved using CGP that have more than one output (2-4bit multipliers, 7-input sorting network etc.) [30].

CGP-median class consists of median circuits that have from 3 to 11 inputs and a single output [19].

Median circuits evolved using compare&swap encoding (**C&S-median**) [19] have from 3 to 25 inputs. A single compare&swap component is counted as two gates.

In the **image filters** class, the image filters evolved in the FPGA are included. Their implementation cost was determined as a cost of the filter which is extracted from the FPGA and synthesized as an independent circuit. The average number of gates is around 2000 [20].

The class of **benchmark** circuits consists of circuits according to [18].

Sorting networks constructed using development are included to the **development – SN** class. Using this approach, arbitrarily large sorting networks can be constructed by means of the chromosome of a constant size [21].

Sorting networks evolved in FPGA are included to the **sort_nets in FPGA** class [13]. Although some PEs operate solely as wires (i.e. they do not influence the complexity of the circuit), we are counting every programmable element

utilized as the equivalent to two gates.

For purposes of this comparison, we consider that the two-input gate evolved in the **FPTA** is equivalent to a single gate, the RS circuit is equivalent to two gates and the D-latch is equivalent to 5 gates [22].

The **PLA** class contains circuits evolved in a PLA. The complexity of these circuits was estimated on the base of PLA size utilized in applications [7, 6].

4.2 A Survey of Design Limits

The mentioned techniques have a limitation in the size of the search space they are able to effectively explore. The limitation is somewhere around 1000 bits in the chromosome. The main reason for this limit is that (1) the corresponding search spaces are extremely rugged and so difficult to search and (2) the computational power available to researches is limited.

The *gate-level* evolution can produce circuits of complexity up to approx. 100 gates. This class is characterized by the fact that all possible input/output combinations are considered in the fitness calculation process. As the complexity of the circuits is low and simultaneously the chromosome is long enough, the chromosome may encode many candidate variants. It means that novel solutions can be discovered easily.

By using an *application-specific encoding* (as in case of C&S-medians or sorting networks in FPGA), more complex circuits can be obtained in comparison to the simple CGP. Similarly to CGP, all possible input/output combinations are evaluated in the fitness function. Hence a very time-consuming fitness evaluation represents the main obstacle in growing complexity.

The *functional-level* approaches have produced complex and innovative solutions. The complexity of circuits is in fact unlimited and depends on a complexity of components used as building blocks. However, because the complexity of evolved circuits is relative high, it is impossible to test all possible input/output combinations and, therefore, only a subset of these combinations can be evaluated in the fitness function (alternatively, some structural properties of candidate circuits can be evaluated). That immediately implies that (1) usually only a suboptimal solution is obtained and (2) the application class is restricted. The level of novelty obtained strongly depends on the functional blocks utilized, i.e. on the amount of domain knowledge introduced.

Developmental approaches are able to produce arbitrarily complex solutions using relatively short chromosomes. In case that area on a chip should be minimized, the problem is that the obtained solutions exhibit regularity, which causes redundancy.

As the *transistor-level* evolution produces only small digital circuits, it leaves a space opened for unconventional implementations of these elementary digital circuits. Similarly, molecular level evolution of digital circuits have become very promising in the recent years [29].

Incremental evolution has not been considered in Fig. 4. because it is not comparable with other approaches in the proposed way.

We can summarize that the complexity of evolved circuits only partially depends on the size of chromosome. It mainly depends on the size of objects encoded in the chromosome or manipulated by instructions encoded in the chromosome (in case of development). That is the reason why one can evolve relatively complex circuits although it is possible to effectively search only in the search space of size approx 2^{1000}.

4.3 Comparison of Computational Effort

Let us consider those circuits that have similar size of the configuration bitstream. In other words, we are interested in circuits designed by EAs operating in the search space of similar size. Table 1 summarizes properties of circuits described by chromosome of the size between 512–868 bits. The columns have the following meaning: *in* is the number of inputs, *out* is the number of outputs, *chrom-size* is the size of chromosome, *gates* is the number of gates the circuit is composed of, *test-vect* is the number of test vectors used in the fitness function to evaluate the circuit[1], *pop-size* is the population size and *gnrs* is the average number of generations to evolve the circuit.

