Search-Based Synthesis of Approximate Circuits Implemented into FPGAs

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

***Abstract*—Approximate computing is capable of exploiting the error resilience of various applications with the aim of improving their parameters such as performance, energy consumption and area on a chip. In this paper, a new systematic approach for the approximation and optimization of circuits intended for LUT-based field programmable gate arrays (FPGAs) is proposed. In order to deliver a good trade-off between the quality of processing and implementation cost, the method employs a genetic programming-based optimization engine. The circuits are internally represented and optimized at the gate level. The resulting LUT-based netlists are obtained using a commercial FPGA tool. In the experimental part, four commonly available commercial FPGA design tools (Xilinx ISE, Xilinx Vivado, Precision, and Quartus) and state-of-the-art academia circuit synthesis and optimization tool ABC are compared. The quality of approximated circuits is evaluated using relaxed equivalence checking by means of Binary decision diagrams. An important conclusion is that the improvements (i.e. area reductions) at the gate level are preserved by the FPGA design tools and thus the number of LUTs is also adequately reduced. It was shown that the current state-of-the-art synthesis tools provide (for some instances) the results that are far from an optimum. For example, a 40% reduction (68 LUTs) was achieved for 'clmb' benchmark circuit (Bus Interface) without introducing any error. Additional 43% reduction can be obtained by introducing only a 0.1% error.**

I. Introduction

In many applications based on FPGAs, it is crucial to minimize the circuit size in order to fit it into a given FPGA. This is actually a standard task supported by FPGA synthesis tools, which typically offer the users to specify various preferences and constraints during the circuit synthesis, optimization and mapping process. In the case of FPGAs, the area is usually measured by the number of look-up tables (LUTs) and delay is estimated from the logic levels and interconnect involved. The requirement of circuit optimization is especially important if the circuit is one of the modules developed for a dynamically reconfigured area of the FPGA.

It has recently been accepted in many application domains that errors can intentionally be introduced to the circuit with the aim of reducing its area, delay or power consumption. This approach, called *approximate computing* [1], typically exploits the error resilience which is present in the applications such as image processing, data mining, prediction or classification. This "approximation" feature is not currently supported by FPGA design tools. There is one tool enabling the user to specify what entity and to what extent it can be approximated (see, Axilog [2]). However, it has not been evaluated for FPGA designs.

In this paper, we propose a search-based method capable of optimizing and approximating combinational circuits. The method represents candidate circuits using directed acyclic graphs (DAG) and performs a global resynthesis and optimization by means of Cartesian genetic programming (CGP) [3]. The main reason for adopting CGP is that this method is capable of providing better trade-offs than commonly-used methods which was demonstrated, for example, in [4].

The proposed method performs the so called functional approximation. It is constructed as a general-purpose design automation method for combinational circuits, which means that the approximations are performed using the same procedure for all problem instances of a given class. The difference lies in the error computation which is specific for each application class. Various approximate implementations showing different compromises between considered system parameters are generated and presented to the user, whose responsibility is to choose the most suitable approximate solution for a given application. The proposed method thus differs from the single-purpose ("ad hoc") approximation approaches (such as [5]) developed to approximate a particular component.

The experimental scenario introduced to evaluate the method in the context of FPGA designs is as follows:

In order to obtain a reference implementation, a given fully functional circuit is described using a dataflow description in Verilog language and then synthesized and optimized by a standard FPGA design tool with the aim of minimizing the area. The resulting netlist consisting of LUTs is considered as a reference implementation. This step is denoted as Scenario I. in Fig. 1.

As our method does not put any restrictions to the magnitude of target error, it can be used to pre-optimize the original circuit providing that the zero target error is used. This two-step process is denoted as Scenario II.a in Fig. 1. A given fully functional circuit is firstly optimized by CGP. The optimized gate-level description is then synthesized and optimized by a standard FPGA design tool.

In the case that a functional non-equivalence between the original circuit and its final implementation can be tolerated, we can apply the approximate scenario. The fully functional circuit is optimized with the aim of minimizing the area provided that the error between the fully functional and

optimized implementation is bounded by the required error. Fig. 1 introduces this approach as Scenario III and Scenario II.b (the circuit already optimized by CGP is approximated).

