Designing Combinational Circuits Using a Multi-objective Cartesian Genetic Programming with Adaptive Population Size

Leandro S. Lima[1], Heder S. Bernardino[1(✉)], and Helio J. C. Barbosa[1,2]

[1] Univerisdade Federal de Juiz de Fora (UFJF), Juiz de Fora, MG, Brazil
leandro.lima@engenharia.ufjf.br, heder@ice.ufjf.br, hcbm@lncc.br
[2] Laboratório Nacional de Computação Científica (LNCC), Petrópolis, RJ, Brazil

Abstract. This paper proposes a multiobjective Cartesian Genetic Programming with an adaptive population size to design approximate digital circuits via evolutionary algorithms, analyzing the trade-off between the most often used objectives: error, area, power dissipation, and delay. Combinational digital circuits such as adders, multipliers, and arithmetic logic units (ALUs) with up to 16 inputs and 370 logic gates are considered in the computational experiments. The proposed method was able to produce approximate circuits with good operational characteristics when compared with other methods from the literature.

Keywords: Combinational circuits · Multi-objective optimization · Cartesian Genetic Programming

1 Introduction

The design of electronic circuits is usually a complex task which requires knowledge of specific methodologies. The use of evolutionary algorithms (EAs) to design digital systems gave rise to the Evolvable Hardware (EH) field [9]. The reasons which brought EH to the spotlight of hardware development are its ability to [9]: (i) reach novel architectures which the conventional methods would hardly provide due to their non-flexible nature, (ii) find fair solutions for problems where the specifications are incomplete, (iii) deliver fair solutions in scenarios where there is not a perfect solution, (iv) achieve fault tolerance at the hardware level, and (v) to make the design less dependent on the expert.

EH has provided good results as in [4,7,9], but EH faces issues such as the representation of solutions, as long chromosomes are usually required to encode complex circuits which lead to large search spaces and, consequently, increased

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-37599-7_49) contains supplementary material, which is available to authorized users.

G. Nicosia et al. (Eds.): LOD 2019, LNCS 11943, pp. 592–604, 2019.
https://doi.org/10.1007/978-3-030-37599-7_49

difficulty in solving the problem. In addition, in the design of combinational digital circuits (CDCs), the processing time grows exponentially with the number of circuit inputs, and EH also uses techniques which are very time consuming [7].

Addressing the design of CDCs, energy efficiency, complexity and delay are features of digital circuits to be analyzed during the manufacturing process once people desire faster, simpler and energetically efficient devices. Approximate Computing (AC), a new paradigm in electronic projects, explores systems which could tolerate loss (i.e. precision) in order to reduce complexity, costs, delay and to increase the energy efficiency of the systems [4,5]. AC can be found in inherently error resilient situations [10], such as multimedia [5], machine learning [11], approximate arithmetic circuits [4], and FIR and IIR approximate filters [9].

The usage of AC and EH in the context of evolutionary design has gained attention as these approaches may lead to the conception of circuit architectures which are different and might be superior to the designs created by specialists [9]. In this scenario, digital circuits obtained via AC are classified as *approximate digital circuits* (ADCs) [10]. Their requirements are relaxed aiming at achieving savings in energy consumption, delay, and complexity. As it may be necessary to increase the complexity of a target circuit to reduce errors, energy efficiency and delay would probably be degraded as these quantities are conflicting. Thus, a multiobjective optimization problem arises [4]. A variety of trade-offs between error, delay, and energy efficiency can be found by a multiobjective approach, enabling the design of a vast number of ADCs. Applying the evolutionary approach in the scenario of intrinsic evolutionary design (e.i. in which the evolutionary process is conducted in the target device) could bring new possibilities as the final solution is already implemented in hardware.

Cartesian Genetic Programming (CGP) [7] is a genetic programming method in which programs are expressed as directed acyclic graphs (DAGs) with their nodes organized in a matrix [7]. CGP allows for a convenient representation when several inputs and outputs are required and, consequently, has become the most popular method in the evolutionary design of CDCs [7].

