Approximate Circuit Design by Means of Evolvable Hardware

Lukas Sekanina and Zdenek Vasicek
Brno University of Technology, Faculty of Information Technology, IT4Innovations Centre of Excellence
Brno, Czech Republic
Email: sekanina@fit.vutbr.cz, vasicek@fit.vutbr.cz

***Abstract*—This paper deals with evolutionary design of approximate circuits. This class of circuits is characterized by relaxing the requirement on functional equivalence between the specification and implementation in order to reduce the area on a chip or minimize energy consumption. We proposed a CGP-based automated design method which enables to find a good trade off between key circuit parameters (functionality, area and power consumption). In particular, the digital approximate circuits consisting of elementary gates are addressed in this paper. Experimental results are provided for combinational single-output circuits and adders where two different metrics are compared for the error assessment.**

I. Introduction

Evolvable hardware (EHW) is usually defined as a non-traditional (unconventional) search-based method for hardware design and adaptation that uses various bio-inspired computational intelligence paradigms. The main reasons why EHW has mainly been studied and developed include its ability to (i) provide novel designs hardly reachable by means of conventional methods, (ii) deliver good solutions for problems where the specification is inherently incomplete and any golden solution does not exist, and (iii) achieve adaptation/fault tolerance directly at the hardware level. Despite some success stories, as reported in e.g. [1], [2], it has not fully penetrated into mainstream hardware design tools. There are several reasons for this fact, in particular, the EHW method utilizes very time consuming algorithms, the scalability of resulting solutions is very limited and the whole process is too non-deterministic for some tasks. This situation also holds for the subfield of digital circuit evolution which this paper deals with.

However, researchers working on algorithms and methodologies that we could classify as *conventional* from the EHW perspective have recently shown an interest in a group of techniques that are currently known as *approximate computing*. In this area, the requirement of perfect functional behavior is relaxed because some users are willing to accept less-than-perfect results, errors are not observed as human perception capabilities are limited (e.g. in image, speech, music and video perception) or even a perfect solution is impossible to define and thus reach. If an approximate solution (i.e. a circuit) is employed then it is expected that significant benefits will be obtained in terms of energy savings, speedup or area on a chip [3], [4], [5].

Approximate circuits have initially been constructed manually, e.g. by removing those parts of existing fully functional designs that did not contribute to the result significantly. For example, approximate adders and multipliers have been reported for signal processing applications such as image filtering and compression [6]. Other approaches exploit timing induced errors due to voltage over scaling and over clocking. The current trend is to create general design methods capable of constructing approximate circuits which never exceed a predefined error [7], [4]. The error can be expressed by various metrics such as worst case error, average error, error probability etc. These 'error-oriented' approaches, however, represent only one of possible methodologies for approximate circuits design.

In this paper it is proposed to employ the evolutionary circuit design as a method for approximate circuits design. In particular, we utilize the fact that search-based design methods (i.e. evolutionary design) always provide a partially working solution even if resources needed for constructing a fully functional solution *are not* available. It has to be noticed that conventional methods do not usually provide any result when allocated resources are insufficient. The method targets on small and middle-size digital circuits represented at the gate or register-transfer (RT) level where EHW is capable of providing novel solutions w.r.t. conventional design tools [2]. The first approximate circuit C_1 is created by means of an EA which can use up to $n-1$ gates (i.e. components in case of the RT level) where n is the number of gates required for reaching a perfect functionality. Next approximations are created by EA where the amount of available gates is gradually reduced. The user thus obtains a set of approximate circuits, each of which typically exhibits different trade-off between the functionality and cost. Other criteria such as power consumption and delay can be included in the optimization framework.

The proposed method has been implemented as an extension of our system for evolutionary design of combinational circuits which is based on Cartesian Genetic Programming (CGP). The experimental evaluation consists of creating approximate circuits for two classes of circuits: combinational single-output circuits where the functionality degradation is simply measured as the number of bits incorrectly calculated for all possible input combinations and combinational adders where two different metrics are compared for the quality assessment. For all circuits, the decreasing quality is contrasted with decreasing power consumption and area on the chip.

The rest of the paper is organized as follows. Section II

surveys current research in the field of approximate circuits and some bio-inspired approaches that could be classified as a contribution to approximate computing. The proposed design methodology is introduced in Section III. Experimental framework is presented together with experimental results in Section IV. Conclusions are given in Section V.

II. Previous Research

In this section we will first present the field of approximate computing as it has been introduced by the community of researchers mainly developing and utilizing conventional/standard circuit design tools. Then we will survey approaches and results from the evolvable hardware literature that have something in common with this notion of approximate computing.

A. Approximate Circuits

Approximate circuits have been intensively studied in recent five years which can be documented by numerous papers on this topic published at major CAD conferences such as DAC and DATE. The concept has been know earlier, for example, Shih-Lien Lu dealt with approximate pipeline circuits in microprocessors in 2004 [3]. In approximate circuits, the requirement on functional equivalence between the specification and implementation is relaxed in order to accelerate computations, reduce the area on a chip or minimize the energy consumption. Over-scaling and functional approximation are two basic design techniques.

In case of *over-scaling*, these circuits are designed to be working perfectly under normal environment. However, their energy consumption can be reduced by voltage over-scaling. Similarly, performance can be increased when the circuit is over-clocked. Timing induced errors are due to the fact that some paths in the circuit fail to meet the delay constraints [8]. Both techniques can be combined.

Functional approximation means that the circuit is designed in such a way that it does not fully implement the logic behavior given by the specification. A simple method is to reduce the precision of computations in case of arithmetic circuits by ignoring the least significant bits. Other methods adopt logic synthesis scenarios in which implementations that satisfy the specification almost perfectly are sought, but the amount of resources is significantly reduced w.r.t. the perfectly working circuit. For example, a two-bit multiplier was manually constructed which consists of 5 gates only and exhibits the delay of $2d$ where d is a unit delay. Its output is correct for 15 out of 16 possible inputs. A usual conventional solution requires 8 gates and exhibits the delay of $3d$. This approximate multiplier has been used in larger multipliers and then employed in image processing applications. Reported power savings are impressive: 30%-50% for a mean error of 1.39% - 3.35% [6].