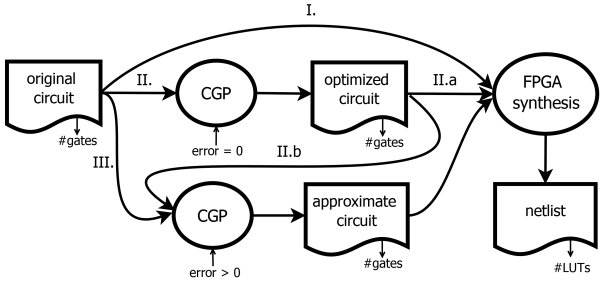

Fig. 1. Proposed experimental scenarios: I. – synthesis only; II.a – pre-optimization and synthesis; II.b – pre-optimization, approximation and synthesis; III. approximation and synthesis.

The proposed method is experimentally evaluated on circuits taken from LGSynth, ITC and ISCAS benchmark libraries. Introducing approximations to general logic could be dangerous in many cases (e.g. for controllers), but there is still an important class of circuits (such as pattern matching circuits, complex encoders and arbiters) in which the error can safely be exchanged for area reduction.

According to our best knowledge, the impact of a particular synthesis tool in the task of synthesis and approximation of circuits have never been evaluated. Another contribution of this paper is to demonstrate that the improvements (i.e. area reductions) obtained by CGP working at the gate level are, in fact, preserved by standard FPGA design tools and thus the number of LUTs is also adequately reduced.

II. Proposed Method

The objective is to minimize the area of circuits that have to be implemented into an FPGA, assuming that approximations are allowed. The proposed CGP-based approximation method, which is employed at the level of original circuits is described in next sections.

A. Circuit Representation

The search-based optimisation method could operate either at the level of gates (components) used in the original circuits or at the level of LUTs available in the FPGA. In this paper, we employed two-input nodes (gates) because it primarily leads to the fastest evaluation of candidate solutions and well-know types of search spaces.

A gate-level n_i-input/n_o-output circuit is represented using a directed acyclic graph which is encoded in a 1D array consisting of n_c gates. This array is internally stored using a string of integers, the so-called chromosome. The set of available logic functions is denoted Γ. For each gate, three integers are included in the chromosome – two labels specifying indices where the gate inputs are connected to and a code of function in Γ. The last part of the chromosome contains n_o integers specifying either the nodes where the primary outputs are connected to or logic constants which can directly be connected to the primary outputs. The main feature of this encoding is that while the size of the chromosome is constant, the size of circuits represented by this chromosome is variable as some gates can remain disconnected.

B. Search method

The search method follows the standard CGP approach [3]. The initial population P is seeded by the accurate circuit (p) and λ offspring circuits created by a point mutation operator modifying h genes of the parent individual p (h randomly chosen integers are replaced by randomly generated integers). In order to generate a new population, λ offspring individuals are again created by a point mutation operator. The parent is either the accurate circuit (in the first generation) or the best circuit of the previous generation (in remaining generations).

There are two design objectives for CGP: minimizing the functionality (error) and the number of gates.

C. Fitness function

For different types of circuits, specific fitness functions have to be constructed. In the case of arithmetic circuits, it is natural to minimize the arithmetic error. For general logic circuits, no additional information is usually available to establish a suitable error metric. Hence we measure the quality of a given approximation as the Hamming distance between the specification and approximate circuit. This can be obtained by converting the approximate circuit and the specification to corresponding Reduced Ordered Binary Decision Diagram (ROBDD) and calling suitable operators over ROBDD.

We employed the average Hamming distance which can be obtained from the ROBDD representation of the candidate circuit and the specification. Let A be a candidate circuit, S the specification (i.e. the accurate circuit) and σ be ROBDD which represents Boolean function $\mathcal{F}_S$ captured by S. To calculate the Hamming distance, corresponding outputs (y_j) of candidate circuit A, represented by ROBDD α, and σ are connected to a set of exclusive-or gates, i.e. $z_j = y_j^\alpha \oplus y_j^\sigma$, $j = 1 \dots n_o$. By means of the *Sat-Count* operation, which is available in common BDD packages such as Buddy [6], one can obtain the number of assignments b_j to the inputs which evaluate z_j to 1. The average Hamming distance between α and σ (and thus A and S) is then

$$fitness_2 = \sum_{j=1}^{n_o} \frac{|\{\vec{t} \in B^{n_i} : \mathcal{F}_{z_j}(\vec{t}) = 1\}|}{n_o 2^{n_i}} = \sum_{j=1}^{n_o} \frac{b_j}{n_o 2^{n_i}}. \quad (1)$$

D. Evolutionary Approximation

In order to evolve circuits showing different compromises between the error and size, a single-objective CGP is executed multiple times with different parameters. Resulting solutions are displayed using a Pareto front.

