Evolutionary Design of Approximate Sequential Circuits at RTL Using Particle Swarm Optimization

Rebiha Kemcha[1,2(✉)], Nadia Nedjah[3], Amin Riad Maouche[2], and Maamar Bougherara[1,4]

[1] Ecole Normale Superieure - Kouba, Algiers, Algeria
rebiha.kemcha@gmail.com
[2] LIMOSE Laboratory, University of Boumerdes, Boumerdes, Algeria
[3] Department of Electronics Engineering and Telecommunications, State University of Rio de Janeiro, Rio de Janeiro, Brazil
[4] LIMPAF Laboratory, University of Bouira, Bouira, Algeria

Abstract. Evolutionary circuit design has the ability to explore a wide part of the design space and can lead to satisfactory circuits without human experience and knowledge. In this work, we use Multi-Objective Particle Swarm Optimization to evolve approximate sequential circuits at Register-Transfer Level. A circuit is represented by a two-dimensional array. We aim to produce functional approximate circuits having good trade-off between accuracy, delay and area. The results show the efficiency of the proposed approach.

Keywords: Evolutionary design · Sequential circuit · Swarm intelligence · Approximate circuit · Integer divider

1 Introduction

Data-rich applications, such as document/image search, image and signal processing, machine learning classifiers, etc., are computationally intensive while achieving their tasks, making their systems more complex and very expensive in terms of resources (delay, power, etc.). Quantum computing, accelerated-based computing and a plenty of other techniques and technologies are used to overcome this problem.

Another promising paradigm, to reduce required resources in such applications, is *Approximate computing*. This paradigm is based on the inherent error resiliency of some applications. An inherent error resilient application tolerates some kinds of errors or loss of certain accuracy in computations, due mostly to the nature of processed data that can be redundant, large and hence noisy. This laxity results from: (i) the limited perceptual ability of human brains and (ii) the "perfect" solution does not exist for some applications.

In this context, approximate circuits represent an attractive solution to replace accurate circuits. They can give the opportunity for such applications

© Springer Nature Switzerland AG 2019
S. Misra et al. (Eds.): ICCSA 2019, LNCS 11620, pp. 671–684, 2019.
https://doi.org/10.1007/978-3-030-24296-1_54

to take full advantage of their error resiliency by relaxing the "correctness" of outputs and trading it against improvements in energy efficiency, area, speed or other parameters [16].

On the other hand, *Evolutionary Circuit Design* is the use of evolutionary algorithms (EA) to evolve electronic circuits instead of conventional design methods such as Karnaugh map or Quine-McCluskey method. The problem of these traditional methods is that they could not lead efficiently to satisfactory results, especially, in case of complex and large circuits. Furthermore, they depend largely on human knowledge and experience. In contrast, the evolutionary design can explore a wide portion of the design space; hence, it can propose newer and/or more performant implementations than those generated with conventional methods. This approach allows a potential evolution of circuits which have known behavioral requirement and a partial even unknown specification.

Thus, evolutionary computation techniques are proving to be useful for design approximate circuits. In fact, the latter can be treated as a multi-objective optimization problem with several conflicting objectives.

The aim of this work is to use multi-objective optimization algorithms to design approximate sequential circuits at Register-Transfer Level (RTL). In RTL, sequential circuits are composed from arithmetic elements, memories, interconnections and clocking. It is known that sequential circuits are cheaper in term of area than combinatorial circuits.

An evolutionary design of an approximate sequential integer divider is proposed herein. Integer division is used in many applications like signal processing, communications, etc., and has a huge complexity when implemented in hardware [13] leading to an excessive consumption of resources. An approximation of such circuit is therefore a good compromise between complexity and accuracy. For this purpose, we use Multi-Objective Particle Swarm Optimization as a method of design by considering error metric, delay and area as objectives to approximate a sequential accurate integer divider.

After the introduction, the paper is organized as follows. Section 2 gives some related researches. Section 3 provides details on MOPSO. Section 4 describes our approach to design approximate sequential circuits as well as the fitness function. In Sect. 5, experiments and results are described. Finally, Sect. 6 concludes the paper.

2 Related Work

Approximate circuits are circuits which meet the needs of error resilient applications while providing a trade-off between accuracy and performance. They constitute an eventual "better" solution than accurate circuits for related domains like: clustering/filtering data, image/signal processing, computer vision algorithms, document search, etc., where it is possible to trade result accuracy against an improvement in power consumption, area, delay or other measures.