Figure 5 shows some of these values in a graphical form. The circuits are sorted according their complexity. Let us assume that the evaluation of a circuit requires a unit time for a single training vector. We can see that the evaluation of a 21-input median circuit is the most time-consuming. It is because the circuit has many inputs and all possible combinations are considered. Recall that for image filters and benchmark circuits not all possible input combinations are considered. The *pxg* is the product of population size and the number of generations (pxg = popsize x gnrs). It informs us how many candidate circuits must be evaluated in order to find the required solution; in other words how it is *difficult* for the evolutionary algorithm to find a solution. It can be seen that this parameter is maximal for CGP although the complexity of these gate-level circuits is not very high. We have to mention that in CGP we have optimized the size of the evolved circuits, i.e. a lot of computational time was spent to improve an already discovered solution. Surprisingly, *pxg* decreases with the complexity of evolved circuits. If we multiply *pxg* by the number of test vectors,

[1] Benchmark circuits are not evaluated using test vectors; instead, the structural analysis of the circuit is performed. As the analysis has a quadratic complexity, we can estimate the number of steps required to evaluate a benchmark circuits as K^2 where K denotes the number of components in the circuit.

Table 1. Properties of circuits whose chromosomes have a similar length

circuit	in	out	chrom-size	gates	test-vect	pop-size	gnrs
9-input median (CGP)	9	1	755	36	2^9	128	11k
4x3-bit multiplier	7	7	700	46	2^7	5	10M
21-input median (C&S)	21	1	800	140	2^{21}	200	270
sorting network in FPGA	16	16	512	256	2^{16}	4	20k
image filter	72	8	616	2000	254^2	4	30k
benchmark circuit	40	40	525	7003	50^2	25	200

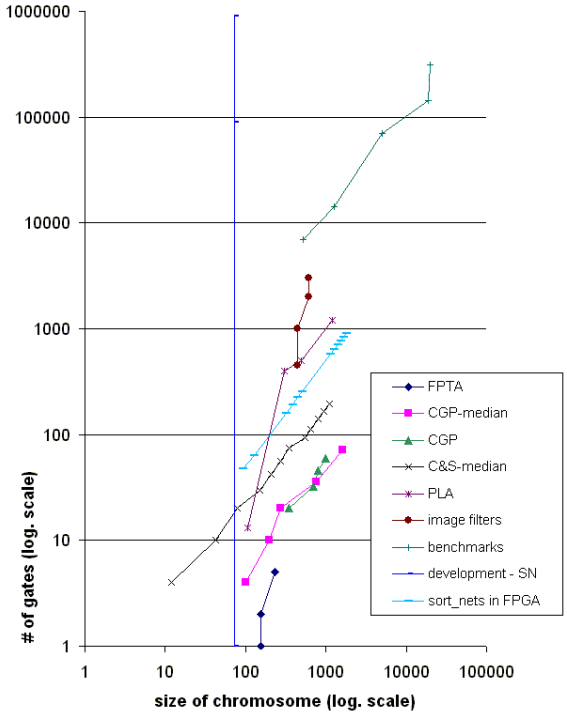

Figure 4. The complexity of the evolved circuits vs the size of chromosome

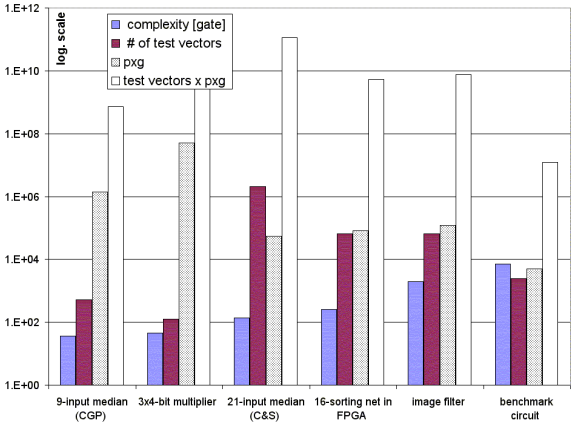

Figure 5. Comparison of circuits whose chromosomes have a similar length

we can obtain the number of test vectors that must be evaluated in the design process. This value corresponds to the *"time of evolution"*; however, only in case that all the experiments are performed on the same machine. In reality, different machines (different PCs and FPGAs) were utilized. We can observe that the "time of evolution" is very similar for almost all circuits. This parameter indicates how much time/resources we are able/willing to invest to the evolutionary design process.