A two-stage procedure is employed for a given error e_i [7]. In the first stage, a given accurate circuit (S) is gradually modified by CGP (according to eq. 1) to exhibit error e_i providing that a 5% difference is tolerated with respect to e_i (tolerating a small error is acceptable; otherwise the search

TABLE I
PARAMETERS OF THE BENCHMARK CIRCUITS: THE NUMBER OF PRIMARY INPUTS (n_i) AND OUTPUTS (n_o), THE NUMBER OF GATES OBTAINED FROM ABC (n_G), THE NUMBER OF GATES OBTAINED USING CGP (n_G^{opt}) AND THE IMPROVEMENT.

circuit	apex7	clma	clmb	comp	duke2	b05	mm9a	mm9b	s1196	s1238	s349	s400	s420	s444	s510	s526	s526n	s832	s967	signet	large	ttt2	x1dn	x9dn
n_i	49	34	46	32	22	34	39	38	31	31	24	20	35	20	25	24	24	23	22	39	38	24	27	27
n_o	36	35	33	3	29	56	36	35	31	31	25	23	18	23	13	21	21	22	29	8	3	21	6	7
n_G	174	237	641	105	294	427	279	315	454	483	102	102	112	106	217	105	113	256	329	630	771	116	164	168
n_G^{opt}	166	228	**392**	96	269	**400**	263	**289**	**410**	**405**	96	96	112	98	204	95	98	**221**	**297**	**352**	**160**	99	**84**	**129**
impr.	4%	3%	38%	8%	8%	6%	5%	8%	9%	16%	5%	5%	0%	7%	5%	9%	13%	13%	9%	44%	79%	14%	48%	23%

could easily stuck in a local extreme). In the second stage, the number of gates is minimized, assuming that the error remains within the required range.

III. RESULTS

The following experimental methodology was utilized. First, the original circuits are optimized and approximated using CGP, as summarized in Fig. 1. Note that the original circuits are highly optimized by ABC. Then, the gate-level circuits are converted to Verilog netlists (one gate is represented by one logic expression) and synthesized. The goal of synthesis is to minimize the area, i.e. the number of (up to 6-input) LUTs. Results are presented for ABC and four commercial tools: Precision RTL 2015.1.6, ISE 14.7, Vivado 2015.2, and Quartus 14.1. The circuit size and delay are extracted from the resulting technology netlists. In the case of ABC, the synthesis and optimization is performed by 15 iterations of `resyn2` script, followed by mapping `if -K 6 -a`. In the case of Xilinx and Precision, the 6-LUT FPGA chip under label Virtex7 XC7VX330 was chosen for the implementation. The FPGA chip EP4S40G of Altera's StratixIV family was taken in Quartus. Only single-output LUTs are considered (i.e. LUT-combining is not permitted) to provide fair conditions for all tools. The implicit setup of CGP parameters follows the recommendations given in [3]: $\lambda = 4$, $h = 5$, $n_c = n_G$, $g_{max} = 10^4$. Γ includes all 2-input gates.

Table I gives the parameters of the chosen benchmark circuits. In most cases, the original circuit size (n_G) obtained by ABC was significantly reduced by CGP in Scenario II.a (see n_G^{opt}). The highest gate reduction was obtained for clmb (38%), signet (44%), large (79%) and x1dn (48%). This improvement is consistent with the fact that conventional logic synthesis and optimization tools provide far from optimum results for many problem instances [4], [8].

After optimizing the original circuits by CGP, we followed Scenario II.b in which CGP is also employed to approximate these circuits. Because of space restriction, results are presented for four errors ($e_1 = 0.2\%$, $e_2 = 0.4\%$, $e_3 = 0.6\%$ and $e_4 = 0.8\%$ in terms of the maximum Hamming distance). The maximum error is $2^{n_i} \cdot n_o$, where n_o is the number of primary outputs. Figure 2 shows resulting LUT counts for the original circuit, after the optimization by CGP ($e_0 = 0$), and for four target errors. The circuits are divided into two groups – smaller circuits with up to 60 LUTs (top) and more complex circuits with up to 200 LUTs (bottom).

