Multiobjective Evolution of Approximate Multiple Constant Multipliers

Jiri Petrlik
Brno University of Technology
Faculty of Information Technology
IT4Innovations Centre of Excellence
Bozetechova 2, 612 66 Brno, Czech republic
Email: ipetrlik@fit.vutbr.cz

Lukas Sekanina
Brno University of Technology
Faculty of Information Technology
IT4Innovations Centre of Excellence
Bozetechova 2, 612 66 Brno, Czech republic
Email: sekanina@fit.vutbr.cz

***Abstract*—Multiple constant multiplier (MCM) is a digital circuit which multiplies its single input by N constants. As MCMs are composed of adders and shifters, their implementation cost is relatively low. In this paper, we propose a method for design of approximate multiple constant multipliers where the requirement on functional equivalence between the specification and implementation is relaxed in order to further reduce the area on a chip or minimize delay. The proposed method is based on multiobjective Cartesian Genetic Programming. It provides many trade-off solutions among accuracy, area and delay.**

I. Introduction

Multiple constant multiplier (MCM) is a digital circuit which multiplies its single input by N constants. It consists of adders, subtractors and shifters. As no multipliers are used, the hardware implementation of MCMs is quite inexpensive. MCMs are often utilized in low power finite-impulse-response (FIR) filters. In order to design a MCM, various heuristics have been proposed [1]. In this task, the main challenge is to minimize the number of adders/subtractors and delay for complex problem instances. It was also shown that MCMs can successfully be designed by evolutionary methods such as Cartesian Genetic Programming (CGP). In some cases the evolutionary methods can provide even better results than the best heuristics [2], [11].

In approximate circuits the requirement on functional equivalence between the specification and implementation is relaxed [3]. For these circuits it is possible to sacrifice a small amount of accuracy in order to obtain a solution which exhibits properties unreachable by fully functional solutions. 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. The current trend is to create general systematic design methods capable of constructing approximate circuits which never exceed a predefined error [4].

In this paper, we propose a method which can provide MCMs multiplying the input by slightly different constants than the specification requires. It leads to the delay and/or component reduction in comparision with the exact solution. Our method for the MCM design is multiobjective which means that we optimize a circuit with respect to more objectives. If the optimized objectives are conflicting, the goal of multiobjective optimization is to find many trade-offs among these objectives. In our work, we focus on the situation in which the accuracy and other parameters like the number of components or delay are optimized at the same time. At the end of the optimization we should find many so-called Pareto optimal trade-offs.

The rest of this paper is organized as follows. Section II describes main principles of CGP, especially the multiobjective CGP. In Section III, the proposed method is introduced. Section IV contains experimental results and discussion. Conclusions are given in Section V.

II. Multiobjective Cartesian Genetic Programming

A. Cartesian Genetic Programming

Cartesian genetic programming is a branch of genetic programming. In CGP a candidate solution is represented as a directed acyclic graph, which differs from traditional genetic programming, where the solution is represented as a tree [5]. It has been shown that CGP can successfully be used for design of variety kinds of circuits such as arithmetic circuits, filters, etc. [6]. The CGP representation is based on a grid of computational nodes. Each of these nodes can hold one of predefined functions and its inputs can be connected either to the outputs of nodes of previous columns or to the primary inputs. Information about functions and interconnection of the nodes is represented in an array of integer values called the chromosome.

The search algorithm commonly used in CGP is inspired in the $(1+\lambda)$ evolutionary strategy [6]. It works as follows: (1) At the beginning $1+\lambda$ randomly generated solutions are created to form the initial population. (2) The quality of solutions is then evaluated using the fitness function. (3) The best solution (the parent) is identified. (4) New λ solutions are created using a mutation operator from the parent. (5) If the termination condition is not fulfilled, then step 2 is taken, otherwise, the fittest solution is the result of CGP. The mutation operator randomly determines α items (genes) in the chromosome and changes their values.

B. Multiobjective CGP

CGP has been proposed as a single objective evolutionary design method [5]. A multiobjective optimization problem is defined as

$$\begin{array}{rll} \text{minimize/maximize} & f_m(x), & m = 1, 2, ..., M \\ \text{subject to} & g_j(x) \geq 0 & j = 1, 2, ..., J \\ & h_k(x) = 0 & k = 1, 2, ..., K \end{array}$$

where f_i are optimized functions, which we try to minimize or maximize. The solutions must fulfill some constraints to be acceptable. We can define two kinds of constraints. The inequity constraints are defined by functions g_j and equity constraints are defined by functions h_k [7].