Concerning the design of approximate circuits, two techniques are described in this area: over-scaling and functional approximation [17]. In the *over-scaling*

technique, approximations are introduced: (i) by creating timing errors to decrease the energy consumption without changing the clock frequency, called "voltage over-scaling" or (ii) by increasing frequency to gain performance without changing the voltage, called "over-clocking" [17].

In the *functional approximation* technique, the logic function computed by the circuit is modified so that it tolerates acceptable errors. For arithmetic circuits, approximations are introduced into building blocks (e.g. adders, multipliers, etc.) for reducing power consumption and design area [16]. These two techniques can be combined.

Circuit approximations are *manually* or *automatically* performed. Manual re-design is the subject of a large number of research works proposing re-design of common arithmetic building blocks such as adders [20], multipliers [21] and dividers [19]. Jiang et al. in [18] survey the design of a plethory of approximate arithmetic circuits and propose others.

However, manual re-design is not efficient to design large and complex circuits. Hence, automated methods as SASIMI [23], SALSA [22] and ABACUS [24] were proposed. These methods use heuristics to approximate circuits and verification engines to check quality of the resulting circuits [25]. They start with a fully functional circuit then apply an iterative stochastic method to approximate it and evaluate it in terms of accuracy and power and/or area [25].

On the other hand, several works address the *evolutionary circuit design*. They exploit the features of evolutionary computation to propose either novel implementations of circuits or more performant ones. Manovit et al. in [1] use Genetic Algorithms (GA) to evolve sequential circuits at Register-Transfer Level (RTL) from partial input/output sequence. They succeed to evolve 2-inputs serial adder, frequency divider, odd parity and modulo-5 detector.

Soliman and Abbas [2] consider a sequential circuit as a grid and separate the combinatorial part from memories. They split the circuit into several subcircuits and evolve them separately at gate level using GA. The authors propose the evolution of 3-bit up-counter with perfect functionality and minimum of logic gates as objectives.

Liang et al. [3] propose a three-step decomposition method and (μ, λ) Evolution Strategy (ES) to evolve Finite State Machines of large sequential circuits. They test their approach on the MCNC benchmark library.

Tao et al. in [4] evolve auto-defined sequence detector, modulo-6 counter, modulo-16 counter and ISCAS089 "lion" circuit. They compare different variants of Genetic Programming (GP) and GA. According to the authors, CGP provides higher results than the other variants of GP, but their GA-based approach was least complex. In [5], the authors combine their first approach with a Lookup-Table (LUT)-based circuit evolution (LCE) to tackle the problem of the evolutionary design of sequential circuit.

After presenting functional approximation and evolutionary design of circuits, we can see clearly that they are closely related [25]. Evolutionary design aims to produce circuits in response to multiple expectations, which can be translated in conflicting objectives among others. Accuracy, power, delay, area, which

can be considered as objectives in an approximate circuit design, are naturally conflicting. For this reason, several works addressed the *evolutionary design of approximate circuits* [26,27]. All of these works use a CGP-based approach to design and optimize proposed circuits. Vasicek et al. in [26] evolved approximate 4-bit multipliers, 9-median and 25-median circuits at gate level.

Vasicek and Sekanina in [27] propose a multi-objective evolution of various combinatorial circuits from different libraries as LGSynth or ISCAS. In order to evaluate candidate circuits, the authors use a two-stage approach: (i) the first stage is to evaluate the approximation (an error metric based on Hamming Distance), and (ii) the second stage, which begins after obtaining a set of "accepted" circuits, consists of fitness function with three weighted terms (error, area and delay).

As far as we know, there is no work dealing with the design of approximate sequential circuits at RTL. Hence, our aim is to use Multi-Objective Particle Swarm Optimization to produce approximate sequential circuits with accuracy, delay and area as objectives.

3 Multi-objective Particle Swarm Optimization

Particle Swarm Optimization (PSO) is an evolutionary computation approach proposed by Kennedy and Eberhart [11]. In this technique, an individual, called particle, simulates the bird's movement through the space while the swarm mimics the social behavior of the flock in the search of food. Thus, PSO is a population-based optimization algorithm considering each particle position as a potential solution of the problem [8].

