A Global Postsynthesis Optimization Method for Combinational Circuits

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

***Abstract*—A genetic programming-based circuit synthesis method is proposed that enables to globally optimize the number of gates in circuits that have already been synthesized using common methods such as ABC and SIS. The main contribution is a proposal for a new fitness function that enables to significantly reduce the fitness evaluation time in comparison to the state of the art. The fitness function performs optimized equivalence checking using a SAT solver. It is shown that the equivalence checking time can significantly be reduced when knowledge of the parent circuit and its mutated offspring is taken into account. For a cost of a runtime, results of conventional synthesis conducted using SIS and ABC were improved by 20-40% for the LGSynth93 benchmarks.**

I. Introduction

Recent works on logic synthesis have shown that commonly used logic synthesis algorithms are not capable of efficient synthesis for some circuit classes, especially for large circuits and circuits containing hard-to-synthesize substructures [1]. The area of the synthesized circuits is of orders of magnitude higher than the optimum.

While synthesis of some large circuits can efficiently be accomplished using advanced decomposition and hierarchical techniques (see e.g. recent work on large-scale Boolean matching [2]), evolutionary algorithms have been adopted to synthesize circuits that are difficult for conventional synthesis [3]. As evolutionary synthesis allows for any transform to be performed over circuit representation it can implicitly discover compact circuit structures unreachable using conventional synthesis. Koza has reported tens of human-competitive results in various areas of science and technology obtained automatically using evolutionary design techniques, in particular using genetic programming [4]. This approach has mainly been adopted for analog circuit design [5], [6]. In case of digital logic synthesis, the evolutionary synthesis has led to innovative designs only for small circuits (with up to 8–12 inputs) mainly because of very time consuming and so non scalable fitness evaluation [7], [8]. Note that in a typical scenario, all possible assignments to the inputs have to be applied in order to evaluate a candidate circuit. In summary, evolutionary circuit design can produce quite compact designs for a cost of a runtime, whereas conventional synthesis (such as ABC or SIS) gets quickly stuck in a local optimum. Unfortunately, the currently used evolutionary approaches are not scalable.

978-3-9810801-7-9/DATE11/©2011 EDAA

In this paper, we propose a genetic programming-based circuit synthesis method that enables to globally optimize the number of gates in circuits that have already been synthesized using common methods such as ABC and SIS. We propose and compare two techniques allowing a significant acceleration of the fitness evaluation of a candidate logic design even for large circuits. Both methods utilize an equivalence checking algorithm to decide whether a candidate circuit is functionally equivalent or not. This operation is performed using a SAT solver. The methods differ in construction of the set of clauses that are submitted to the SAT solver. It is shown that the equivalence checking time can significantly be reduced when knowledge of the mutation operator is taken into account (the idea is that the parent circuit and its offspring created using the mutation operator share many subcircuits, i.e. in order to check their functional equivalence it is sufficient to check whether their non-shared subcircuits are functionally equivalent).

Apart from the synthesis of conventionally hard to synthesize circuits the method can be used to reduce the number of gates in already synthesized circuits. Experimental evaluation which has been performed using the LGSynth93 benchmark suite shows that the proposed method can significantly reduce the number of gates in circuits synthesized using ABC or SIS. For the cost of runtime, genetic programming is able to produce compact circuits that conventional synthesis is not able to reach.

II. Cartesian Genetic Programming

Cartesian Genetic Programming (CGP) is a widely-used method for evolution of digital circuits and programs [9]. A candidate entity (circuit) is modeled as an array of n_c (columns) $\times$ n_r (rows) of programmable nodes (gates). The number of inputs, n_i, and outputs, n_o, is fixed. Each node input can be connected either to the output of a node placed in the previous l columns or to one of the circuit inputs. Feedback is not allowed. Each node is programmed to perform one of n_a-input functions defined in the set Γ (n_f denotes $|\Gamma|$). Each node is encoded using three integers (x, y, z) where x denotes the index for the first input, y denotes the index for the second input and z is the function code. While the size of chromosome is fixed, the size of phenotype is variable (i.e. some nodes are not used). Every individual is encoded using $n_c \times n_r \times (n_a + 1) + n_o$ integers.