Many evolutionary methods for the multiobjective optimization have been proposed. Most of them are based on the idea of Pareto dominance and Pareto optimality. Pareto dominance can be used for comparisons of solutions with multiple objectives. We can say that the solution x dominates the solution y, if the following two conditions hold. (1) The solution x is no worse than y in all objectives. (2) The solution x is strictly better for at least one objective than y. Pareto-optimal solution x is a solution for which any other solution y which dominates x does not exist. The goal of multiobjective optimization is to find Pareto-optimal solutions. If the objectives are conflicting, there can exist many Pareto optimal solutions. Then the goal of the multiobjective optimization is to find possibly all Pareto optimal trade-offs among them.

Evolutionary algorithms are usually successful in multiobjective optimization, because they internally work with a set of candidate solutions. NSGA-II is a genetic algorithm which can be used for multiobjective optimization [8]. It is based on sorting individuals according to the dominance relation. A method for multiobjective CGP was proposed in [9]. It uses a modified NSGA-II algorithm instead of the $1 + \lambda$ search strategy. A few modifications of the NSGA-II is proposed. One of them is such that the correct functionality is considered as a constraint. It means that if a circuit is not functionally correct, the solution is infeasible and it is not evaluated further. Whilst the algorithm is searching for functionally correct circuits, it uses the $(\mu + \lambda)$ evolutionary strategy.

III. Proposed method

In previous approaches to multiobjective CGP, the functionality was considered as a constraint (which must be fulfilled) rather than an optimized objective. Our approach relaxes this constraint. It allows MCM circuits to multiply the input by slightly different constants than it is defined in the specification. The method enables us to obtain many Pareto optimal trade-offs among accuracy and other objectives such as the number of components and delay. At the end of evolution, the designer can choose a proper circuit, which shows an acceptable accuracy and area/delay.

The goal is to create a MCM multiplier with N constants. Let $y_1, y_2, \ldots, y_n$ be desired constants and let $y'_1, y'_2, \ldots, y'_n$ be constants implemented by a candidate circuit. In the multiobjective CGP, there is defined the following constraint

$$\sum_{i=1}^{N} |(y_i - y'_i)| = 0 \tag{1}$$

which is fulfilled by candidate circuits which multiply the input by those constants that are given in the specification. We propose to redefine this constraint to

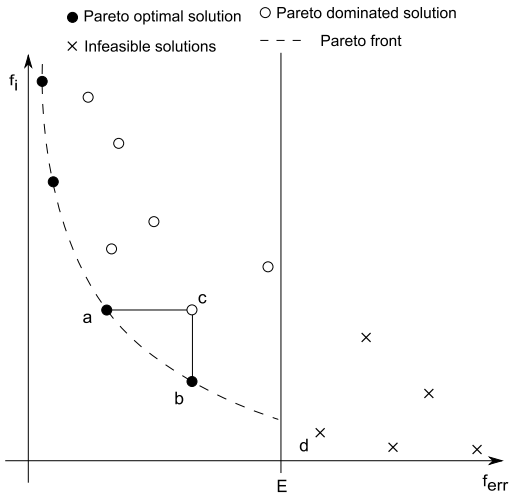

Fig. 1. Relation between optimized function f_{err} and acceptable treshold E. When $f_{err} > E$ then the solution is infeasible.

$$\sum_{i=1}^{N} (y_i - y'_i)^2 < E \tag{2}$$

where E defines the maximal acceptable difference among the desired outputs and the outputs of the candidate circuit. This constant can be set by user according to his/her demands. The second modification of our method consists in introducing a new fitness function f_{err} reflecting the accuracy

$$f_{err} = \sum_{i=1}^{N} (y_i - y'_i)^2. \tag{3}$$

In order to optimize the number of components (f_1), adders (f_2) and delay (f_3), we will obtain four fitness functions to optimize: f_1, f_2, f_3 and f_{err}. We expect many Pareto optimal trade-offs among these four objectives, but solutions will always stay within some acceptable accuracy level defined by E.