Manual re-design is a typical approach adopted to create approximate circuits [6]. Systematic methods for synthesis have been proposed recently [4]. The most recent SALSA methodology, presented at DAC 2012, starts with a RT level description of the exact version of the circuit and an error constraint that specifies the type and amount of error that the implementation can exhibit [7]. The methodology introduces the so-called Q-function which takes the outputs from both the original circuit and approximate circuit and decides if the quality constraints are satisfied. The Q-function outputs a single Boolean value. The SALSA algorithm attempts to modify the approximate circuit with the goal to keep the output of the Q function unchanged. The SALSA method iteratively and separately approximates the outputs of the approximate subcircuit by exploiting the fact that one output can remain unchanged for some primary inputs (the so-called observability don't cares are exploited). The main advantage is that these steps can be done by conventional synthesis tools.

Various approaches can be used to evaluate the quality of approximation. For some cases, e.g. arithmetic circuits, a golden solution (i.e. perfectly functional circuit) can be constructed. In other cases, e.g. image filters, no golden circuits exist. We will compare several error functions during our experiments. An approach for modeling and analysis of circuits for approximate computing which addresses the issues of comparison of approximate and conventional designs, accuracy calculations and verification has been proposed in [5]. It has to be stressed that traditional design and verification techniques are not directly applicable for approximate circuits.

B. Evolutionary Circuit Design

Some results that the evolvable hardware community has created so far can be understood as at least partially relevant for approximate computing.

Famously Thompson's evolutionary design of a tone discriminator circuit in the XC6216 FPGA can be considered as an early approach to approximate computing. Thompson's EA discovered a discriminator requiring significantly less resources than usual solutions would occupy in the same FPGA [9]. Though the evolved discriminator was fully functional, its robustness was limited. Higher sensitivity to fluctuations in environment (external temperature, power supply voltage) and dependability on a particular piece of FPGA were reported. Hence we can observe a trade off between the robustness and the amount of resources in the FPGA.

Miller has introduced a CGP-based method for finite impulse response (FIR) filter design [10]. During the evolutionary design process target filters are composed from elementary gates, ignoring thus completely the well-developed techniques based on multiply–and–accumulate structures. The main practical result of this approach is that the evolved filters are extremely area-efficient (and thus potentially energy efficient) in comparison with conventional filters. However, only partial functionally has been obtained. Evolved filters worked correctly only for a very limited amount of potential input signals. Another Miller's work addressed the problem of evolutionary design of digital circuits that can approximate real numbers [11]. Similarly to the previous approach, the circuits are composed from elementary gates.

Kneiper et al. investigated robustness of EHW-based classifiers [12]. A classifier system was reported which is able to cope with changing resources at run-time. During optimization, the number of pattern matching elements was modified and its influence on classification accuracy was studied. The performance and accuracy was recognized as sufficient as long as a certain amount of resources is present in the system.

Our previous work on multifunctional circuits has led to an adaptive FIR filter architecture which can adjust the filtering quality by means of polymorphic gates [13]. A similar concept has been employed in digital controllers that gracefully degrade when some inconvenient situations arise, e.g. when battery goes low or a chip temperature crosses some safe level [14].

The concept of inherent fault tolerance investigated in the EHW field seems to be very relevant to approximate computing. Inherent fault tolerance means that in case of a failure of a circuit element, the evolutionary algorithm is usually able to recover the original functionality using the remaining elements or using another member of the population which might be insensitive to the failure [15]. If a critical number of elements is damaged, the original function cannot fully be recovered; however, a partial functionality can be obtained. For instance, functional recovery of a quadrature decoder after a stuck-at-zero fault was demonstrated for a model of an FPGA [16].

We can sum up that the EHW method is in principle applicable for approximate circuit design assuming that the scalability problems of EHW can be eliminated for a particular application.

III. Proposed Methodology

As the proposed method is based on CGP we will first recall main principles of CGP for circuit evolution according to [17]. The overall design methodology will then be introduced in this section.

A. Circuit Evolution Using CGP

1) Encoding: The CGP approach utilizes a set of processing nodes (gates) arranged in n_c columns and n_r rows to represent a target circuit. Each node can perform one function taken from an a priori given finite set of functions Γ. A specific interconnection of the elements gives rise to a functional circuit. The circuit utilizes n_i primary inputs and n_o primary outputs. The circuit inputs as well as the output of each node in the grid have assigned a unique integer (index) according to which the interconnection is defined. Similarly, the functions of the nodes are identified by integer values. Since the circuits considered in this paper can be realized as combinational circuits in which no feedback is allowed, it is necessary to consider this limitation in the representation scheme. Each gate can be connected either to the output of a gate placed in previous l columns or to one of the circuit inputs. For example, if l equals 1, then it means that the input of a node in column c can only be connected to an arbitrary output of a node in column $c-1$ or to one of the circuit's inputs.

A candidate solution represented by CGP can be described by means of a sequence of integers specifying functions of the nodes and interconnection of the nodes. The CGP encoding consists of $n_c.n_r$ triplets (i_1, i_2, f) determining for each element its function f and input indices i_1 and i_2 which the element's inputs are connected to. It is assumed here that the gates have up to two inputs. The tail part of the CGP encoding contains n_o integers specifying the nodes where the primary outputs are connected to.

2) Search Method: For the search in the search space induced by the CGP representation, a technique denoted as *evolutionary strategy* is usually adopted [17]. The population consists of a finite number of chromosomes, each of which represents a candidate circuit using the CGP encoding. At the beginning of evolution, the first population is randomly generated and consequently evaluated. In order to create a new population, the chromosome of the highest-scored candidate circuit is selected as the new parent and by applying a *point mutation* (i.e. h genes are modified) offspring are generated in the count needed to fill up the rising population. The steps of the evolution loop are repeated in the next generations until either a circuit is found whose output fulfills the criteria specified by the designer or a maximal generation count is reached.