5 Conclusions

In this paper, the state of the art in the area of evolutionary design of digital circuits was surveyed. In particular, the level of complexity of evolved circuits with respect to the size of the search space was investigated. It was demonstrated that innovative results can be obtained by all mentioned approaches, i.e. there is no reason to prefer some approaches. In order to complete the proposed analysis, digital circuits evolved with a specific aim (such as with the requirements on fault-tolerance, testability, polymorphism etc.) will be included.

Acknowledgment

The research was performed with the Grant Agency of the Czech Republic under contract No. 102/04/0737 *Modern Methods of Digital Systems Synthesis*.

References

[1] T. Aoki, N. Homma, and T. Higuchi. Evolutionary Synthesis of Arithmetic Circuit Structures. *Artificial Intelligence Review*, 20:199–232, 2003.

[2] M. Garvie and A. Thompson. Evolution of combinationial and sequential on-line self-diagnosing hardware. In *2003 NASA/DoD Conference on Evolvable Hardware*, pages 167–173. IEEE Computer Society, 2003.

[3] T. Gordon and P. Bentley. Evolving hardware. In A. Zomaya, editor, *Handbook of Nature Inspired and Innovative Computing*, pages 387–432. Springer Verlag, 2005.

[4] T. G. W. Gordon and P. J. Bentley. Towards development in evolvable hardware. In *Proc. of the 2002 NASA/DoD Conference on Evolvable Hardware*, pages 241–250, Washington D.C., US, 2002. IEEE Computer Society Press.

[5] P. Haddow and G. Tufte. Bridging the genotype–phenotype mapping for digital fpgas. In *Proc. of the 3rd NASA/DoD Workshop on Evolvable Hardware*, pages 109–115, Los Alamitos, CA, US, 2001. IEEE Computer Society.

[6] T. Higuchi, M. Iwata, D. Keymeulen, H. Sakanashi, M. Murakawa, I. Kajitani, E. Takahashi, K. Toda, M. Salami, N. Kajihara, and N. Otsu. Real-World Applications of Analog and Digital Evolvable Hardware. *IEEE Transactions on Evolutionary Computation*, 3(3):220–235, 1999.

[7] T. Higuchi, T. Niwa, T. Tanaka, H. Iba, H. de Garis, and T. Furuya. Evolving Hardware with Genetic Learning: A First Step Towards Building a Darwin Machine. In *Proc. of the 2nd International Conference on Simulated Adaptive Behaviour*, pages 417–424. MIT Press, 1993.

[8] B. I. Hounsell, T. Arslan, and R. Thomson. Evolutionary design and adaptation of high performance digital filters within an embedded reconfigurable fault tolerant hardware platform. *Soft Computing*, 8(5):307–317, 2004.

[9] L. Huelsbergen. Finding general solutions to the parity problem by evolving machine-language representations. In *Proc. of the Genetic Programming 1998 Conference*, pages 158–166, San Francisco, CA, 1998. Morgan Kaufmann.

[10] J. R. Koza et al. *Genetic Programming III: Darwinian Invention and Problem Solving*. Morgan Kaufmann, San Francisco, 1999.

[11] T. Kalganova. Bidirectional Incremental Evolution in Extrinsic Evolvable Hardware. In *Proc. of the 2nd NASA/DoD Workshop on Evolvable Hardware*, pages 65–74. IEEE Computer Society, 2000.