Several results deserve a further analysis. Almost all fully functional circuits were improved at the gate level by CGP. However, a reduction counting more than 24 gates (Xilinx counts from 6 to 24 two-input gates for one LUT depending on the number of inputs used) was reported only in 11 cases (bold values in Table I). The small improvement in the number of gates obtained by CGP could led to increasing the number of LUTs after the synthesis. This is visible, for example, in circuit s510, which seems to be a difficult case for the synthesis tools because the spread in the number of LUTs is almost 100%. Another interesting case is signet, for which only Vivado provided a good result when optimizing the original circuit. It represents a rare case in which Scenario I provided slightly better result than Scenario II.b.

Allowing errors led to reducing the number of LUTs in all instances. However, the circuit response to the approximation is hard to predict. In some cases, the size is only slightly reduced (s349, s526n). In the case of comp and other more complex circuits (clma, clmb, duke2, s1196, s1238, s967, signet, and large), a considerable area reduction is visible even for the small error of 0.2%. As expected, the number of LUTs decreases with the increasing error. On the other hand, there are cases where some approximated circuits require more LUTs to be implemented compared to the approximations exhibiting lower error (see s510, ttt2, x9n, b05).

IV. DISCUSSION

In summary, pre-optimizing of fully functional circuits before the actual approximation is conducted seems to be a positive step. On the selected benchmark set, the average gate reduction is 16.3% without introducing any error to circuit functionality. At the level of LUTs, the average reduction is 10.7% which corresponds with 18 LUTs. Additional reduction of 26.7% LUTs was achieved by relaxing the requirement that the circuits are exactly correct and introducing a 0.2% error.

The complexity of the chosen benchmark circuits is roughly identical with circuits used for the evaluation of methods such as SASIMI [9], SALSA [10] and ABACUS [11]. Targeting these methods towards middle-size circuits is reasonable as approximations are typically introduced to carefully selected subcircuits which significantly contribute to power and area characteristics of the whole complex circuit.

Unfortunately, a direct comparison with other approximation methods is hard to perform because neither the implementations of the methods nor the results for common benchmark