B. Fitness function

For the evolutionary design of digital circuits the fitness value of a candidate circuit is usually defined as:

$$fit1 = b \tag{1}$$

where b is the number of correct output bits obtained as response for all possible assignments to the inputs. In other words, the goal is to minimize the Hamming distance between the obtained truth table and required truth table. After reaching $b = b_{max} = n_o 2^{n_i}$ other objectives can be optimized, e.g. the number of gates.

Obviously, this method is not scalable, mainly because the evaluation time grows exponentially with the increasing number of primary inputs. Techniques proposed to overcome the scalability problem have been surveyed in [2].

If the problem is formulated as a symbolic regression problem, the goal of evolution is usually to minimize the *mean absolute error* of a candidate circuit response y and target response t. The fitness function is then defined as:

$$fit2 = \sum_{j=1}^{k} |y(j) - t(j)| \tag{2}$$

where k is the number of fitness cases.

C. The Overall Scheme

The proposed methodology addresses the problem of approximate circuit synthesis by constructing a set of circuits that attempt to approximate the target functionality using constrained resources. These circuits are then analyzed in terms functionality violation (error w.r.t. specification). The

circuit showing the best trade off between consumed resources and measured error is taken as the final solution. We will primarily study the trade off between the number of gates and error. As computing the power consumption is a very time consuming operation in the fitness function, it will not be used in the fitness function. However, we will measure the power consumption of the best evolved circuits at the end of evolution. The power consumption is calculated using the SIS tool (power_estimate command) which estimates, by means of symbolic simulation, the power dissipated in a circuit due to switching activity [18].

Let us assume that the area is expressed as the number of gates. Our method starts with evolutionary design of a target circuit C to find out the minimum number of gates n required for its implementation. Approximations of C are created in iterations. Approximate circuit C_1 is created by means of CGP which can consume up to $n-1$ gates. The goal of CGP is to minimize the error function. The evolution is stopped when the stopping condition is satisfied. The first approximation exhibits the error e_1. Approximate circuits $C_2 \dots C_k$ are then constructed wherein up to $n-2 \dots n-k$ gates are supplied for CGP. At the end the user has to choose the solution which exhibits the best trade off between functionality and the number of gates (or power consumption).

IV. Experimental Results

Experimental evaluation consists of evolutionary circuit design under sufficient and limited resources. The experiments are performed for two classes of circuits: single output circuits and combinational adders of various sizes (i.e. multiple output circuits). In both cases, the evolutionary design is firstly carried out to find the minimum number of gates needed for obtaining a perfectly working circuit. Resulting circuits are then considered as reference circuits for evolutionary design of approximate circuits.

A. Single Output Circuits

Circuits with a single output represent a special case in our context because there is only one reasonable way to define the fitness – as the number of correctly calculated output bits for all possible inputs (see eq. 1). Three circuits from the LGSynth93 suite (cm152, sym9 and t481) and 9-input majority circuit are used as benchmark problems. The 9-input majority is included as its optimal size is known to be 15 AND gates and 15 OR gates.

The CGP is set with sufficiently redundant resources, i.e. $n_r = 1$, $n_c = 35$ ($n_c = 40$ for the 9-input majority). Setting of other parameters corresponds with the recommendations given in [17]: $l = n_c$, $\lambda = 4$, $h = 5\%$, and $\Gamma = \{$BUF,NOT,AND,OR,XOR,NAND,NOR,XNOR$\}$. The initial population is always randomly generated. The maximum number of evaluations is $100 \cdot 10^6$. For the 9-majority a reduced set consisting of $\{$BUF,AND,OR$\}$ is utilized. The number of evaluations is $300 \cdot 10^6$.

Table I summarizes the results achieved by CGP in the task of circuit optimization under decreasing, but still sufficient resources (n_c is variable). The columns are the success rate in finding a fully functional solution according to eq. 1 (succ.rate), the minimum number of gates obtained (best n_g), the average number of evaluations to find a fully functional solution (meen n_e $[\cdot 10^6]$) and the average functionality. Both averages were computed from all runs. The success rate decreases with decreasing resources which corresponds with earlier findings about the importance of redundancy for CGP [17].

TABLE I
Statistical results of CGP for single-output circuits

maj9											
n_c	40	39	38	37	36	35	34	33	32	31	30
succ.rate	100%	100%	100%	98%	97%	90%	84%	63%	27%	9%	2%
best n_g	30	30	30	30	30	30	30	30	30	30	30
mean n_e	2.3	2.8	4.0	4.5	5.7	9.3	15.1	19.4	22.7	27.7	14.6
mean func.	100%	100%	100%	99%	99%	99%	99%	99%	99%	99%	98%
sym9											
n_c	35	34	33	32	31	30	29	28	27	26	25
succ.rate	56%	42%	40%	36%	25%	18%	14%	9%	11%	6%	1%
best n_g	24	24	24	23	24	24	24	24	24	24	24
mean n_e	66	86	87	83	75	88	141	63	142	261	196
mean func.	99%	98%	98%	98%	98%	97%	97%	96%	96%	95%	94%
cm152											
n_c	30	29	28	27	26	25	24	23	22	21	20
succ.rate	100%	100%	98%	96%	91%	84%	57%	34%	6%	0%	-
best n_g	21	22	22	21	21	21	21	21	22	-	-
mean n_e	1.9	2.2	2.7	3.8	3.8	4.8	6.5	6.6	8.0	-	-
mean func.	100%	100%	99%	99%	99%	99%	98%	97%	95%	94%	92%
t481											
n_c	30	29	28	27	26	25	24	23	22	21	20
succ.rate	62%	61%	53%	39%	25%	32%	15%	12%	0%	0%	-
best n_g	21	22	21	22	21	21	21	21	-	-	-
mean n_e	11.1	11.7	10.6	12.6	12.6	12.7	14.7	16.3	-	-	-
mean func.	95%	95%	95%	93%	91%	91%	90%	88%	86%	86%	84%

Table II gives the parameters of the best discovered circuits. The n_g column denotes the number of gates. Other columns are devoted to the best, worst and mean power consumption and area. These values were calculated by the SIS tool after mapping the evolved circuits to a reference gate set. We can observe that the power consumption significantly varies for resulting circuits. The best evolved circuits are significantly better in comparison with conventionally optimized circuits (see Table III).