[12] Y. Kasai, E. Takahashi, M. Iwata, Y. Iijima, H. Sakanashi, M. Murakawa, and T. Higuchi. Adaptive waveform control in a data transceiver for multi-speed ieee1394 and usb communication. In *Evolvable Systems: From Biology to Hardware*, volume 3637 of *LNCS*, pages 198–204, 2005.

[13] J. Korenek and L. Sekanina. Intrinsic evolution of sorting networks: A novel complete hardware implementation for fpgas. In *Evolvable Systems: From Biology to Hardware*, volume 3637 of *LNCS*, pages 46–55. Springer Verlag, 2005.

[14] M. Murakawa et al. Evolvable hardware at function level. In *Proc. of the Parallel Problem Solving from Nature IV, Lecture Notes in Computer Science, vol. 1141*, pages 62–71, Berlin Heidelberg New York, 1996. Springer.

[15] T. Martinek and L. Sekanina. An evolvable image filter: Experimental evaluation of a complete hardware implementation in fpga. In *Evolvable Systems: From Biology to Hardware*, volume 3637 of *LNCS*, pages 76–85. Springer Verlag, 2005.

[16] J. Miller, D. Job, and V. Vassilev. Principles in the Evolutionary Design of Digital Circuits – Part I. *Genetic Programming and Evolvable Machines*, 1(1):8–35, 2000.

[17] T. Pecenka. Fittest_bench06 benchmark circuits. http://www.fit.vutbr.cz/∼pecenka/cirgen, 2006.

[18] T. Pecenka, Z. Kotásek, L. Sekanina, and J. Strnadel. Automatic discovery of rtl benchmark circuits with predefined testability properties. In *2005 NASA / DoD Conference on Evolvable Hardware*, pages 51–58. IEEE Computer Society, 2005.

[19] L. Sekanina. Evolutionary design space exploration for median circuits. In *EvoWorkshops*, volume 3005 of *LNCS*, pages 240–249. Springer, 2004.

[20] L. Sekanina. *Evolvable Components: From Theory to Hardware Implementations*. Natural Computing Series, Springer Verlag, 2004.

[21] L. Sekanina and M. Bidlo. Evolutionary design of arbitrarily large sorting networks using development. *Genetic Programming and Evolvable Machines*, 6(3):319–347, 2005.

[22] L. Sekanina and S. R. Zebulum. Evolutionary discovering of the concept of the discrete state at the transistor level. In *Proc. of the 2005 NASA/DoD Conference on Evolvable Hardware*, pages 73–78. IEEE Computer Society Press, 2005.

[23] A. Stoica. Evolvable hardware for autonomous systems. In *CEC Tutorial*, 2004.

[24] E. Stomeo, T. Kalganova, C. Lambert, N. Lipnitsakya, and Y. Yatskevich. On evolution of relatively large combinational logic circuits. In *2005 NASA/DoD Conference on Evolvable Hardware*, pages 59–66. IEEE Computer Society, 2005.

[25] A. Thompson, P. Layzell, and S. Zebulum. Explorations in Design Space: Unconventional Electronics Design Through Artificial Evolution. *IEEE Transactions on Evolutionary Computation*, 3(3):167–196, 1999.

[26] J. Torresen. A scalable approach to evolvable hardware. *Genetic Programming and Evolvable Machines*, 3(3):259–282, September 2002.

[27] J. Torresen. An evolvable hardware tutorial. In *Field Programmable Logic and Application*, volume 3203 of *LNCS*. Springer, 2004.

[28] J. Torresen, W. J. Bakke, and L. Sekanina. Recognizing speed limit sign numbers by evolvable hardware. In *Parallel Problem Solving from Nature*, volume 3005 of *LNCS*, pages 682–691. Springer, 2004.

[29] J. M. Tour. *Molecular Electronics*. World Scientific, 2003.

[30] V. Vassilev and J. F. Miller. Scalability problems of digital circuit evolution. In *Proc. of the 2nd NASA/DoD Workshop of Evolvable Hardware*, pages 55–64, Los Alamitos, CA, US, 2000. IEEE Computer Society.

[31] X. Yao. Following the Path of Evolvable Hardware. *Communication of the ACM*, 42(4):67–79, 1999.