CGP utilizes evolutionary strategy ES($1 + \lambda$) that operates with the population of $1 + \lambda$ individuals. The initial solution (the seed) is constructed either randomly or by means of mapping of the circuit obtained from conventional synthesis (and specified e.g. in the BLIF format) to the CGP representation.

The fitness value of a candidate circuit is *traditionally* defined in CGP as $fitness = B + (n_c n_r - z)$, where B is the number of correct output bits obtained as response for all possible assignments to the inputs, z denotes the number of gates utilized in a particular candidate circuit and $n_c.n_r$ is the total number of gates available. The last term $n_c n_r - z$ is considered only if the circuit behavior is perfect, i.e. $B = n_o 2^{n_i}$. This approach is not scalable because of exponentially growing evaluation time with respect to the number of inputs.

III. Proposed Method

A. Overview

The proposed method starts with the circuit synthesis using a conventional synthesis algorithm. From the conventional and routinely used synthesis methods we have chosen the SIS [10] tool (version sis1.2) which provided in most cases better results than the ABC tool [11] (version abc70930) – see Table IV. The tools were applied with the standard setting. In order to improve their results it is usually useful to apply them on their own results iteratively [11].

The result of conventional synthesis is converted to the CGP representation and used to seed the initial population of CGP. The fitness calculation carried out by proposed method differs from the traditionally used formula. Instead of evaluating all possible assignments to the inputs, the candidate circuit is verified against the reference circuit (the result of conventional synthesis) as described in Section III-B.

B. Fitness Calculation

Since the satisfiability (SAT) solvers were improved during the last few years, the SAT-based equivalence has been used. In this case, the circuits to be checked are transformed into one Boolean formula which is satisfiable if and only if the circuits are functionally equivalent [12]. The SAT solvers assume that the equivalence checking problem is expressed using Boolean formula φ in conjunctive normal form (CNF).

For our purposes, the most suitable transformation of the circuit to CNF is represented by Tseitin's algorithm proposed in [13]. Each gate is converted to the CNF according to the equations given in Table I.

TABLE I
CNF representation of some common gates

Gate	Corresponding CNF representation
$y = \mathrm{NOT}(x_1)$	$(\neg y \vee \neg x_1) \wedge (y \vee x_1)$
$y = \mathrm{AND}(x_1, x_2)$	$(y \vee \neg x_1 \vee \neg x_2) \wedge (\neg y \vee x_1) \wedge (\neg y \vee x_2)$
$y = \mathrm{OR}(x_1, x_2)$	$(\neg y \vee x_1 \vee x_2) \wedge (y \vee \neg x_1) \wedge (y \vee \neg x_2)$
$y = \mathrm{XOR}(x_1, x_2)$	$(\neg y \vee \neg x_1 \vee \neg x_2) \wedge (\neg y \vee x_1 \vee x_2) \wedge$ $(y \vee \neg x_1 \vee x_2) \wedge (y \vee x_1 \vee \neg x_2)$
$y = \mathrm{NAND}(x_1, x_2)$	$(\neg y \vee \neg x_1 \vee \neg x_2) \wedge (y \vee x_1) \wedge (y \vee x_2)$
$y = \mathrm{NOR}(x_1, x_2)$	$(y \vee x_1 \vee x_2) \wedge (\neg y \vee \neg x_1) \wedge (\neg y \vee \neg x_2)$

Let C_A and C_B be combinational circuits, both with k inputs denoted as $x_1 \dots x_k$ and m outputs denoted as $y_1 \dots y_m$ and $y'_1 \dots y'_m$ respectively. For SAT based equivalence checking of two circuits, corresponding primary outputs y_i and y'_i are connected using the XOR-gate. The corresponding primary inputs are connected as well. The goal is to obtain one circuit that has only k primary inputs $x_1 \dots x_k$ and m primary outputs $z_1 \dots z_m$, $z_i = XOR(y_i, y'_i)$. In order to disprove the equivalence, it is necessary to identify at least one XOR-gate which evaluates to 1 for an input assignment, i.e. it is necessary to find an input assignment for which the corresponding outputs y_i and y'_i provide different values and thus $z_i = 1$. This can be done by checking one output z_i after another (i.e. a CNF is created and solved for each output separately) or by the all outputs approach (all outputs are connected using the m-input OR gate; thus one CNF is created and solved only).