TABLE II
Parameters of the best evolved single-output circuits

		power [uW]			area		
circ	n_g	best	worst	mean	best	worst	mean
maj9	30	170.2	259.8	216.0 ± 15.3	69.0	101.0	83.2 ± 5.0
cm152	21	143.4	460.1	235.5 ± 44.9	50.0	107.0	72.9 ± 9.0
sym9	23	432.1	1598.0	739.7 ± 179.1	115.0	165.0	138.4 ± 10.1
t481	21	197.1	871.4	303.2 ± 69.3	56.0	135.0	76.2 ± 10.0

Figure 1 shows the trade off between the number of gates and power consumption for approximate circuits where the number of gates decreases from the amount needed to obtain

TABLE III
RESULTS OF CONVENTIONAL OPTIMIZATION BY SIS

Circuit	PI	PO	gates	power [uW]
maj9	9	1	42	254.8
cm152	11	1	22	152.2
sym9	9	1	156	1148.0
t481	16	1	23	154.8

a fully functional circuit down to 1 gate. The best, worst and mean functionality calculated from 100 independent runs is given in percentage points. Similarly, the best, worst and mean *reduction* of power consumption is provided. As a reference value (0%) we took the average power consumption calculated from 10 fully functional circuits with the lowest power consumption. In this interpretation, 100% represents a circuit consuming 0 uW. The cross symbol indicates fully functional solutions. The circle symbol indicates a reduction in power consumption for a circuit which exhibits the highest fitness for a given number of gates.

In order to observe basic parameters of approximate circuits after removing 0%...90% gates we constructed Table IV. From all the evolved circuits we selected those circuits which show the highest functionality for a given reduction. If there are two or more circuits sharing the same functionality, power consumption was taken as the second criterion. Power reduction is computed with respect to the circuit given in the first column (reference). It has to be noted that even more impressive power reductions could be obtained if reference solutions were taken from Table III. The negative sign means that the corresponding circuit exhibits higher power consumption with respect to the reference circuit. This situation occurs when the switching activity has higher impact on the power consumption (in this case negative) than the reduction of number of gates.

B. Multiple Output Circuits

For multiple-output circuits more options exist to define the fitness function. For example, the Sum of Hamming Distances (SHD as defined in eq 1) is not well suited for evolution of arithmetic circuits since it does not take into account the importance of bits at circuit outputs. Reaching almost perfect fitness value (e.g. $n_o 2^{n_i} - 1$) may even imply a big worst case error because the MSB is incorrectly computed.

Hence we will compare SHD with another approach. We will calculate the sum of absolute differences (SAD) between the output vectors and required vectors. All vectors will be interpreted as unsigned integers. The approach is, in fact, expressed by eq. 2 where all possible input combinations are utilized to generate test cases. The goal is to compare SHD and SAD in terms of the computational cost and the quality of reachable circuits for combinational 3-bit, 3.5-bit (i.e. 3 bits + 4 bits) and 4-bit adders.

The CGP is set with sufficiently redundant resources, i.e. $n_r = 1$, $n_c = 20$ for 3-bit addres, 25 for 3.5 adder and 30 for 4-bit adders, $l = n_c$, $\lambda = 4$, $h = 5\%$. The set of gates is chosen as $\Gamma = \{$BUF,NOT,AND,OR,XOR,NAND,NOR,XNOR$\}$. The initial population is always generated randomly. CGP-based

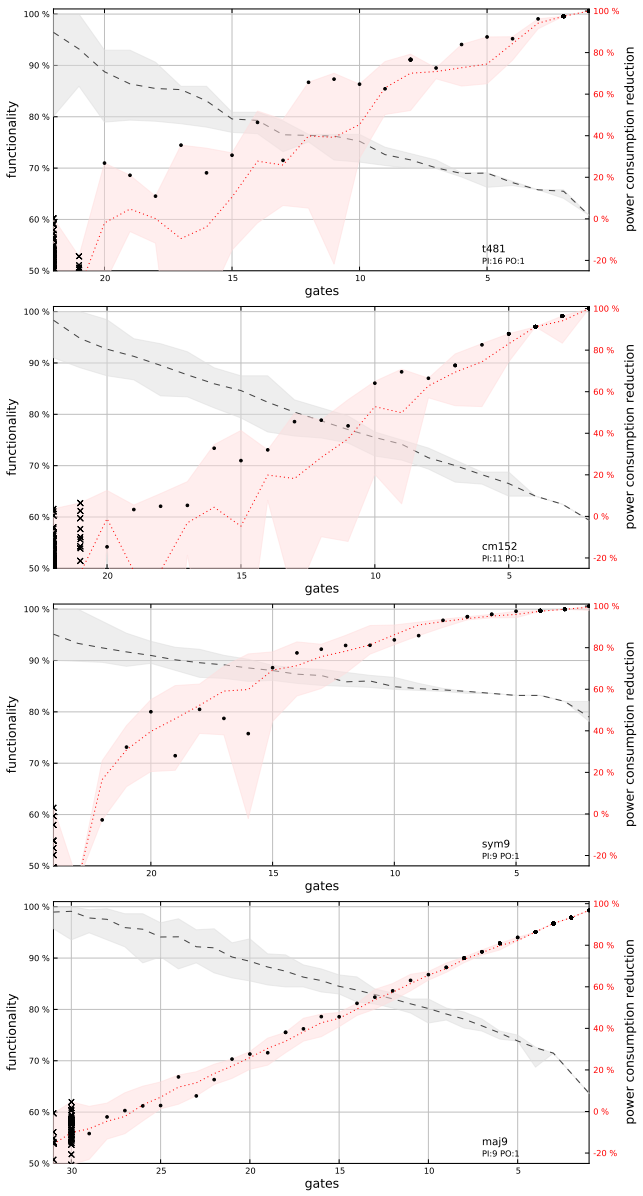


Fig. 1. Functionality with respect to power reduction for single-output circuits.

design is repeated 100 times for each adder and the fitness criterion (SHD and SAD). The evolution is terminated after reaching the predefined number of evaluations which is given in Table V.