Here, we propose a technique to design approximate combinational digital circuits (ACDC) based on CGP where the size of the population varies according to the number of non-dominated solutions. One candidate solution starts the search process, the population is allowed to grow up to a maximum number of non-dominated solutions, and, when the population size exceeds a predefined threshold, the candidate solutions with the lowest crowding distances [3] are eliminated. We study here the trade-off between the delay, output error, and power dissipation when designing approximate digital arithmetic circuits. In particular, 8-bit adder (A8) and 8-bit multiplier (M8) are commonly used in the context of AC and EH [4]. Thus, we included them in computational experiments. In addition, the Arithmetic Logic Unit (ALU) is considered here. The results found are compared to those obtained by other methods from the literature.

2 Related Work

Vasicek et al. [10] developed ADCs in which the requirements on functional equivalences between the specifications and implementations were relaxed leading to gains in the speed of computation, area occupied on a chip, and energy consumption. That approach can also be used in multimedia and image compression applications. For instance, most users would not notice variations in the brightness degree in some pixels of an image [5].

Hrbacek et al. [4] proposed a multiobjective approach to design ADCs using CGP to represent candidate circuits, and the Non-Dominated Sorting Genetic Algorithm II (NSGA-II) [3] to explore the search space by analyzing the trade-off between error, delay and power dissipation. The initial population was composed of fully functional circuits, instead of random circuits, and approximate versions of M8 and S8 were designed with significant power consumption savings.

Kaufmann et al. [6] proposed a local search algorithm called hybrid evolutionary strategy (hES) based on the evolutionary strategy and the concepts of Pareto dominance. The hES method uses the Fast non-dominated sort and Crowding-distance of NSGA-II [3], and alternates its evolutionary process using global and local search. That method and its periodization with NSGA-II was compared with Strength Pareto Evolutionary Algorithm 2 (SPEA2) and NSGA-II when designing 2-bit multiplier and adder. CGP's representation was adopted, the functional quality of the solutions was treated as a constraint, and circuit area and delay were the objectives. The authors concluded that for the evolution of digital circuits which uses CGP as a representation model, hES and its periodization are significantly better than NSGA-II and SPEA2.

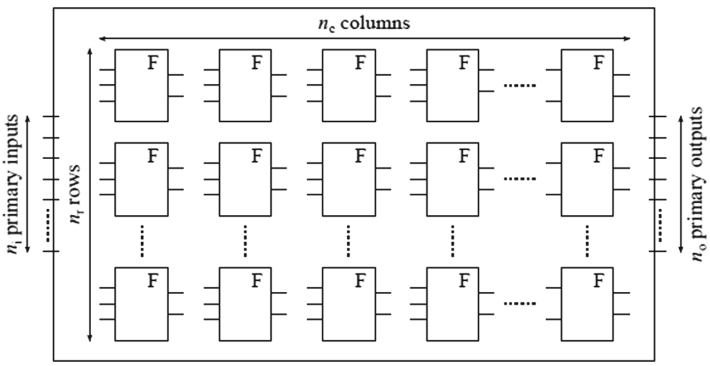

Fig. 1. Illustration of the representation used in CGP (extracted from [4]).

3 Cartesian Genetic Programming

Cartesian Genetic Programming (CGP) [7], is an evolutionary technique where a circuit is represented by a grid of $n_r \times n_c$ interconnected nodes, as in Fig. 1. The

processing elements, represented by F, can be either elementary gates (e.g. AND, OR, XOR) or functional level components (e.g. adders, comparators, shifters, and multipliers). Nodes inputs can be connected either to one of the n_i primary inputs or to a node located in the previous l columns; l is defined as "levels-back", which controls the connectivity of the graphs. Each node has a fixed number of inputs, N_{ni}, and outputs, N_{no}. The nodes are able to perform one of the functions from the set Γ ($F \in \Gamma$). Each one of the n_o primary outputs can be connected either to a primary input or to an output of a node. A candidate circuit represented by CGP is described by a sequence of integers which represents the functions and the connections between the nodes. The encoding of a chromosome consists of $n_r \times n_c$ triplets (i_1, i_2, F). Thus, for the case considered here where the gates have two inputs, it is possible to represent a node with its input indices i_1 and i_2 (other nodes), and a function ψ. The last part of GP's encoding contains n_o integers which specify the primary output nodes of the circuit.