Assume that C is a k-input/m-output circuit composed of n gates and the goal is to reduce the number of gates. The first step involves creating a reference solution by converting C to the corresponding CNF φ_1 using the approach described above. The fitness calculation consists of the following steps. A new instance of the SAT solver is created and initialized with the reference circuit. Then, a candidate solution is transformed to a list of clauses that are submitted into the SAT solver. The transformation includes reading the CGP representation according to the indexes of the nodes. If a node contributes to the phenotype, it is converted to the corresponding CNF according to Table I, otherwise it is skipped. In the next step, a miter circuit is created. This step requires to apply the XOR gate to each output pair and combine the XOR outputs by m-input OR gate. Finally, the SAT solver determines whether the submitted set of clauses is satisfiable or not. If the CNF is satisfiable, the fitness function returns 0 (the candidate circuit and the reference circuit are not equivalent); otherwise the number of utilized gates is returned.

C. Improved Equivalence Checking

One of the hardest cases is the equivalence checking of the combinational multipliers where the time needed to decide whether two multipliers are functionally equivalent is enormous even for instances with operands of modest size. In order to improve the performance of SAT solvers in this particular case, various techniques have been proposed in literature. A common goal of the proposed techniques is to modify (preprocess) the input CNF instance in order to decrease the proving effort of the SAT solver. For example, a preprocessing tool which derives implications according to the computed implication graph is proposed in [14].

In order to shorten the decision time in our case, we have proposed a new approach which utilizes knowledge of the dissimilarities between the reference circuit and checked circuit to reduce the size of CNF instance. Because the evolutionary approach is based on the modify-and-test approach, we can easily determine the difference between the parent individual and its modified (mutated) version during the mutation phase

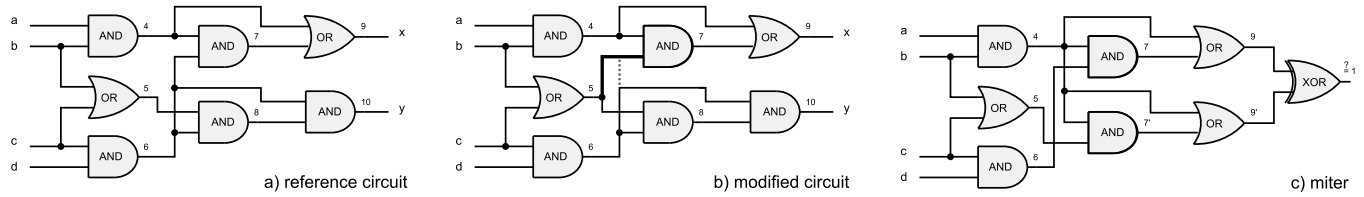

Fig. 1. Construction of the miter circuit (c) from the reference circuit (a) and modified (mutated) circuit (b)