First we used CGP to find the smallest adders under decreasing, but still sufficient resources. Table VI shows statistical results for SHD with the same meaning of rows as in Table I. One can again observe that the success rate decreases with decreasing redundant resources. Table VII provides the same statistical evaluation for the SAD-based fitness function. It can be stated that it is more difficult and more time consuming to find a correct adder using SAD in comparison with SHD. However, we will see later that it still makes sense to utilize SAD in the fitness function.

TABLE IV
ACHIEVABLE POWER REDUCTION FOR SINGLE-OUTPUT CIRCUITS.

maj9										
removed	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%
gates	(30)	(27)	(24)	(21)	(18)	(15)	(12)	(9)	(6)	(3)
func.	100%	98%	97%	92%	90%	86%	83%	80%	75%	71%
power	0%	-4%	12%	21%	34%	42%	55%	67%	79%	89%
saving	(170)	(177)	(148)	(133)	(110)	(97)	(75)	(55)	(35)	(18)
cm152										
removed	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%
gates	(21)	(18)	(16)	(14)	(12)	(10)	(8)	(6)	(4)	(2)
func.	100%	93%	89%	87%	81%	76%	73%	68%	64%	59%
power	0%	-1%	25%	24%	38%	55%	58%	73%	82%	90%
saving	(143)	(145)	(106)	(107)	(87)	(63)	(59)	(37)	(25)	(13)
sym9										
removed	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%
gates	(23)	(20)	(18)	(16)	(13)	(11)	(9)	(6)	(4)	(2)
func.	100%	93%	92%	90%	88%	87%	85%	83%	83%	82%
power	0%	19%	20%	34%	65%	68%	75%	91%	93%	97%
saving	(316)	(255)	(249)	(206)	(109)	(100)	(77)	(27)	(19)	(8)
t481										
removed	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%
gates	(21)	(18)	(16)	(14)	(12)	(10)	(8)	(6)	(4)	(2)
func.	100%	90%	85%	80%	76%	76%	72%	69%	67%	65%
power	0%	23%	33%	53%	68%	68%	77%	83%	86%	95%
saving	(240)	(183)	(161)	(113)	(74)	(76)	(53)	(38)	(33)	(11)

TABLE V
ALLOWED EVALUATIONS ($\times 10^6$) IN A SINGLE CGP RUN

Adder	SHD	SAD
3×3	10	100
4×3	50	500
4×4	100	1000

Table VIII summarizes the parameters of the best-evolved adders under SHD and SAD-based fitness functions. Columns give the number of gates and the the best, worst and mean power consumption and area. Both fitness functions led to the same (and probably minimal) solutions. The area and power consumption statistics are very similar.

In order to observe basic parameters of approximate circuits when resources are reduced by 10%...90%, we constructed Table IX and X. From all the evolved circuits we selected those circuits which show the highest functionality for a given reduction. Additional metrics were included to the tables to easily compare both approaches. Table IX contains the f_{SAD} row which gives the functionality according to the SAD-based fitness function. The *Avgad* is the average absolute error calculated from all incorrectly computed vectors. The *Maxad* denotes the worst case error a particular adder will produce for at least one input combination. The *errs* is the percentage of inputs vectors which an incorrect output is calculated for.

Figures 2 and 3 show the trade off between the number of gates and power consumption for approximate adders where the number of gates decreases from the amount needed to obtain a fully functional circuit down to 1 gate.

TABLE VI
EVOLUTIONARY DESIGN OF ADDERS WITH THE SHD-BASED FITNESS FUNCTION

adder 3,3														
n_c	20	19	18	17	16	15	14	13	12	11	10	9	8	7
succ.rate	100%	100%	100%	100%	100%	99%	97%	90%	79%	-	-	-	-	-
best n_g	12	12	12	12	12	12	12	12	12	-	-	-	-	-
mean n_e	0.2	0.1	0.2	0.2	0.3	0.3	0.6	0.7	1.3	-	-	-	-	-
mean f.	100%	100%	100%	100%	100%	99%	99%	99%	99%	96%	95%	92%	89%	87%
adder 3,4														
n_c	25	24	23	22	21	20	19	18	17	16	15	14	13	12
succ.rate	100%	100%	100%	100%	98%	100%	98%	98%	92%	84%	62%	50%	-	-
best n_g	14	14	14	14	14	14	14	14	14	14	14	14	-	-
mean n_e	0.3	0.3	0.3	0.5	0.6	0.7	0.5	1.0	1.5	2.6	1.9	5.3	-	-
mean f.	100%	100%	100%	100%	99%	100%	99%	99%	99%	99%	99%	98%	96%	94%
adder 4,4														
n_c	30	29	28	27	26	25	24	23	22	21	20	19	18	17
succ.rate	100%	100%	100%	100%	100%	100%	100%	100%	99%	98%	93%	84%	57%	46%
best n_g	17	17	17	17	17	17	17	17	17	17	17	17	17	17
mean n_e	0.8	0.8	0.9	0.9	1.2	1.2	1.4	1.6	1.8	2.6	3.5	4.2	4.8	11.4
mean f.	100%	100%	100%	100%	100%	100%	100%	100%	99%	99%	99%	99%	99%	98%