Every particle tries to "fly" toward the best performing particle at that moment and therefore it is given a position and a velocity. The particle position $\vec{x}_i$ is a vector of as many coordinates as the dimensions of the search space. Every iteration t, the position is dynamically adjusted according to values of particle parameters, its previous best position (called *local best*) and the swarm's best known position (called *best particle*). The position is updated using the velocity information as described in Eq. 1:

$$\vec{x}_i(t) = \vec{x}_i(t-1) + \vec{v}_i(t) \tag{1}$$

where $\vec{v}_i(t)$ is the velocity. It is computed as described in Eq. 2:

$$\vec{v}_i(t) = w.\vec{v}_i(t-1) + c_1.r_1.(\vec{x}_{pi} - \vec{x}_i) + c_2.r_2.(\vec{x}_{gi} - \vec{x}_i), \tag{2}$$

where, the term $w.\vec{v}_i(t-1)$, which is based on the previous velocity, is the inertia. It is used to control the trade-off between the exploitation and exploration of the search space [8]. c_1 and c_2 are arbitrary positive constants called *cognitive* and *social coefficient* respectively, and r_1 and $r2$ are random variables $\in [0, 1]$ of uniform distribution.

The cognitive term $c_1.r_1.(\vec{x}_{pi} - \vec{x}_i)$ controls the effect of the local best $\vec{x}_{pi}$ on the next position of the particle i. The social term, $c_2.r_2.(\vec{x}_{gi} - \vec{x}_i)$ attracts the particle i to the global best position $\vec{x}_{gi}$ found by the particles of the swarm.

Multi-Objective Particle Swarm Optimization (MOPSO) is the extention of PSO to solve multiobjective problems. The main idea to do this, is to adapt the basic algorithm by adding [9]: (i) an external archive to store non-dominated solutions, (ii) a selection mechanism of leaders and (iii) an archiving method to prune repository of non-dominated solutions once full. The first proposed MOPSO algorithm was developed in [12] by Moore and Chapman in 1999. Algorithm 1 shows the main steps of a basic MOPSO process. Since the first version of MOPSO, a plurality of variants have been proposed. For evolving circuits in this work, we used a variant of MOPSO algorithms namely SMPSO.

Algorithm 1. Pseudocode of a basic Multi-objective PSO

```
Create the initial swarm
Evaluate all the particles of the swarm
Initialize the velocity for each particle
Initialize the particles memory
Initialize the leaders
iteration ← 0
while iteration < max_iteration do
  Compute velocity of particles
  Update position of particles using Eqs 1 and 2
  Mutate particles
  Evaluate particles fitness
  Update Leaders archive
  Update the particles memory
  iteration ← iteration + 1
end while
Return result from the archive of leaders
```

Speed-constrained Multi-objective PSO (SMPSO) is a multiobjective particle swarm optimization algorithm proposed by Nebro et al. in [7]. It is based on Optimized Multi-Objective Particle Sswarm Optimizer (OMOPSO) [14] features, which include: (i) the use of the crowding distance to filter out the list of leaders (ii) the combination of two mutation operators acting on different subsets of the swarm to accelerate the convergence and (ii) the use of a clustering technique, which is ϵ-dominance to limit the number of solutions in the external archive.

SMPSO incorporates a mechanism for *velocity constriction* and introduces a polynomial mutation operator for $\frac{1}{6}$ of the swarm.

The authors of SMPSO propose to control the accumulated velocity of each variable j in each particle i with *velocity constriction* as described in Eq. 3:

$$v_{i,j}(t) = \begin{cases} delta_j & \text{if } v_{i,j}(t) > delta_j, \\ -delta_j & \text{if } v_{i,j}(t) \leq -delta_j, \\ v_{i,j}(t) & \text{otherwise.} \end{cases} \tag{3}$$

where

$$delta_j = \frac{upperLimit_j - lowerLimit_j}{2} \tag{4}$$

where $upperLimit_j$ and $lowerLimit_j$ are limits of the variable j of the particle position in the search space. These limits prevent the swarm explosion when the velocity of particles become too high. Nebro et al. used the constriction coefficient χ defined firstly by Clerc and Kennedy in [10], based on the values of cognitive and social coefficients c_1 and c_2.

4 Proposed Approach

In RTL, a sequential circuit contains arithmetic components such as adders, multipliers, etc., storage elements such as registers, flip flops, etc. and interconnection elements such as buses, multiplexers, etc. A circuit is controlled by a system clock.