(i.e. no additional processing to determine the dissimilarities is required). The proposed extension of the previous algorithm works as follows. We can construct a set Δ that contains the indexes of modified primary outputs or the indexes of such gates where at least one gene has been modified. An example of the reference circuit and modified circuit is shown in Fig 1. Here, $\Delta = \{7\}$ because the second input of gate 7 has been modified. Then, we can calculate auxiliary sets Δ_e and Δ_r. The Δ_e set contains the indexes of all the gates and outputs in the mutated circuit which are directly or indirectly connected to the outputs of the gates of Δ. The Δ_r set contains the indexes of all the gates in the reference parental circuit that contribute to any of the outputs listed in Δ_e. According to the knowledge of Δ_e and Δ_r, we can construct a set Δ_f containing the indexes of all the gates of mutated circuit that have to be included to CNF. All the sets can be constructed in linear time. In our example $\Delta_e = \{7, 9, x\}$, $\Delta_r = \{4, 6, 7, 9\}$ and $\Delta_f = \{9, 7, 5\}$. Finally, the SAT solver is applied on the clauses representing all the gates that are included in Δ_r and Δ_f, and only those outputs that are in Δ_e. In our example, the final circuit consists of 8 gates (7 + 1 XOR). This is a significant reduction with respect to the approach described in Section III-B that led to 17 gates (14 + 2 XOR + 1 OR).

IV. Results

In order to compare the time of evaluation for standard fitness function t_{CGP}, the proposed SAT-based fitness function t_{sat} (Section III-B) and the improved SAT-based fitness function t_{imp} (Section III-C), the problem of the combinational multiplier optimization has been chosen. Further experiments were performed using the LGSynth93 benchmark set (only circuits with more than 20 inputs were considered).

The CGP parameters are as follows: $\lambda = 2$, $\Gamma = \{\mathrm{BUF}, \mathrm{NOT}, \mathrm{AND}, \mathrm{OR}, \mathrm{XOR}, \mathrm{NAND}, \mathrm{NOR}, \mathrm{XNOR}\}$, $l = N_g$, 1 mutation/chromosome, $n_c = N_g$ and $n_r = 1$ where N_g is the number of gates of the reference (seed) circuit, i.e. in the circuit created using conventional synthesis. The circuits were mapped to the 2-input gates using SIS. The experiments were carried out on a cluster consisting of Intel Xeon X5670 2.4GHz processors using the Sun Grid Engine that enables to run the experiments in parallel. The MiniSAT 2 (version 070721, http://minisat.se) has been used as a SAT solver.

A. Fitness Evaluation Time

Table II gives the mean evaluation time for the three fitness functions. The results were obtained from fifty 10-minute independent runs of CGP in the task of w-bit combinational multiplier evolution. The size of the initial circuits (seed) that were synthesized using the SIS tool, is given in the N_g column. In the last column, we can observe a significant speedup achieved using the improved SAT-based fitness function.

B. LGSynth93 Benchmarks

The improved CNF construction (as described in Section III-C) enables very fast evaluation of candidate solutions. Hence more generations can be produced by CGP and thus more compact designs can be discovered within a given time. This phenomenon has been confirmed using the LGSynth93 benchmarks. Table III compares the number of gates obtained after 3 and 12 hours of optimization using the SAT-based fitness function (Ng_{sat}) and improved SAT-based fitness function (Ng_{imp}). The results clearly show the more runtime available, the more compact circuits obtained in comparison to the reference circuit (the N_g column) synthesized using SIS. The Ne columns give the mean number of evaluations which has been performed within a given time limit. The results were obtained from fifty independent runs of CGP.

Figure 2 shows convergence curves for two selected benchmark circuits – apex1 (the largest one) and ex4p (the highest number of inputs). We can observe that the improved SAT-based fitness function exhibits better convergence in comparison with the common SAT-based fitness function.

Table IV contains the best results obtained using the noncommercial and commercial tools. We have used the standard settings for the tools and technology library with the same set of gates as CGP. It can be seen that the commercial synthesis tools provide results that are comparable with the noncommercial synthesis tools such as ABC and SIS. The results from SIS and ABC were obtained by iterative application of the synthesis script (1000 iterations). None of the tools provide better result than CGP (when CGP is seeded using the first result provided by SIS) with the exception of apex5 where the number of gates is very similar.