The situation with the constraint defined by E and the newly added objective function f_{err} is depicted in Fig. 1. The horizontal axis shows the values of f_{err}, the vertical axis gives the values of some objective function f_i (delay, etc.). The Pareto optimal solutions are marked by black dots. It is possible to see decreasing values of function f_i, which has to be minimized. But on the other hand, f_{err}, which describes the difference against desired output values, is growing. At some threshold, when f_{err} is high, we consider the solution unacceptable. This threshold is defined by constant E. In our method, we have utilized the NSGA-II algorithm with controlled elitism [10] to achieve this goal.

IV. Experimental results

In order to evaluate the proposed method, six different MCMs containing 3, 5, 8, 10, 15 and 20 constants were chosen (Tab. I). The objective is to minimize the number of components, the number of adders/subtractors and delay. The number of adders/subtractors means the number of active

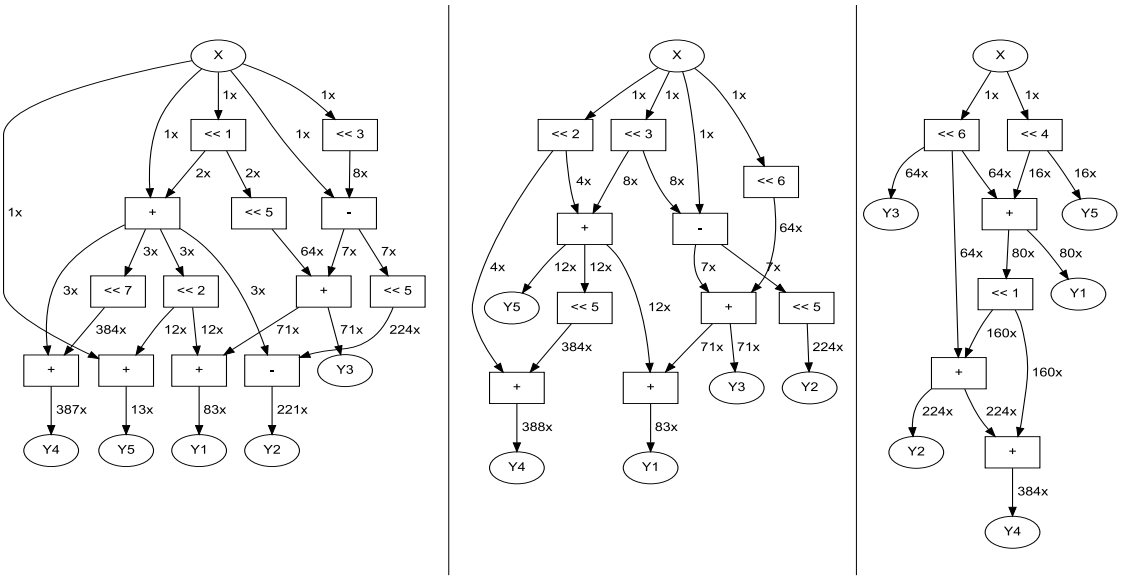

Fig. 2. Three different implementations of MCM with constants (83, 221, 71, 387, 13). Exact solution (left), $f_{err} = 13$ (middle) and $f_{err} = 85$ (right).

nodes of CGP, which are used as adders or subtractors. The number of components is the number of adders/subtractors and shifters together. Delay is the longest path between the input and output computed as the number of nodes along this path.

For each MCM, we performed 20 independent runs of CGP (12×10 nodes utilized). The values of other CGP parameters are: $\lambda = 100$; the number of generations 20.10^6; $\alpha = 7$, and $E = 250$. This CGP setting was determined on the basis of our previous experience [11], [6]. The set of functions consists of adder, subtractor and 1 – 15-bit left shift operations. Table I gives the average time to perform one million generations of CGP on an Intel Xeon 5355 CPU running at 2.66 GHz.

Fig. 2 shows three different implementations of the MCM (5 constants) obtained from a single run of the algorithm. The exact solution ($f_{err} = 0$) is depicted on the left-hand side. It contains the highest number of computational nodes (13). On the right-hand side there is depicted the MCM with $f_{err} = 85$, which utilizes only 6 computational nodes. A compromise implementation between these two solutions is shown in the middle of Fig. 2.

TABLE I. SETS OF MCM COEFFICIENTS USED FOR EVALUATION OF THE PROPOSED METHOD.