In the standard CGP, the population consists of a fixed number of individuals, commonly, randomly initialized. Also, some previous knowledge of the problem can be used. The optimization process when designing circuits is normally an evolutionary strategy $(1+\lambda)$-ES, where λ offspring are created by the application of a point mutation (i.e. $\mu_r\%$ of genes are modified) to the current solution. These $1+\lambda$ individuals are evaluated and the best one survives to the next generation. The process is repeated until a stop criterion is reached.

3.1 Objective Functions

Four objectives (to be minimized) are often considered in the design: *area*, *error*, *power dissipation*, and *propagation delay*. They are described in the sequence.

Area. Analyzing the area of a digital circuit is relevant as it affects the manufacturing cost. The larger the circuit c, the larger the area of the printed circuit board necessary for its implementation. The area occupied by c can be estimated by counting its processing elements (e.g., transistors, elementary ports, functional level components). Here, the area of c is defined as its number of gates.

Error Functions. The Hamming distance between two binary words is often adopted as the error measure in the evolutionary design of digital circuits [4]. Let $O^{(i)}_{orig}$ and $O^{(i)}_{app}$ denote, respectively, the output binary word of a fully functional digital circuit and that of an ADC generated for the input vector i. The Hamming distance is defined as $d_h = \sum_{\forall i}(O^{(i)}_{orig} \oplus O^{(i)}_{app})$, where $\oplus$ is *XOR*, i.e., the number of positions in which the bits of the two binary words differ from each other.

Power Dissipation. The main sources of power dissipation of a digital circuit are [2]: switching component of power ($P_{switching}$), short-circuit component of power ($P_{short-circuit}$), and the leakage current component of power ($P_{leakage}$). The power dissipation is then computed by $P = a_{0\to1}.C_L.f_{clk}.V^2_{dd} + I_{sc}.V_{dd} + I_{leakage}.V_{dd}$, where C_L is the load capacitance, f_{clk} is the clock frequency, $a_{0\to1}$ is the node transition activity factor, I_{sc} is the short circuit current, $I_{leakage}$ is

the leakage current, and V_{dd} is the supply voltage. The node transition factor quantifies the average number of times a logic gate makes a state transition that dissipates power within a period of *clock*. It can be defined as $a_{0\to1} = p_0.p_1 = p_0.(1-p_0)$.

Propagation Delay. The delay of a digital circuit c, the time spent for the changes in the input cause any effects in the output, is defined here as D_c and is calculated as the delay of the longest path according to $D_c = \max_{\forall p \in path} \sum_{c_i \in p} t_d(c_i)$, where t_d is the delay of a cell c_i and t_d is normally provided by the manufacturers.

4 Non-dominated Sorting Genetic Algorithm II

Most of the multiobjective EAs (MOEA) are based on the *Pareto dominance* concept which states that a solution $\mathbf{x_1}$ dominates a solution $\mathbf{x_2}$ if: (i) $\mathbf{x_1}$ is no worse than $\mathbf{x_2}$ in all objectives; and (ii) $\mathbf{x_1}$ is strictly better than $\mathbf{x_2}$ in at least one objective. NSGA-II [3] has two main features: non-dominated ranking and crowding distance. In the non-dominated ranking, the individuals in the set R_t –parent population (P_t) plus offspring population (Q_t)– are sorted according to their dominance. All Pareto optimal solutions, which are the feasible non-dominated solutions, form the first front F_1. The non-dominated solutions from $R_t \setminus F_1$[1] compose the second front F_2, and so on. A rank corresponding to the front index is assigned to each individual, i.e., i is given to the solutions in F_i. The individuals in the first fronts form P_{t+1}.