4.1 Circuit Representation

A circuit is represented by a matrix $\mathcal{M}$ of m rows and n columns. It has α primary inputs and β primary outputs. The matrix is composed of cells. A cell represents a component, a list of inputs and a list of outputs. A component can be selected from a list of X components enumerated in Table 1.

Table 1. List of components

Component	Code	#Inputs	#Bits	Area (GE)	Time (ns)
Adder	1	2	16	384	345
Subtractor	2	2	16	384	345
Multiplier	3	2	16	384	380
Register	4	1	16	82	15
Counter	5	0	16	82	2.11
Multiplexer	6	3	16	113	5.25
Equal-to	7	2	16	104	31.5
Greater-than	8	2	16	252	40
Less-than	9	2	16	252	40
And	10	1	1	2	3.5
Or	11	1	1	3	4.7
Not	12	1	1	1	3.8
Buffer	13	1	1	2	3.5

A cell of a column c, where $0 < c \leq n - 1$, draws its inputs from outputs of λ previous columns and outputs of μ next columns except those of the column

which contains the cell in question. In addition, inputs of a cell in the first column, $c = 0$, may be the primary inputs. The outputs of circuit cells are labeled from $\alpha + 1, \alpha + 2, ...$ to $\alpha + \delta$, where:

$$\delta = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \delta_{i,j}, \tag{5}$$

where $\delta_{i,j}$ is the number of outputs of a cell in the ith row and the jth column. Figure 1 shows an example of a sequential circuit and its corresponding representation. Clock is assumed for registers and counters.

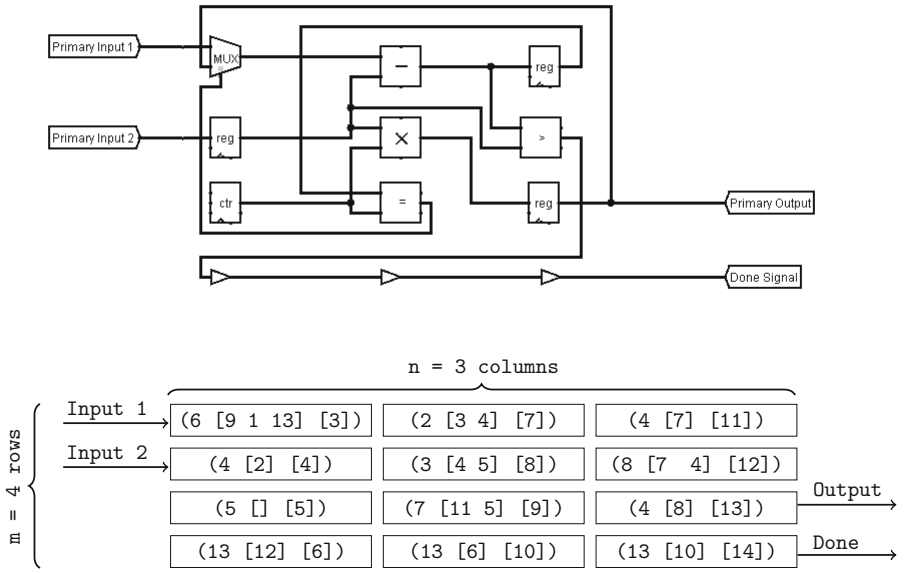


Fig. 1. Circuit representation.

In a circuit designed in this way, the component of each column perform its operation in parallel manner with other column components. Each column is a stage of operations and is executed sequentially with other columns.

4.2 Objectives

The fitness function takes into account three objectives of the evolved circuit $\mathcal{M}$ as defined in Eq. 6.

$$F(\mathcal{M}) = \begin{bmatrix} Error(\mathcal{M}) \\ Area(\mathcal{M}) \\ Time(\mathcal{M}) \end{bmatrix}, \tag{6}$$

where $Error(\mathcal{M})$ is the error metric measuring the results accuracy of the circuit, $Time(\mathcal{M})$ is the execution time and finally $Area(\mathcal{M})$ is the components area.