TABLE II
THE MEAN EVALUATION TIME IN MILLISECONDS FOR THREE FITNESS FUNCTIONS IN THE TASK OF w-BIT MULTIPLIER EVOLUTION.

w	**PI**	**PO**	N_g	t_{cgp}	t_{sat}	t_{imp}	t_{sat}/t_{imp}
7	14	14	238	8	1	4	0,3
8	16	16	416	45	250	8	33,1
9	18	18	540	183	1 789	17	105,4
10	20	20	680	901	6 431	44	146,0
11	22	22	836	n/a	316 333	88	3 607,8

TABLE III

THE MIN. NUMBER OF GATES OBTAINED USING THE SAT-BASED FITNESS FUNCTION (Ng_{sat}) AND IMPROVED SAT-BASED FITNESS FUNCTION (Ng_{imp}) FOR THE LGSYNTH93 BENCHMARKS. THE Ne COLUMNS GIVE THE MEAN NUMBER OF EVALUATIONS IN MILLIONS.

				3h runtime						12h runtime					
circuit	**PI**	**PO**	N_g	Ng_{sat}	Ng_{imp}	**impr.**	Ne_{sat}	Ne_{imp}	**speedup**	Ng_{sat}	Ng_{imp}	**impr.**	Ne_{sat}	Ne_{imp}	**speedup**
apex1	45	45	1408	1179	946	20%	0,22	0,50	2,3	921	847	8%	0,22	0,49	2,2
apex2	39	3	235	104	93	11%	2,66	10,04	3,8	98	90	8%	3,20	10,77	3,4
apex3	54	50	1405	1280	1099	14%	0,23	0,54	2,4	1167	1038	11%	0,21	0,52	2,5
apex5	117	88	784	675	618	8%	0,93	2,22	2,4	633	613	3%	1,02	2,21	2,2
cordic	23	2	67	32	32	0%	10,44	17,46	1,7	32	32	0%	13,59	17,84	1,3
cps	24	109	1128	870	643	26%	0,32	0,81	2,5	698	585	16%	0,36	0,80	2,2
duke2	22	29	430	286	264	8%	0,98	1,79	1,8	268	260	3%	1,22	1,92	1,6
e64	65	65	192	133	138	-4%	3,52	2,39	0,7	129	129	0%	3,37	2,46	0,7
ex4p	128	28	500	404	368	9%	1,69	6,79	4,0	394	349	11%	1,96	7,08	3,6
misex2	25	18	111	76	73	4%	8,48	12,28	1,4	70	71	-1%	9,97	13,36	1,3
vg2	25	8	95	79	80	-1%	6,23	5,83	0,9	74	78	-5%	5,09	5,83	1,1

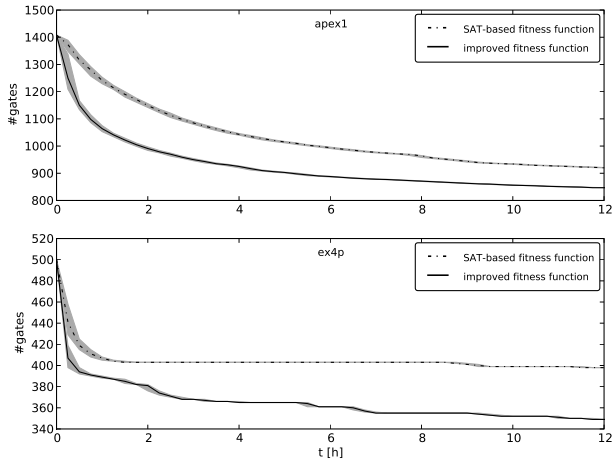

Fig. 2. Convergence curves for the apex1 and ex4p benchmarks. The mean, minimum and maximum number of gates from 50 independent runs of CGP for two variants of fitness function.