However, as the parent population size is $|R_t|/2$, where $|\cdot|$ denotes the cardinality of a set, one can determine i such that $|F_1 \cup F_2 \cup \cdots \cup F_{i-1}| \leq |R_t|/2$ and $|F_1 \cup F_2 \cup \cdots \cup F_i| > |R_t|/2$. Thus, some individuals in F_i can not be included in the next population, as the population size is fixed. The crowding-distance is calculated as the semi-perimeter of the hyperrectangle formed by the values of adjacent neighbor of each candidate solution in the objectives space and it is adopted by NSGA-II to select the candidate solutions in less populated regions.

5 The Proposed Method

The idea here is to design ACDC, considering multiple objectives, using CGP. Most of the MOEAs, such as NSGA-II, maintain a population of fixed size during the evolutionary process; the larger the population, the larger the number of trade-offs between the different objectives analyzed, as a larger population allows for better coverage of the search space. As a consequence, the chances of premature convergence is reduced. However, the computational cost for each iteration of the method associated with this population may be significant. In contrast, a small population leads to a coarser coverage, which can result in the non-exploration of promising areas of the search space increasing the probability

[1] The symbol "\" represents the operator of set difference.

that the algorithm gets stuck at a local optimum [8]. An attractive alternative is thus to permit that the size of the population varies during the search process. For this, we propose here the variation of the number of individuals in the population from one to a (user defined) maximum value. The proposed approach uses CGP to represent the candidate circuits, builds a set of Pareto solutions observing the trade-off between key circuit parameters and, when the population size becomes larger than Tam_Max, the crowding distance from NSGA-II [3] is applied to select individuals. Algorithm 1 presents a pseudocode of the proposal.

Algorithm 1. Pseudocode of the proposed technique.

```
1  t = 0;
2  P_t ← Initialize-Population(P_t);
3  Q_t ← ∅;
4  while Circuit-Evaluation ≤ N do
5  |   R_t ← P_t ∪ Q_t;
6  |   F_1 ← Non-dominated-Individuals(R_t);
7  |   P_t ← F_1;
8  |   if |P_t| > Tam_Max then
9  |   |   Crowding-distance(P_t);
10 |   |   Crowded-comparison-operator-sort(P_t, ≺_n);
11 |   |   P_t ← P_t[1 : Tam_Max];
12 |   end
13 |   for i ← 1 to |P_t| do
14 |   |   Q_i ← Mutate(P_i);
15 |   end
16 |   t = t + 1;
17 end
```

Initially, the population of parents (P_t) is initialized with a single (randomly generated) individual and the population of offspring (Q_t) is empty. The proposal evolves the candidate circuits while the stop criterion is not met. During the search, the parent and offspring populations are combined into a single population $R_t = P_t \cup Q_t$. Then, the non-dominated individuals from R_t, F_1, are selected to compose the population P_t. Thus, the size of the population P_t is not fixed, as the number of non dominated solutions may vary. Crowding distance [3] is applied when the population size is larger than a user-defined threshold (Tam_Max). This is used in order to avoid the population size to become very large. Crowding distance can be used to preserve the diversity of the solutions and is calculated as half of the perimeter of the cuboid formed by the nearest neighbor with respect to the objective function values. The individuals are sorted in descending order (Crowded-comparison-operator-sort) and the first ones are selected to compose P_t. Finally, new candidate circuits are generated using point mutation ($Mutate(P_i)$), the most commonly used mutation operator in CGP [7]. In this context, one allele of a gene is randomly selected and its

value is replaced by another valid value chosen randomly. The point mutation rate ($\mu_r\%$) is a parameter that affects the number of genes mutated.

To deal with constrained problems, the proposed method uses the Constrained NSGA-II approach [3]. In this context, it is said that a solution $\mathbf{x_1}$ dominates a solution $\mathbf{x_2}$ if any of the following conditions is true: (i) $\mathbf{x_1}$ is feasible and $\mathbf{x_2}$ is not; (ii) $\mathbf{x_1}$ and $\mathbf{x_2}$ are both infeasible, but $\mathbf{x_1}$ has a smaller overall constraint violation; and (iii) $\mathbf{x_1}$ and $\mathbf{x_2}$ are both feasible and $\mathbf{x_1}$ dominates $\mathbf{x_2}$.