Accuracy. The Error metric of an approximate circuit is estimated by comparing the resulting output of the evolved circuit with the outputs produced by an exact solution. To evaluate this objective, we use an universal function proposed by De Oliveira et al. [15]. This function is defined as follows:

$$Error(\mathcal{M}) = log\left(1 + \frac{T_1 + T_2 + T_3}{\rho^2}\right) \tag{7}$$

where term T_1 is the sum of squared errors, which is able to indicate similarities between the measured value $\widehat{y}$ and the expected value y. N is the length of the sequence of inputs/outputs. T_1 is described as follows:

$$T_1 = \sum_{i=1}^{N} (\widehat{y} - y)^2. \tag{8}$$

Term T_2 is the sum of squared Fourier Transform magnitudes of measured and expected values. This term gives extent of the difference between a result computed by the evolved circuit and the one given by the target circuit. Equation 9 shows the formula of T_2.

$$T_2 = \sum_{i=1}^{N} (|\mathcal{F}(\widehat{y}) - \mathcal{F}(y)|)^2, \tag{9}$$

Term T_3 is the sum of squared Fourier Transform phases of measured and expected values. The phase indicates how many results are different from the evolved circuit to the target one. It is computed as follows:

$$T_3 = \sum_{i=1}^{N} \left(\underline{/\mathcal{F}(\widehat{y})} - \underline{/\mathcal{F}(y)}\right)^2, \tag{10}$$

The last term is ρ^2 which is the square of the Correlation of Pearson ρ. The term $\frac{1}{\rho^2}$ tends to infinity if the measured result $\widehat{y}$ and the expected result y are unrelated and it is close to 1 if $\widehat{y}$ and y are positively or negatively correlated.

$$\rho^2 = \left(\frac{Cov(\widehat{y}, y)}{\sqrt{Var(\widehat{y}).Var(y)}}\right)^2 \tag{11}$$

Area. The total area of the circuit $Area(\mathcal{M})$ (Eq. 12) is an estimation of the area occupied by all involved components $x \in \mathcal{M}$. For simplicity, we do not compute the area of buses and other interconnections except multiplexers. The area is measured using Gate Equivalent (GE). GE represents an estimation of a component area based upon the number of interconnected logic gates used to implement the same functionality of the related component [6]. The area $Area_x$ of each component x is estimated using 2-inputs NAND gate.

$$Area(\mathcal{M}) = \sum_{x \in \mathcal{M}} Area_x. \tag{12}$$

Delay. The performance is, herein, considered as delay taken by the circuit to deliver the first intermediate result. All operations in the same matrix column j are executed in parallel manner. In addition, a column is a stage of operations. t_j is the largest component time among all components of this column. The total time $Time(\mathcal{M})$ is the sum of columns times.

$$Time(\mathcal{M}) = \sum_{j=0}^{n-1} t_j, \tag{13}$$

where t_j is the maximum time of column j.

4.3 Search Methodology

The initial population is seeded with an accurate circuit using a single-objective PSO algorithm (SPSO2011). The tolerated error is a parameter defined by the user as well as the size of the seed. This population is then taken by the multi-objective particle swarm optimization algorithm to start the evolutionary approximation of the accurate circuit. The outline of our approach's steps is as follows:

1. Seed the population with accurate circuit to produce the initial population
2. Initialize all cells with a component x picked from the list of components
3. Update outputs of all cells
4. Interconnect components between them
5. Designate the primary outputs and enable signal of the output register
6. Evaluate the fitness of the circuit
7. Apply the multi-objective optimization algorithm

5 Experiments and Results

This section presents the results of experiments to show the capability of particle swarm optimization to evolve approximate circuits.

In this work, we used jMetal framework [28], which contains several evolutionary algorithms. SMPSO and SPSO2011 are adapted to deal with our problem configuration. Several simulations were performed to find out how to set up the required parameters. For SPSO2011, we kept same setting of parameters implemented in jMetal: $w = 0.721$, $c = 1.193$, $\#iter = 10000$, $swarm\ size = 100$ and number of particles to inform is set to 10. For SMPSO, c_1 and c_2 coefficients are randomly generated from $[1.5, 2.5]$ and weight w is also randomly generated from $[0.1, 0.5]$.