TABLE VII
EVOLUTIONARY DESIGN OF ADDERS WITH THE SAD-BASED FITNESS FUNCTION

adder 3,3 (SAD)												
n_c	20	19	18	17	16	15	14	13	12	11	10	9
succ.rate	46%	58%	24%	24%	12%	12%	8%	0%	0%	0%	0%	0%
best n_g	12	12	12	12	12	12	12	-	-	-	-	-
mean n_e	36	41	39	44	52	49	61	-	-	-	-	-
mean func.	99%	99%	98%	98%	98%	98%	98%	97%	97%	97%	97%	95%
adder 3,4 (SAD)												
n_c	25	24	23	22	21	20	19	18	17	16	15	14
succ.rate	69%	66%	60%	44%	28%	16%	24%	2%	2%	0%	0%	0%
best n_g	14	14	14	14	14	14	14	14	14	-	-	-
mean n_e	120	117	158	167	210	162	139	122	216	-	-	-
mean func.	99%	99%	99%	99%	99%	99%	99%	99%	99%	99%	98%	98%
addder 4,4 (SAD)												
n_c	30	29	28	27	26	25	24	23	22	21	20	19
succ.rate	48%	44%	20%	26%	18%	6%	16%	2%	8%	0%	0%	0%
best n_g	17	17	17	17	17	17	17	17	17	-	-	-
mean n_e	417	396	195	474	282	388	389	342	640	-	-	-
mean func.	99%	99%	99%	99%	99%	99%	98%	98%	98%	98%	98%	97%

The main conclusion is that the SHD-based fitness function leads to adders showing on average four times higher (mean as well as absolute) error in comparison with the adders evolved using the SAD-based fitness function. The *maxad* also indicates that the adders evolved using SHD even exhibit some errors in MSB which is not acceptable in practice. This fact is demonstrated by error tables given in Figure 4. The error tables were constructed for the best evolved 3-bit adders which occupy 50% resources needed for constructing a fully functional 3-bit adder.

V. CONCLUSIONS

We proposed a CGP-based automated design method that is able to address the problem of the evolutionary design of digital approximate circuits consisting of elementary gates.

TABLE VIII
THE BEST-EVOLVED ADDERS UNDER SHD AND SAD-BASED FITNESS FUNCTIONS

		power [uW]			area		
circ	gates	best	worst	mean	best	worst	mean
adder 3,3 (SHD)	12	108.5	372.5	196.9 ± 35.8	41.0	73.0	55.1 ± 5.8
adder 3,4 (SHD)	14	143.5	443.7	241.6 ± 44.9	52.0	82.0	65.0 ± 5.9
adder 4,4 (SHD)	17	191.1	681.4	301.0 ± 55.3	62.0	102.0	77.6 ± 7.2
adder 3,3 (SAD)	12	118.4	339.4	193.3 ± 35.6	41.0	69.0	54.0 ± 5.8
adder 3,4 (SAD)	14	163.0	556.7	242.2 ± 50.4	54.0	81.0	64.9 ± 5.9
adder 4,4 (SAD)	17	201.5	570.0	305.8 ± 65.7	60.0	102.0	76.9 ± 7.1

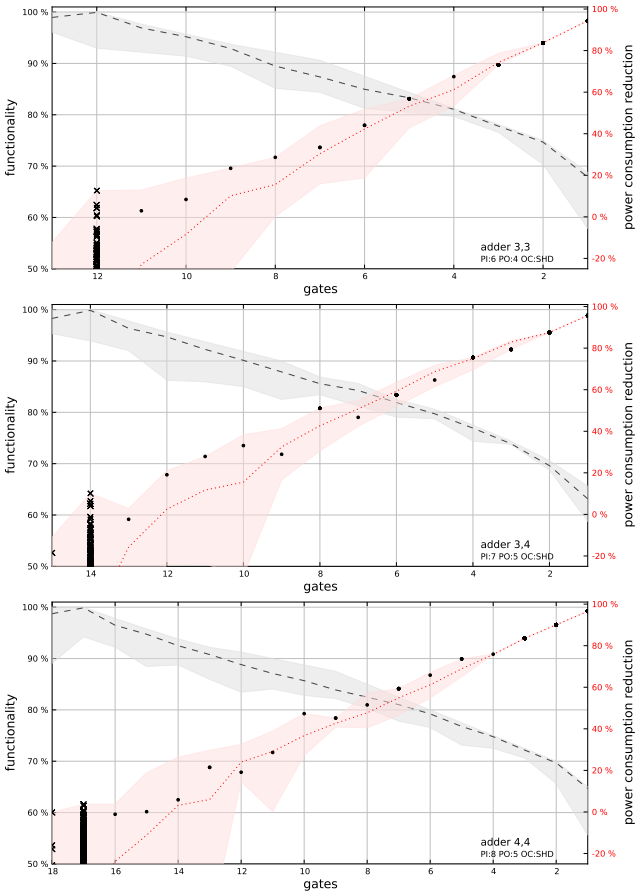


Fig. 2. SHD: Functionality w.r.t. power reduction for adders

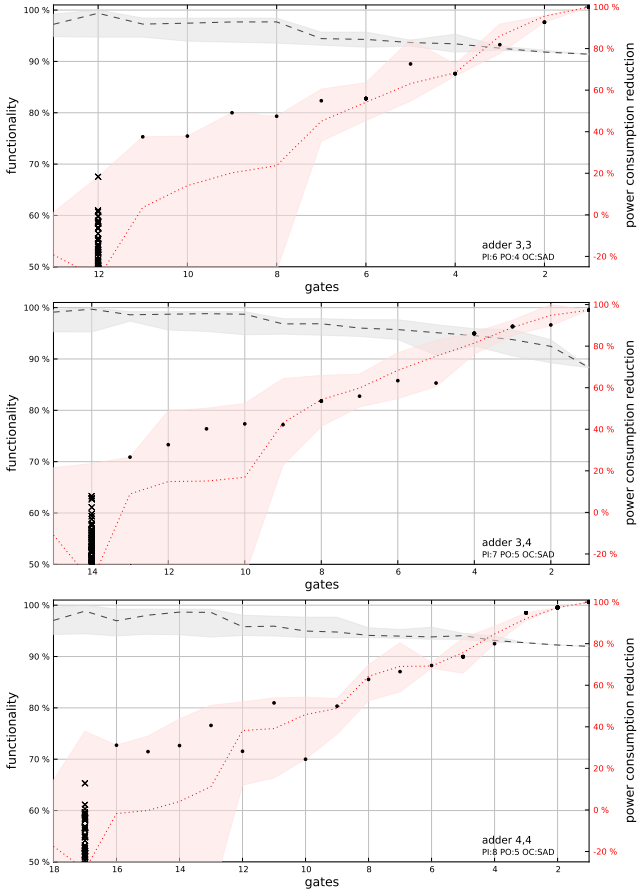
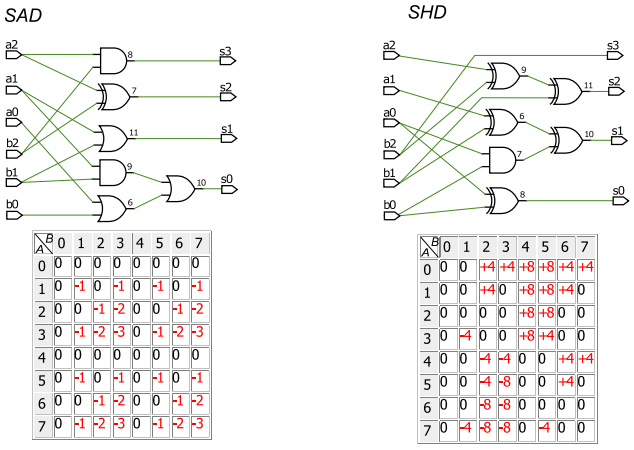