6 Computational Experiments

The experiments were conducted in the extrinsic evolutionary scenario (search occurs in software) using the truth tables of w-bit ($w = 2, 8$) adder and multiplier, and an ALU circuits. The problems are labeled here as Aw and Mw, respectively, for adder and multiplier circuits, and they can be used to construct filtering structures, such as Finite Impulse Response Filter (FIR filter) and Infinite Impulse Response Filter (IIR Filter). These filters are quite important in digital signal processing such as image processing, video processing, audio processing, and wireless communication systems [5]. Also, adders and multipliers are used in circuits dedicated to calculate hyperbolic and trigonometric functions, such as the Coordinate Rotation Digital Computer (CORDIC) [1]. The Ripple Carry Adder and Ripple Carry Array Multiplier were adopted as reference architectures for the 2- and 8-bit adder and multiplier, respectively. Finally, the architecture of ALU is important for digital signal processing, and it is present in all computing devices such as microprocessors, computers and embedded systems. The SN54/74LS181 architecture[2], a 4-bit ALU which can perform 16 logic operations and also a diversity of arithmetic operations, was adopted here. The output is encoded by a 8-bit word, 4 bits are reserved for each variable, and 6 bits are used as controllers (logical/arithmetical operator). The data provided by a current manufacturer of digital devices is used to calculate power dissipation and delay (V_{dd}, C_L and t_d in Sect. 3.1). Nexperia[3] was selected as its gates have low t_d values. The gates are: 74LVC1G08 (AND), 74LVC1G08 (NAND), 74LVC1G32 (OR), 74LVC1G86 (XOR), 74LVC1G02 (NOR), HEF4077B (XNOR). These circuits are important but simples, as we conducted the experiments as a preliminary evaluation of the proposed approach.

The proposed CGPMO+APS[4] was compared to hES and its variant with periodization (hn^{10}) [6], and the CGP combined with NSGA-II [4]. The experiments were divided into three scenarios: (i) constrained multiobjective optimization (no domain information added), (ii) the design of ACDCs, (no domain information added); and (iii) the design of ACDCs with information from the domain specialist.

[2] http://pdf1.alldatasheet.com/datasheet-pdf/view/5671/MOTOROLA/SN54LS181.html.

[3] http://www.nexperia.com/products/logic/gates.

[4] The source-code of CGPMO+APS is available at https://github.com/ciml.

6.1 Scenario 1 – Constrained Multiobjective Optimization

Here, S2 and M2 are the problems, the Hamming distance is the measure of functional quality (a constraint in this scenario), and the area and delay are the objective functions to be minimized. The computational experiments were conducted as in [6], and the results obtained by the proposed CGPMO+APS are compared to those found by hES and hn[10]. The initial population of the CGPMO+APS is composed by 1 individual randomly generated, $\Gamma = \{0, 1, a, b, \overline{a}, \overline{b}, a.b, a.\overline{b}, \overline{a}.b, \overline{a}.\overline{b}, a \oplus b, a \oplus \overline{b}, a + b, a + \overline{b}, \overline{a} + b, \overline{a} + \overline{b}, a.\overline{c} + b.c, a.\overline{c} + \overline{b}.c, \overline{a}.\overline{c} + b.c, \overline{a}.\overline{c} + \overline{b}.c\}$ is the set of Boolean functions that can be executed by processing nodes, $\mu_r = 5\%$, $n_r = 1$, $n_c = 200$, $l = 200$, 400.000 fitness evaluations were allowed for S2, and 1.600.000 fitness evaluations for M2. Finally, 20 independent runs were performed.