We consider evolving a sequential integer divider shown in Fig. 2a. The accurate circuit has 2 integer inputs (dividend and divisor), 1 integer output (quotient), 3 rows and 4 columns of components. The last column of circuits contains registers to store intermediate results of division. Clock is assumed for components like registers and counters. In this circuit, sequential integer division is

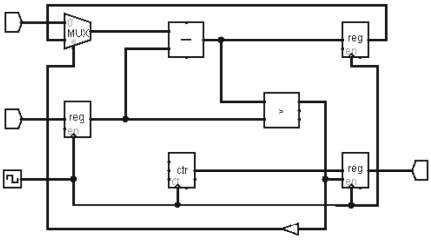
performed by successive subtractions of dividend and divisor (for the first clock cycle) or intermediate remainder (for the rest). This operation takes a number of cycles equals to the value of quotient. We evaluate the area of the circuit by summing all components area. The delay is estimated as the maximum time taken to deliver the first intermediate result. The number of clock cycles depends on the operands' values.

We inject the accurate circuit as a gravity center in SPSO2011 algorithm. This technique allows to define the neighborhood of the accurate circuit leading to the generation of some different versions of the concerned circuit.

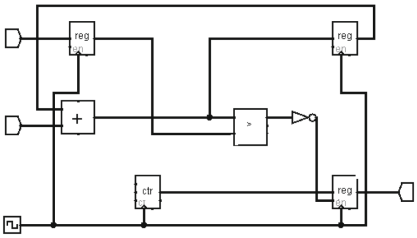
The representation of an approximate circuits produced by SPSO2011-SMPSO is shown in Fig. 2b. Our algorithm found this solution after 5859 iterations. This circuit has an error of 5.66, which means that it is ±1 from the

Table 2. Accurate and approximate divider performances

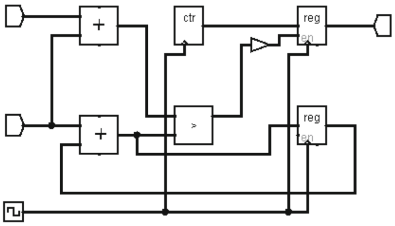
	Accurate	Approximate	First circuit found	Best area and time
Error	0	±1	±1	±3
Area	1083	972	970	889
Time	415	400	438.5	395.5



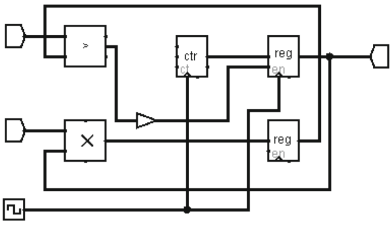
(a) Accurate (E=0, A=1083, T=415)



(b) Approximate (E=+1, A=972, T=400)



(c) Approximate (E=+2, A=1191, T=400)



(d) Approximate (E=+3, A=889, T=395.5)

Fig. 2. Accurate and evolved approximate integer dividers (E = error, A = area, T = time)

accurate result. The first approximation presented in Fig. 2c, was found after 461 iterations, and in terms of time and area, the best circuit shown in Fig. 2d, was found after 14069 iterations. Table 2 presents performances of these circuits.

In matrix representation, the accurate circuit has 3 rows and 4 columns. For its part, the first approximate circuit has 3 rows and only 3 columns, which led to a reduction of area and time. In fact, the time is reduced by 4.7%, but the largest improvement is that of area, which is reduced by 22.5%.

It has been noted that circuits having smaller area than the accurate divider have fewer columns, that is, the components of some columns are not connected to the rest of circuit components. The cells of such components are not involved in the computation of the results and thus they are not taken into account in the performances of the overall circuit.

Table 2 shows the performances of the accurate circuit, an approximate one, first circuit found and best circuit in terms of area and time. The second circuit (approximate) has better area and time than the accurate circuit. The reduction of the area is about 10.2% and that of time is 3.75%.

It has been observed that the error metric $Error(\mathcal{M})$ gives 5.66 for circuits with ± 1 from accurate results for a 32-input/output sequence. Each circuit is run 32 times to form the results sequence which is compared with the accurate results in order to define the circuit error metric. In addition, it is noteworthy to mention that circuits with an error metric less than or equal to 7.80 are valid which means that they are sequential and give harmonious results. The value of the error metric depends, obviously, on the results sequence but also largely on the length of the input/output sequence. The circuit that has an error metric of 5.66 gave 4.23 with a 8-input/output sequence. The fitness function has to be normalized to give values independently from the length of test sequences.

The resulting Pareto front is presented in Fig. 3. During the 25000 iterations, only 37 circuits were sequential, which 56.75% circuits gave results with ± 4 from the accurate result and the rest were invalid. Note that all circuits of the Pareto front have an error metric less or equal than ± 3. Circuits with a ± 4 were expensive in terms of area and time comparing to the accurate circuit (e.g., $Area = 1273\ GE$ and $Time = 437.11$ ns).