N (the number of constants)	Values of constants	CGP time (1 million generations)
3	2925, 23111, 13781	27 m 28 s
5	83, 221, 71, 387, 13	27 m 24 s
8	5, 31, 105, 107, 541, 611, 721, 123314	27 m 35 s
10	117, 1123, 743, 221, 1069, 7605, 987, 16689, 3033, 29	26 m 50 s
15	3, 8, 33, 104, 109, 511, 621, 831, 1001, 1031, 2000, 2002, 2411, 123314, 124211	26 m 47 s
20	1, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71	29 m 1 s

Fig. 3 shows Pareto fronts obtained. One can see that with increasing f_{err} other fitness values are decreasing. The results of our method were compared with results reported in papers [2], [11] which do not consider the approximate design scenario. The best results for the MCM with 3 constants are given in Table II. Implementations showing various trade-offs can easily be identified. Fig. 4 shows the progress of optimization of the 8-constant MCM. Average fitness values calculated from the best individual of each generation and each run are reported.

TABLE II. TRADE-OFFS OBTAINED FOR THE MCM WITH COEFFICIENTS 2925, 23111 AND 13781.

f_{err}	Number of operations	Adders/Subtractors	Delay
0 by [1]	16	8	8
0 by [2]	14	8	7
1	15	8	6
1	16	8	5
1	14	8	7
1	15	9	5
2	15	8	5
2	13	7	6
5	12	6	6
5	11	6	7
5	14	7	5
9	19	11	4
10	17	9	4
11	16	8	4
19	13	6	5
35	12	6	5
35	15	8	4
83	12	5	7
179	14	7	4

V. CONCLUSIONS

A new method for design of approximate MCMs was proposed. In this method, the requirement on functional equivalence between the specification and implementation is relaxed in order to reduce the number of adders/subtractors and delay. Initial experiments have confirmed that approximate MCMs can be evolved. In our future work we will evaluate the method using more complex MCMs. Another goal is to reduce the computational time.

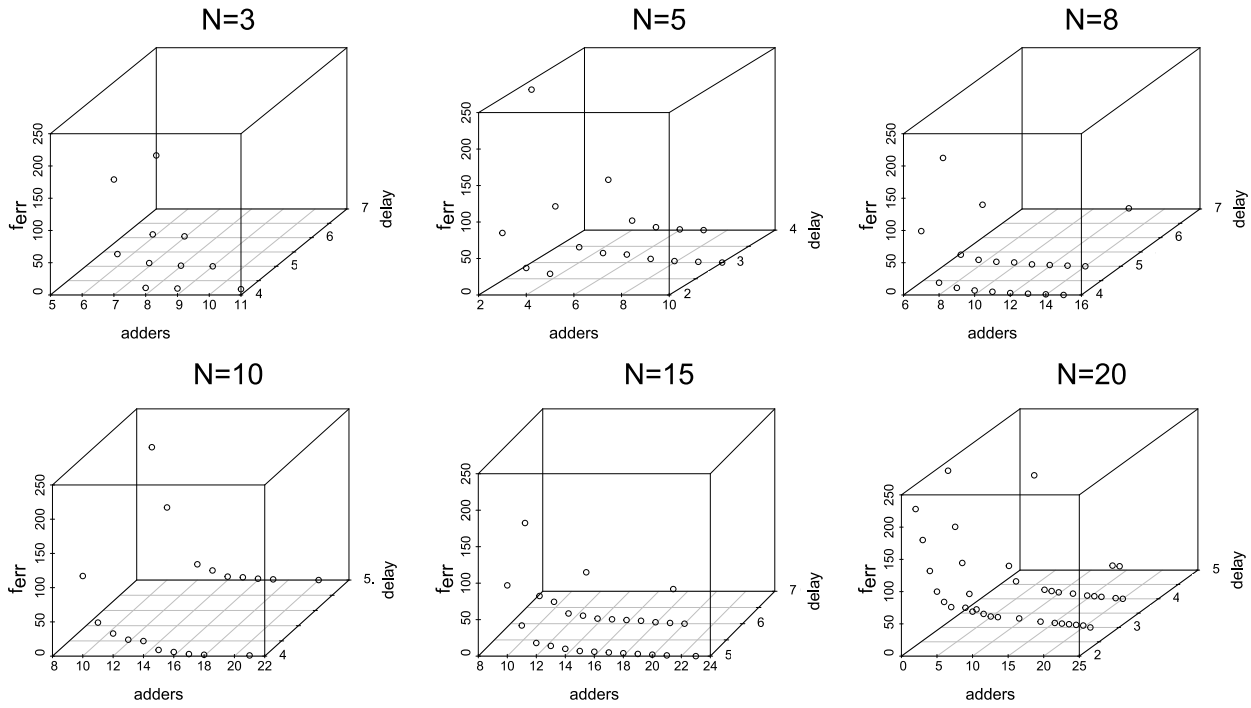

Fig. 3. The best obtained trade-offs for MCMs from Table I.