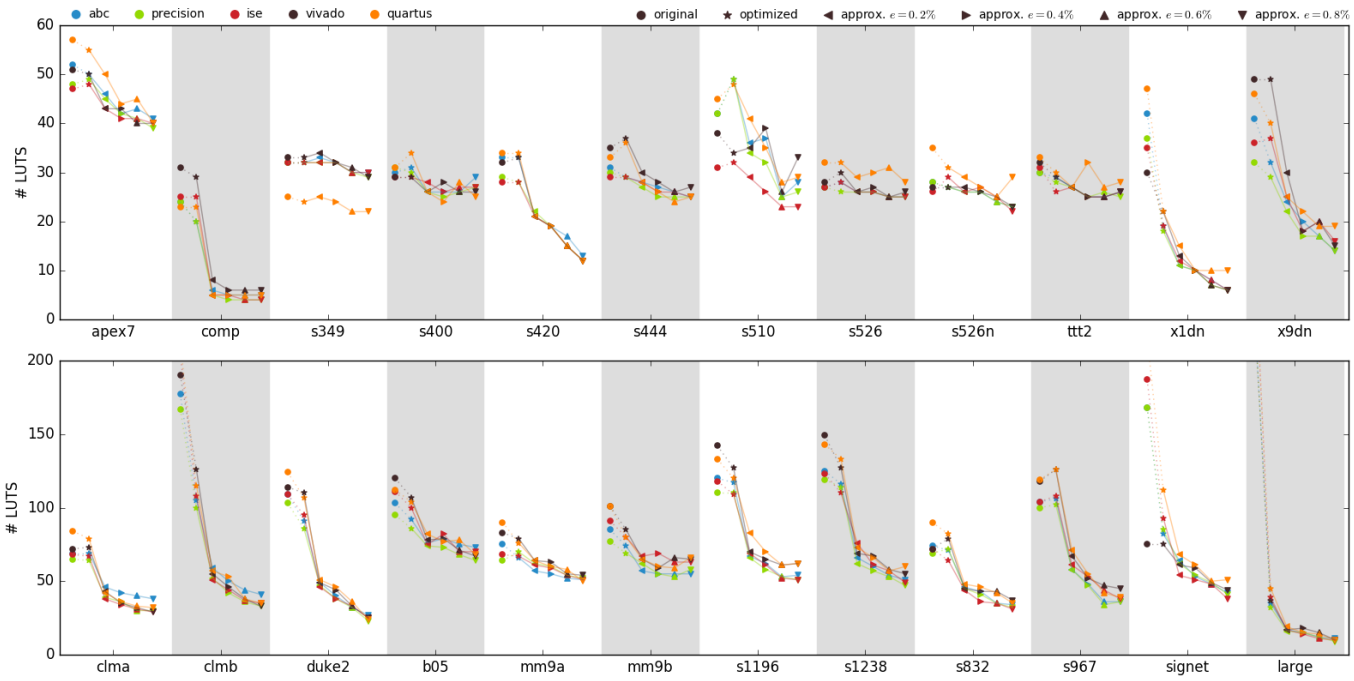

Fig. 2. The number of gates for original, optimized and approximate (errors of 0.2%, 0.4%, 0.6% and 0.8%) implementations of benchmark circuits.

circuits are available. The CGP runtimes are typically in the order of tens of minutes (2.93 GHz CPU). As no execution times are usually reported in the literature dealing with circuit approximation, we can only give the execution times of SALSA [10] which ranged from 4 minutes to 2.5 hours depending on the circuit complexity (2.29 GHz CPU).

An interesting result is that the gate-level optimization and approximation conducted by CGP is preserved by common FPGA synthesis tools, assuming that the original circuit is of a reasonable complexity. Interestingly, this result is valid even if the number of LUTs required to implement a given circuit is relatively low as it was demonstrated for 8 circuits consisting of less than 50 LUTs.

V. Conclusions

We introduced a CGP-based methodology enabling to approximate combinational circuits intended for FPGA implementations. The methodology was evaluated using standard benchmark circuits, where the error was expressed as the average Hamming distance and measured by means of BDDs. Due to the limited space we did not discuss the circuit delay; however, it has never been worsened by introducing the approximations. By modifying only the fitness (error) function the method can easily be extended to approximate other types of combinational circuits such as arithmetic circuits, image or signal processing components. We have shown that the results provided by commercial FPGA design tools can significantly be improved by introducing a pre-optimization phase conducted by CGP.

Acknowledgments

This work was supported by the Czech science foundation project 14-04197S.

References

[1] S. Mittal, "A survey of techniques for approximate computing," *ACM Computing Surveys*, pp. 1–34, 2016.
[2] A. Yazdanbakhsh, D. Mahajan *et al.*, "Axilog: Language support for approximate hardware design," in *Design, Automation and Test in Europe, DATE'15*. EDA Consortium, 2015, pp. 1–6.
[3] J. F. Miller, *Cartesian Genetic Programming*. Springer-Verlag, 2011.
[4] Z. Vasicek and L. Sekanina, "A global postsynthesis optimization method for combinational circuits," in *Proc. of the Design, Automation and Test in Europe, DATE*. EDA Consortium, 2011, pp. 1525–1528.
[5] P. Kulkarni, P. Gupta, and M. D. Ercegovac, "Trading accuracy for power in a multiplier architecture," *J. Low Power Electronics*, vol. 7, no. 4, pp. 490–501, 2011.
[6] J. Lind-Nielsen and H. Cohen. BuDDy - A Binary Decision Diagram Package. [Online]. Available: http://buddy.sourceforge.net
[7] Z. Vasicek and L. Sekanina, "Evolutionary design of complex approximate combinational circuits," *Genetic Programming and Evolvable Machines*, vol. 17, no. 2, pp. 169–192, 2016.
[8] J. Cong and K. Minkovich, "Optimality Study of Logic Synthesis for LUT-Based FPGAs," *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems*, vol. 26, no. 2, pp. 230–239, 2007.
[9] S. Venkataramani, K. Roy, and A. Raghunathan, "Substitute-and-simplify: a unified design paradigm for approximate and quality configurable circuits," in *Design, Automation and Test in Europe, DATE'13*. EDA Consortium San Jose, CA, USA, 2013, pp. 1367–1372.
[10] S. Venkataramani, A. Sabne *et al.*, "Salsa: systematic logic synthesis of approximate circuits," in *The 49th Annual Design Automation Conference 2012, DAC '12*. ACM, 2012, pp. 796–801.
[11] K. Nepal, Y. Li, R. I. Bahar, and S. Reda, "Abacus: A technique for automated behavioral synthesis of approximate computing circuits," in *Proceedings of the Conference on Design, Automation and Test in Europe*, ser. DATE '14. EDA Consortium, 2014, pp. 1–6.