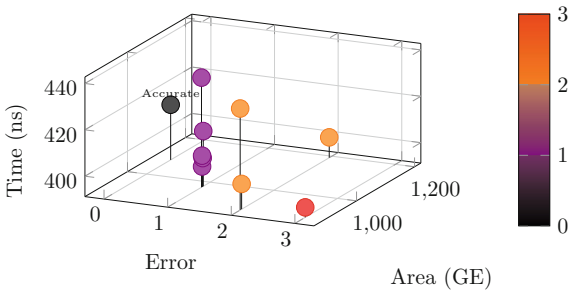


Fig. 3. Pareto Front

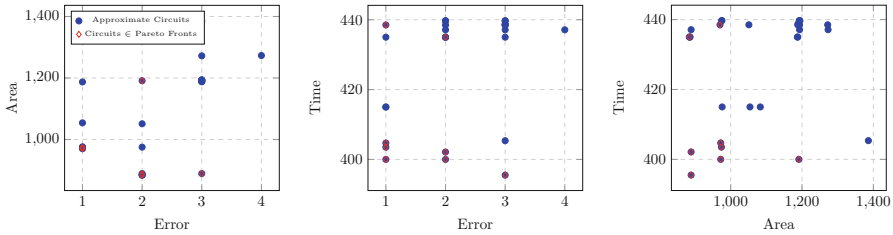


Fig. 4. Performances of 21 valid circuits

Figure 4 presents the performances of some obtained valid sequential circuits considering only two objectives at the same time. Circuits of the pareto front are also plotted. According to the middle plot, more than half of the circuits have more than 420 ns which means that they are worse than the accurate circuit in term of time. Otherwise, according to the plot on the right of Fig. 4, over half of the circuits have less than 1080 *GE*, i.e., they are better than the accurate circuit in term of area.

6 Conclusion

The functional approximation of arithmetic circuits presents a solution for error resilient applications by providing good compromises between accuracy and performance. This work aims to explore the use of MOPSO to design sequential approximate integer dividers at Register-Transfer Level. Two algorithms were used for this purpose, namely SPSO2011 and SMPSO. We evaluate error metric, area and time for each evolved circuit. The obtained results show that an improvment of 22.5% was achieved in term of area. The proposed method shows the possibility to use particle swarm in the design of approximate sequential circuits. In our future research, we will work on producing more results and increasing the scalability of our method and its extension to design accurate sequential circuits.

References

1. Manovit, C., Aporntewan, C., Chongstitvatana, P.: Synthesis of synchronous sequential logic circuits from partial input/output sequences. In: Sipper, M., Mange, D., Pérez-Uribe, A. (eds.) ICES 1998. LNCS, vol. 1478, pp. 98–105. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0057611
2. Soliman, A., Abbas, H.: Synchronous sequential circuits design using evolutionary algorithms. In: Canadian Conference on Electrical and Computer Engineering 2004 (IEEE Cat. No. 04CH37513), pp. 2013–2016. IEEE (2004)
3. Liang, H., Luo, W., Wang, X.: A three-step decomposition method for the evolutionary design of sequential logic circuits. Genet. Program. Evolvable Mach. **10**(3), 231–262 (2009)