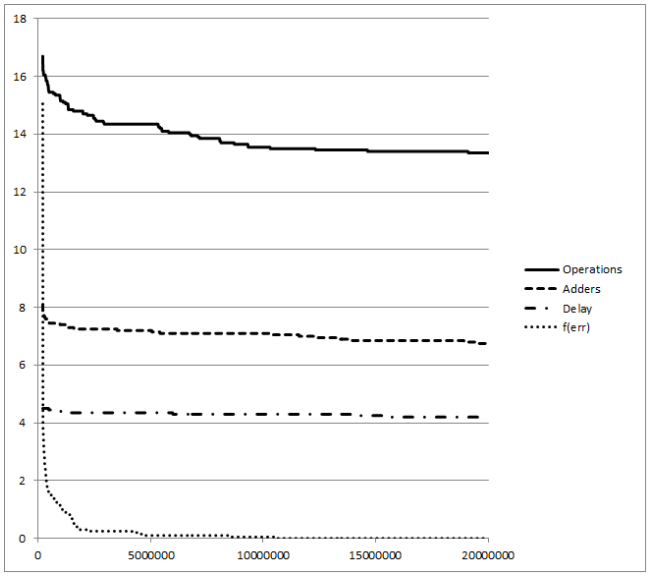

Fig. 4. The progress of evolution for the MCM with 8 coefficients. Best results averaged from 20 independent runs are shown.

ACKNOWLEDGMENT

This work was supported by IT4Innovations Centre of Excellence CZ.1.05/1.1.00/02.0070, the Czech Science Foundation project P103/10/1517 and the research program MSM 0021630528.

REFERENCES

[1] Voronenko, Y. and Püschel, M., Multiplierless Multiple Constant Multiplication, ACM Transactions on Algorithms, vol. 3, no. 2, 2007, pp. 1–28

[2] Vasicek, Z., Zadnik, M., Sekanina, L. and Tobola, J., On Evolutionary Synthesis of Linear Transforms in FPGA, In Proc. of the Conference on Evolvable Systems: From Biology to Hardware, LNCS 5216, Springer Verlag, 2008, pp. 141–152

[3] Kulkarni, P., Gupta, P., Ercegovac, M., Trading Accuracy for Power with an Underdesigned Multiplier Architecture, VLSI Design, 24th International Conference on VLSI Design, 2011, pp. 346–351

[4] Venkatarami, S., Sabne, A., Kozhikkottu, J., Kaushik, R., Raghunatan, A., SALSA: systematic logic synthesis of approximate circuits, The 49th Annual Design Automation Conference 2012, DAC'12, ACM, 2012, pp. 796–801

[5] Miller, J., F. and Thomson, P., Cartesian Genetic Programming, In Proceedings of the Third European Conference on Genetic Programming (EuroGP2000). LNCS 1802, Springer Verlag, 2000, pp. 121–132

[6] Miller, J., F., et. al., Cartesian Genetic Programming. Springer, Heidelberg, 2011

[7] Deb, K., Multi-objective Optimization Using Evolutionary Algorithms. Chichester, UK: Wiley, 2003

[8] Deb, K., Agrawal, S., Pratap, A. and Meyarivan, T., A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Trans. Evolut. Comput. 6 (2), pp. 182–197, 2002

[9] Hilder, J., Walker, J., A., Tyrrell, A., Use of Multi-Objective Fitness Function to Improve Cartesian Genetic Programming Circuits, NASA/ESA Conference on Adaptive Hardware and Systems, 2010, pp. 179–185

[10] Deb, K. and Goel, T., Controlled Elitist Non-dominated Sorting Genetic Algorithms for Better Convergence, In Proc. of the Evolutionary Multi-Criterion Optimization, LNCS 1993, Springer, 2001, pp. 67–81

[11] Petrlik, J., Sekanina, L., Multiobjective evolution of multiple constant multipliers, In Proc. of the 18th Int. Conference on Soft Computing Mendel, Brno, 2012, pp. 64–69