In [6], the Additive Epsilon Indicator (AEI) and the Kruskal-Wallis non-parametric test were used. Here we do not present a statistical comparison using the results of CGPMO+APS as the results of each independent run were not provided in [6]. However, the capacity of generating feasible circuits with various combinations of area and delay was analyzed in [6]. Thus, we provide here a comparison in terms of feasible circuits generated by the techniques considered and these values are presented in Table 1. In this scenario we analyze the number of independent runs which resulted in functionally correct solutions. The results indicate that the proposed CGPMO+APS obtained better results in terms of fully functional circuits when compared to the other methods, as it is the only technique which found functionally correct circuits in all 20 independent runs.

Table 1. Number of independent runs with functionally correct solutions.

Method	S2	M2	#correct circuits S2	#correct circuits M2
hES	12	15	–	–
hn[10]	11	14	–	–
CGPMO+APS	**20**	**20**	**26**	**24**

6.2 Scenario 2 – Approximate Design of CLCs from Scratch

The present set of experiments is composed of three objective functions commonly employed when designing ADCs in the context of EH: (i) Hamming distance, (ii) delay, and (iii) power dissipation. The circuits S8, M8, and ALU were used as problems to be solved. The approach proposed in [4], labeled here as CGP+NSGA-II, was implemented and its results are used in the comparative analysis. Both CGP+NSGA-II and the proposed CGPMO+APS used the same parameter settings: $\Gamma = \{$AND, NAND, OR, XOR, NOR, XNOR$\}$, $\mu_r = 5\%$, $n_r = 1$. The number of columns were $l = n_c = 300, 200, 1000$ for the ALU, S8, and M8, respectively, and these values are those used in [4]. Also, the population size of CGP+NSGA-II is equal to 50.

Table 2. Parameters of circuits with the lowest error values when the population is randomly initialized and 30×10^6 circuit evaluations are allowed. The reference models (with error = 0) are the ALU SN54/74LS181 with delay = 22.22 ns and power = 1.14 mW, the Ripple Carry Adder with delay = 25.70 ns and power = 0.50 mW, and the Ripple Carry Array Multiplier with delay = 74.30 ns and power = 3.21 mW. Here, error is the percentage of incorrect outputs values, and delay and power are the ratio of the values observed in the circuits generated and those of the reference architecture.

	CGPMO+APS			CGP+NSGA-II		
	Error (%)	Delay (% ns)	Power (% mW)	Error (%)	Delay (% ns)	Power (% mW)
ALU	29.72	560.81	51.24	36.64	701.80	51.65
	29.81	517.57	41.41	37.35	685.14	48.87
	29.86	866.22	40.74	37.57	685.14	37.30
	29.91	371.62	27.81	37.67	685.14	36.70
	30.21	371.17	25.00	37.92	684.23	33.31
	30.38	327.93	25.76	38.14	383.33	50.62
	30.48	326.13	24.93	38.30	669.37	27.60
	30.52	326.13	21.49	38.32	373.87	32.45
	30.60	309.91	25.21	38.57	358.11	24.22
	30.79	317.12	21.42	38.96	358.11	23.19
S8	15.49	570.82	79.63	13.89	592.22	95.11
	15.54	570.82	73.98	14.18	578.21	90.20
	16.58	570.82	73.00	14.70	605.45	74.73
	16.63	564.20	64.86	14.87	578.21	87.25
	17.08	309.73	69.81	15.10	571.60	91.39
	17.32	564.20	64.27	15.44	591.05	70.79
	17.35	571.21	59.36	15.92	570.82	81.49
	17.60	296.50	68.99	16.49	310.12	97.90
	17.70	564.20	60.14	16.66	591.44	67.47
	17.95	296.50	63.68	16.88	302.72	77.83
M8	38.87	504.04	52.12	42.14	414.27	64.27
	38.92	504.04	32.81	42.36	317.50	56.22
	38.97	311.57	40.60	42.39	316.42	62.08
	38.98	308.88	38.06	42.79	411.44	47.87
	38.99	311.57	33.66	42.87	309.56	81.27
	39.02	308.88	33.67	42.91	409.15	49.12
	39.23	304.71	37.07	42.96	306.46	68.29
	39.25	219.11	47.72	42.99	411.31	39.85
	39.27	306.59	36.50	43.09	409.15	46.55
	39.28	219.11	46.86	43.17	299.87	69.49