4. Tao, Y., Cao, J., Zhang, Y., Lin, J., Li, M.: Using module-level evolvable hardware approach in design of sequential logic circuits. In: 2012 IEEE Congress on Evolutionary Computation, pp. 1–8. IEEE (2012)
5. Tao, Y., Zhang, Y.: An extrinsic EHW system for the evolutionary optimization and design of sequential circuit. In: Proceedings of the 2018 Artificial Intelligence and Cloud Computing Conference, pp. 174–180. ACM (2018)
6. Nedjah, N., Mourelle, L.D.M.: Encodings and genetic operators for efficient evolutionary design of digital circuits. Int. J. Bio-Inspired Comput. **9**, 197–210 (2017)
7. Nebro, A., Durillo, J., Garcıa-Nieto, J., Coello, C.A.C., Luna, F., Alba, E.: SMPSO: a new PSO metaheuristic for multi-objective optimization. In: Proceedings of the 2009 IEEE Symposium on Computational Intelligence in Multi-criteria Decision-making, pp. 66–73 (2009)
8. Nedjah, N., Mourelle, L.d.M.: Evolutionary multi-objective optimisation: a survey. Int. J. Bio-Inspired Comput. **7**(1), 1–25 (2015)
9. Strickler, A., Pozo, A.: Evolving connection weights of artificial neural network using a multi-objective approach with application to class prediction. In: Nedjah, N., Lopes, H.S., Mourelle, L.M. (eds.) Designing with Computational Intelligence. SCI, vol. 664, pp. 177–197. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-44735-3_10
10. Clerc, M., Kennedy, J.: The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE Trans. Evol. Comput. **6**(1), 58–73 (2002)
11. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948, November 1995
12. Moore, J., Chapman, R.: Application of particle swarm to multiobjective optimization, Technical report (1999)
13. Moller, N., Granlund, T.: Improved division by invariant integers. IEEE Trans. Comput. **60**(2), 165–175 (2011)
14. Sierra, M.R., Coello Coello, C.A.: Improving PSO-based multi-objective optimization using crowding, mutation and ε-dominance. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 505–519. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31880-4_35
15. De Oliveira, P.J.A., Nedjah, N., Mourelle, L.D.M.: Uma Heurística Geral Para a Comparação de Sinais. In: Proceedings of XXXVIII Ibero-Latin American Congress on Computational Methods in Engineering (CILAMCE), Florianópolis, SC, Brazil (2017)
16. Venkatesan, R., Agarwal, A., Roy, K., Raghunathan, A.: MACACO: modeling and analysis of circuits for approximate computing. In: 2011 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 667–673. IEEE (2011)
17. Hashemi, S., Reda, S.: Approximate multipliers and dividers using dynamic bit selection. In: Reda, S., Shafique, M. (eds.) Approximate Circuits, pp. 25–44. Springer, Cham (2019). https://doi.org/10.1007/978-3-319-99322-5_2
18. Jiang, H., Liu, L., Lombardi, F., Han, J.: Approximate arithmetic circuits: design and evaluation. In: Reda, S., Shafique, M. (eds.) Approximate Circuits, pp. 67–98. Springer, Cham (2019). https://doi.org/10.1007/978-3-319-99322-5_4
19. Chen, L., Lombardi, F., Montuschi, P., Han, J., Liu, W.: Design of approximate high-radix dividers by inexact binary signed-digit addition. In: Proceedings of the on Great Lakes Symposium on VLSI 2017, pp. 293–298. ACM (2017)
20. Kahng, A.B., Kang, S.: Accuracy-configurable adder for approximate arithmetic designs. In: Proceedings of the 49th Annual Design Automation Conference, pp. 820–825. ACM (2012)

21. Lin, C.-H., Lin, C.: High accuracy approximate multiplier with error correction. In: 2013 IEEE 31st International Conference on Computer Design (ICCD), pp. 33–38. IEEE (2013)
22. Venkataramani, S., Sabne, A., Kozhikkottu, V., Roy, K., Raghunathan, A.: SALSA: systematic logic synthesis of approximate circuits. In: DAC Design Automation Conference 2012, pp. 796–801. IEEE (2012)
23. Venkataramani, S., Roy, K., Raghunathan, A.: Substitute-and-simplify: a unified design paradigm for approximate and quality configurable circuits. In: 2013 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1367–1372. IEEE (2013)
24. Nepal, K., Li, Y., Bahar, R.I., Reda, S.: ABACUS: a technique for automated behavioral synthesis of approximate computing circuits. In: 2014 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1–6. IEEE (2014)
25. Sekanina, L., Vasicek, Z.: Evolutionary computing in approximate circuit design and optimization. In: 1st Workshop on Approximate Computing (WAPCO 2015), pp. 1–6 (2015)
26. Vasicek, Z., Sekanina, L.: Evolutionary design of approximate multipliers under different error metrics. In: 17th International Symposium on Design and Diagnostics of Electronic Circuits & Systems, pp. 135–140. IEEE (2014)
27. Vasicek, Z., Sekanina, L.: Evolutionary design of complex approximate combinational circuits. Genet. Program. Evolvable Mach. **17**(2), 169–192 (2016)
28. Durillo, J.J., Nebro, A.J., Alba, E.: The jMetal framework for multi-objective optimization: design and architecture. In: IEEE Congress on Evolutionary Computation, pp. 1–8. IEEE (2010)