TABLE IV

THE MIN. NUMBER OF GATES OBTAINED USING THE NONCOMMERCIAL TOOLS (SIS, ABC), COMMERCIAL TOOLS (C1,C2,C3) AND THE PROPOSED METHOD.

circuit	**SIS**	**ABC**	**C1**	**C2**	**C3**	**CGP**	**impr.**
apex1	1394	1862	1439	**1272**	1368	847	33,4%
apex2	**151**	225	221	195	299	90	40,4%
apex3	1405	1737	1494	**1332**	1515	1038	22,1%
apex5	751	768	728	**609**	921	613	-0,7%
cordic	67	61	67	**49**	90	32	34,7%
cps	1128	1109	1150	975	**967**	585	39,5%
duke	406	**356**	417	366	357	260	27,0%
e64	192	384	**183**	191	255	129	29,5%
ex4p	488	523	468	**467**	555	349	25,3%
misex2	111	121	94	**89**	108	71	20,2%
vg2	95	113	88	**83**	109	78	6,0%

V. CONCLUSIONS

We have proposed a new global optimization method for combinational circuits. Applying the SAT solver in the fitness function allowed us to significantly reduce the computational requirements of the fitness function and thus evolve much larger circuits than any other evolutionary design method is capable of. Furthermore, we significantly reduced the number of gates in the solutions that can be delivered by conventional synthesis methods. The only cost is an additional (post synthesis) time consuming computation.

VI. ACKNOWLEDGMENTS

This work was partially supported by the Grant Agency of the Czech Republic under contract No. P103/10/1517 and by the research programme MSM 0021630528.

REFERENCES

[1] P. Fiser and J. Schmidt, "Small but nasty logic synthesis examples," in *Proc. 8th Int. Workshop on Boolean Problems*, 2008, pp. 183–190.

[2] H. Katebi and I. L. Markov, "Large-scale boolean matching," in *Design, Automation and Test in Europe, DATE 2010*. IEEE, 2010, pp. 771–776.

[3] P. Fiser, J. Schmidt, Z. Vasicek, and L. Sekanina, "On logic synthesis of conventionally hard to synthesize circuits using genetic programming," in *Proc. of the 13th Int. IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems*. IEEE, 2010, pp. 346–351.

[4] J. R. Koza, "Human-competitive results produced by genetic programming," *Genetic Programming and Evolvable Machines*, vol. 11, no. 3–4, pp. 251–284, 2010.

[5] T. McConaghy, P. Palmers, G. G. E. Gielen, and M. Steyaert, "Simultaneous multi-topology multi-objective sizing across thousands of analog circuit topologies," in *DAC 2007*. IEEE, 2007, pp. 944–947.

[6] A. Das and R. Vemuri, "A graph grammar based approach to automated multi-objective analog circuit design," in *DATE 2009*. IEEE, 2009, pp. 700–705.

[7] T. Aoki, N. Homma, and T. Higuchi, "Evolutionary Synthesis of Arithmetic Circuit Structures," *Artificial Intelligence Review*, vol. 20, no. 3–4, pp. 199–232, 2003.

[8] A. P. Shanthi and R. Parthasarathi, "Practical and scalable evolution of digital circuits," *Applied Soft Computing*, vol. 9, no. 2, pp. 618–624, 2009.

[9] J. F. Miller and P. Thomson, "Cartesian Genetic Programming," in *Proc. of the 3rd European Conference on Genetic Programming EuroGP2000*, ser. LNCS, vol. 1802. Springer, 2000, pp. 121–132.

[10] 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.

[11] Berkley Logic Synthesis and Verification Group, *ABC: A System for Sequential Synthesis and verification*. [Online]. Available: http://www.eecs.berkeley.edu/˜ alanmi/abc/

[12] E. Goldberg, M. Prasad, and R. Brayton, "Using SAT for combinational equivalence checking," in *DATE '01: Proceedings of the conference on Design, automation and test in Europe*. Piscataway, NJ, USA: IEEE Press, 2001, pp. 114–121.

[13] G. S. Tseitin, "On the complexity of derivation in propositional calculus," in *Studies in Constructive Mathematics and Mathematical Logic, Part II*, 1968, pp. 115–125.

[14] F. V. Andrade, L. M. Silva, and A. O. Fernandes, "Improving SAT-based combinational equivalence checking through circuit preprocessing," in *26th International Conference on Computer Design, ICCD 2008*, 2008, pp. 40–45.