Fig. 3. SAD: Functionality w.r.t. power reduction for adders



A \ B	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	-1	0	-1	0	-1	0	-1
2	0	0	-1	-2	0	0	-1	-2
3	0	-1	-2	-3	0	-1	-2	-3
4	0	0	0	0	0	0	0	0
5	0	-1	0	-1	0	-1	0	-1
6	0	0	-1	-2	0	0	-1	-2
7	0	-1	-2	-3	0	-1	-2	-3

A \ B	0	1	2	3	4	5	6	7
0	0	0	+4	+4	+8	+8	+4	+4
1	0	0	+4	0	+8	+8	+4	0
2	0	0	0	0	+8	+8	0	0
3	0	-4	0	0	+8	+4	0	0
4	0	0	-4	-4	0	0	+4	+4
5	0	0	-4	-8	0	0	+4	0
6	0	0	-8	-8	0	0	0	0
7	0	-4	-8	-8	0	-4	0	0

Fig. 4. Best evolved 3-bit adders occupying 50% area and their error tables

Experimental results clearly showed that SHD-based fitness function, traditionally used by the evolvable hardware community, is not suitable for evolution of approximate adders since the mean error of resulting approximate adders is significantly higher in comparison with the SAD-based fitness function. We expect that the same holds for other circuits where bit positions are significant. Unfortunately, the SAD-based approach is more computationally demanding than SHD.

The proposed method provides many alternative solutions (in terms of the trade off between functionality and power consumption) which is not possible using existing conventional methods. This paper did not address the scalability problem of evolutionary design. More complex arithmetic circuits can be constructed by composing smaller approximate circuits as demonstrated e.g. in [6]. Our future research will be focused on evolutionary design of complex general approximate circuits. We plan to utilize functional level evolution, incremental evolution, equivalence checking and other approaches that have been used by the evolvable hardware community so far.

VI. ACKNOWLEDGMENTS

This work was supported by the Czech science foundation project P103/10/1517, the research programme MSM 0021630528, the Brno

TABLE IX
SHD: ACHIEVABLE POWER REDUCTION FOR ADDERS.

adder 3,3

#	100%	90%	80%	70%	60%	50%	40%	30%	20%	10%
gates	(12)	(10)	(9)	(8)	(7)	(6)	(4)	(3)	(2)	(1)
func.	100%	95%	93%	92%	90%	87%	81%	78%	75%	68%
power	0%	-4%	11%	17%	22%	34%	59%	65%	76%	88%
saving	(108)	(113)	(95)	(89)	(84)	(71)	(44)	(37)	(25)	(12)
fSAD	100%	93%	93%	90%	85%	84%	79%	82%	86%	87%
avgad	0.0	6.4	5.3	6.0	6.5	5.7	5.0	4.8	3.4	2.7
maxad	0	8	8	8	8	8	8	8	6	4
errs	0%	15%	18%	25%	34%	43%	65%	57%	62%	75%

adder 3,4

#	100%	90%	80%	70%	60%	50%	40%	30%	20%	10%
gates	(14)	(12)	(11)	(9)	(8)	(7)	(5)	(4)	(2)	(1)
func.	100%	95%	93%	90%	86%	85%	80%	77%	70%	65%
power	0%	9%	18%	20%	43%	39%	58%	69%	82%	91%
saving	(143)	(129)	(116)	(114)	(80)	(87)	(60)	(43)	(25)	(12)
fSAD	100%	98%	96%	86%	95%	94%	94%	93%	75%	75%
avgad	0.0	4.0	5.3	9.4	4.4	4.0	3.4	3.4	9.2	8.4
maxad	0	4	8	16	10	4	6	6	16	16
errs	0%	12%	18%	45%	34%	43%	53%	62%	85%	92%

adder 4,4

#	100%	90%	80%	70%	60%	50%	40%	30%	20%	10%
gates	(17)	(15)	(13)	(11)	(10)	(8)	(6)	(5)	(3)	(1)
func.	100%	95%	92%	90%	88%	85%	80%	77%	72%	65%
power	0%	-3%	17%	25%	43%	48%	62%	70%	80%	93%
saving	(191)	(197)	(156)	(143)	(107)	(99)	(71)	(56)	(37)	(12)
fSAD	100%	93%	92%	85%	84%	83%	83%	86%	86%	87%
avgad	0.0	12.1	8.7	11.7	11.0	9.3	7.8	6.4	6.1	4.7
maxad	0	16	20	20	20	20	16	12	14	10
errs	0%	16%	25%	39%	43%	57%	67%	67%	71%	87%

TABLE X
SAD: ACHIEVABLE POWER REDUCTION FOR ADDERS.

adder 3,3 (SAD)