Here we analyze the results obtained by CGPMO+APS and CGP+NSGA-II when the initial population was randomly generated and 30×10^6 objective function evaluations were allowed. Table 2 presents the operational characteristics of 10 candidate solutions for ALU, S8, and M8 with the smallest errors considering all independent runs. The techniques analyzed here were not able to design fully functional ALUs, S8, and M8 from scratch. It can also be noticed that no circuits reached delay values lower than or equal to the delay of the reference circuits. The ALU models obtained by CGPMO+APS show a smaller level of error than those found by CGP+NSGA-II. The (10) ALUs designed by CGPMO+APS and presented in Table 2 dissipate less power and have a smaller delay than those (10) obtained by CGP+NSGA-II. Regarding S8, the circuits generated dissipate less power than the reference architecture. Among the S8 circuits, that with the largest power dissipation (79.63% of the power dissipated by the reference architecture) was found by CGPMO+APS and it is superior, in terms of energy efficiency, to all 10 circuits that present the lowest error values found by CGP+NSGA-II. The error values of the S8 circuits obtained by CGPMO+APS and CGP+NSGA-II are similar. It is noted that the 10 M8 models with the smallest error values obtained by both approaches dissipate less power than the Ripple Carry Array Multiplier. Also, the M8 circuits found by CGPMO+APS dissipate less power and have smaller delay than those obtained by CGP+NSGA-II.

Table 3 presents the hypervolume found by CGPMO+APS and CGP+NSGA-II, and the p-values of the Kruskal-Wallis tests. CGPMO+APS presented the highest mean hypervolume values for all circuits analyzed. According to the Kruskal-Wallis test, the results found by the proposed CGPMO+APS are statistically different from those obtained by CGP+NSGA-II for S8 and ALU.

6.3 Scenario 3 – Approximate CLCs Using Conventional Models

Besides the design of ADCs from scratch, CGPMO+APS and CGP+NSGA-II were also applied to optimize conventional architectures by observing the trade-off between error, power, and delay. As a result, the techniques provide a set of ADCs to the specialist, who can choose the best for his/her application. Also, 30×10^6 circuit evaluations are allowed here.

The operational features of 10 circuits with the smallest errors for ALUs, S8, and M8 are shown in Table 4. CGPMO+APS and CGP+NSGA-II were able to obtain fully functional circuits in terms of error, and these circuits dissipate less power or present lower delay values than those of the base architecture for all the cases tested (ALU, S8, and M8). CGPMO+APS generated an ALU with lower values of delay and power dissipation than those of the conventional architecture. Also, with respect to S8 and M8, the fully functional circuits generated by the proposed method are better in delay or power dissipation. The results obtained by CGPMO+APS are, in general, better than those found by CGP+NSGA-II. This advantage is specifically large with respect to the delay values of ALU.

Table 3. Hypervolume values. The reference points are the highest values obtained for each objective function: $(0.5221, 192.3000, 0.5861)$, $(0.4722, 291.7000, 0.4865)$, and $(0.5257, 374.5000, 2.6115)$, respectively, for ALU, S8, M8. Kruskal-Wallis test is applied and the p-values are also presented.

Circuit	Method	Best	Median	Mean	STD	Worst	p-value
ALU	CGPMO+APS	0.3532	0.3049	**0.2965**	0.0438	0.2276	3.76×10^{-7}
	CGP+NSGA-II	0.3366	0.2720	0.2700	0.0449	0.1464	
S8	CGPMO+APS	0.4059	0.3439	**0.3475**	0.0314	0.2935	9.77×10^{-4}
	CGP+NSGA-II	0.3770	0.3041	0.3049	0.0456	0.2388	
M8	CGPMO+APS	0.3192	0.1524	**0.1615**	0.0873	0.0218	6.90×10^{-2}
	CGP+NSGA-II	0.1873	0.1475	0.1406	0.0384	0.0418	

Table 5 presents the hypervolume found by CGPMO+APS and CGP+NSGA-II. CGPMO+TP presented the highest mean values of hypervolume for all circuits analyzed. According to the Kruskal-Wallis test, the results of CGPMO+APS are statistically better than those of CGP+NSGA-II for all the tested circuits.