#	100%	90%	80%	70%	60%	50%	40%	30%	20%	10%
gates	(12)	(10)	(9)	(8)	(7)	(6)	(4)	(3)	(2)	(1)
func.	100%	98%	98%	98%	95%	95%	95%	92%	92%	91%
power	0%	23%	37%	35%	44%	45%	60%	77%	90%	98%
saving	(118)	(90)	(74)	(76)	(65)	(64)	(47)	(27)	(11)	(1)
fSHD	100%	82%	82%	82%	68%	68%	70%	63%	58%	56%
avgad	0.0	1.0	1.0	1.0	1.6	1.6	1.7	1.5	1.6	1.8
maxad	0	1	1	1	3	3	3	3	4	4
errs	0%	25%	25%	25%	43%	43%	43%	75%	76%	78%

adder 3,4 (SAD)

#	100%	90%	80%	70%	60%	50%	40%	30%	20%	10%
gates	(14)	(12)	(11)	(9)	(8)	(7)	(5)	(4)	(2)	(1)
func.	100%	99%	99%	97%	97%	97%	96%	95%	93%	88%
power	0%	26%	34%	36%	49%	51%	58%	84%	88%	96%
saving	(163)	(119)	(106)	(102)	(82)	(78)	(67)	(25)	(18)	(5)
fSHD	100%	85%	85%	76%	73%	74%	62%	62%	62%	53%
avgad	0.0	1.0	1.0	1.6	1.6	1.7	1.5	1.7	2.1	4.0
maxad	0	1	1	3	3	3	3	4	4	11
errs	0%	25%	25%	43%	43%	43%	71%	78%	93%	92%

adder 4,4 (SAD)

#	100%	90%	80%	70%	60%	50%	40%	30%	20%	10%
gates	(17)	(15)	(13)	(11)	(10)	(8)	(6)	(5)	(3)	(1)
func.	100%	99%	99%	97%	97%	95%	95%	94%	92%	91%
power	0%	17%	31%	44%	13%	56%	64%	69%	93%	99%
saving	(201)	(166)	(137)	(112)	(175)	(86)	(71)	(61)	(13)	(1)
fSHD	100%	85%	85%	73%	74%	63%	62%	65%	55%	55%
avgad	0.0	1.0	1.0	1.6	1.7	2.0	1.8	3.0	2.8	2.9
maxad	0	1	1	3	3	7	4	7	8	8
errs	0%	25%	25%	43%	43%	69%	78%	57%	85%	88%

University of Technology project FIT-S-11-1 and the IT4Innovations Centre of Excellence CZ.1.05/1.1.00/02.0070.

REFERENCES

[1] P. C. Haddow and A. M. Tyrrell, "Challenges of evolvable hardware: past, present and the path to a promising future," *Genetic Programming and Evolvable Machines*, vol. 12, no. 3, pp. 183–215, 2011.

[2] L. Sekanina, "Evolvable hardware," in *Handbook of Natural Computing*. Springer Verlag, 2012, pp. 1657–1705.

[3] S.-L. Lu, "Speeding up processing with approximation circuits," *IEEE Computer*, vol. 37, no. 3, pp. 67–73, 2004.

[4] D. Shin and S. K. Gupta, "Approximate logic synthesis for error tolerant applications," in *Design, Automation and Test in Europe, DATE 2010*. IEEE, 2010, pp. 957–960.

[5] R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan, "Macaco: Modeling and analysis of circuits for approximate computing," in *2011 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2011, pp. 667–673.

[6] P. Kulkarni, P. Gupta, and M. D. Ercegovac, "Trading accuracy for power with an underdesigned multiplier architecture," in *VLSI Design*. IEEE, 2011, pp. 346–351.

[7] S. Venkataramani, A. Sabne, V. J. Kozhikkottu, K. Roy, and A. Raghunathan, "Salsa: systematic logic synthesis of approximate circuits," in *DAC*. ACM, 2012, pp. 796–801.

[8] D. Mohapatra, "Approximate computing: Enabling voltage over-scaling in multimedia applications," Ph.D. dissertation, Purdue University, 2011.

[9] A. Thompson, P. Layzell, and S. Zebulum, "Explorations in Design Space: Unconventional Electronics Design Through Artificial Evolution," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 3, pp. 167–196, 1999.

[10] J. F. Miller, "On the filtering properties of evolved gate arrays," in *1st NASA-DoD Workshop on Evolvable Hardware*. IEEE Computer Society, 1999, pp. 2–11.

[11] J. F. Miller and P. Thomson, "Evolving digital electronic circuits for real-valued function generation using a genetic algorithm," in *University of Wisconsin*. Morgan Kaufmann, 1998, pp. 863–868.

[12] T. Knieper, P. Kaufmann, K. Glette, M. Platzner, and J. Torresen, "Coping with resource fluctuations: The run-time reconfigurable functional unit row classifier architecture," in *Proc. of the 9th Conference on Evolvable Systems: From Biology to Hardware*, ser. LNCS, vol. 6274. Springer, 2010, pp. 250–261.

[13] L. Sekanina, R. Ruzicka, and Z. Gajda, "Polymorphic fir filters with backup mode enabling power savings," in *Proc. of the 2009 NASA/ESA Conference on Adaptive Hardware and Systems*. IEEE Computer Society, 2009, pp. 43–50.

[14] R. Ruzicka, "Dependable controller design using polymorphic counters," in *Proc. of 12th Euromicro Conference on Digital System Design*. IEEE Computer Society, 2009, pp. 355–362.

[15] G. Greenwood and A. M. Tyrrell, *Introduction to Evolvable Hardware*. IEEE Press, 2007.

[16] J. D. Lohn, G. V. Larchev, and R. F. DeMara, "A genetic representation for evolutionary fault recovery in virtex fpgas," in *Proc. of the 5th Conference on Evolvable Systems: From Biology to Hardware*, ser. LNCS, vol. 2606. Berlin: Springer, 2003, pp. 47–56.

[17] J. F. Miller, *Cartesian Genetic Programming*. Springer-Verlag, 2011.

[18] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A. Sangiovanni-vincentelli, "Sis: A system for sequential circuit synthesis," University California, Berkeley, Tech. Rep., 1992.