Table 4. Parameters of circuits with the lowest error values when the population is initialized with a conventional architecture and 30×10^6 circuit evaluations are allowed. The reference models (with error $= 0$) are those in the caption of Table 2.

	CGPMO+APS			CGP+NSGA-II		
	Error (%)	Delay (% ns)	Power (% mW)	Error (%)	Delay (% ns)	Power (% mW)
ALU	0.00	99.55	64.80	0.00	395.50	64.34
	0.00	404.05	64.74	0.00	723.87	64.30
	0.00	411.71	64.71	0.00	746.85	64.11
	0.00	544.14	64.11	0.52	395.50	62.55
	0.33	99.55	63.46	1.04	746.85	61.24
	0.68	86.04	65.11	1.70	396.40	60.59
	0.93	404.05	62.39	2.12	395.50	62.15
	1.04	404.05	61.85	2.61	403.15	60.20
	1.09	86.04	65.04	2.73	396.40	60.42
	1.27	86.04	64.66	2.86	723.87	59.93
S8	0.00	84.82	128.18	0.00	104.28	98.54
	0.00	107.39	99.28	0.00	104.67	71.58
	0.00	114.01	85.39	1.39	95.72	99.61
	1.37	86.77	99.05	1.40	101.56	98.43
	2.02	80.93	98.75	2.02	75.88	97.45
	2.60	60.31	100.74	2.28	68.48	97.71

(*continued*)

Table 4. (*continued*)

	CGPMO+APS			CGP+NSGA-II		
	Error (%)	Delay (% ns)	Power (% mW)	Error (%)	Delay (% ns)	Power (% mW)
	2.71	81.71	97.76	2.78	95.72	91.94
	3.13	80.93	98.30	3.58	67.70	97.71
	3.32	61.09	97.51	3.82	55.25	100.08
	3.47	86.77	97.33	4.17	97.28	91.00
M8	0.00	100.40	99.95	0.00	100.13	99.98
	0.38	100.40	99.93	0.82	98.11	99.77
	0.41	98.11	99.88	2.37	98.11	99.49
	0.61	98.11	99.77	5.32	110.76	98.79
	1.09	98.11	99.65	5.39	98.11	99.37
	1.15	100.13	99.52	5.40	98.11	99.22
	1.30	95.55	99.64	5.57	100.40	98.53
	1.57	95.28	99.21	6.14	97.57	98.37
	2.03	93.53	98.39	6.41	93.53	98.28
	3.02	99.86	97.74	6.77	93.53	98.24

Table 5. Hypervolumes values and p-values of the Kruskal-Wallis test. The reference points are: $(0.4284, 165.8000, 1.1348)$, $(0.4722, 74.4000, 0.6369)$, and $(0.5353, 82.3000, 3.2135)$, respectively, for ALU, S8, and M8.

Circuit	Method	Best	Median	Mean	STD	Worst	p-value
ALU	CGPMO+APS	0.5354	0.4904	**0.4909**	0.0347	0.4340	2.8701×10^{-9}
	CGP+NSGA-II	0.4088	0.3397	0.3357	0.0450	0.2687	
S8	CGPMO+APS	0.5025	0.4849	**0.4687**	0.0345	0.4192	6.7183×10^{-8}
	CGP+NSGA-II	0.4422	0.4016	0.3944	0.0392	0.3326	
M8	CGPMO+APS	0.3730	0.3101	**0.2996**	0.0503	0.2028	3.1732×10^{-5}
	CGP+NSGA-II	0.2601	0.2204	0.2135	0.0368	0.1478	

7 Concluding Remarks and Future Work

A multiobjective Cartesian Genetic Programming technique with adaptive population size (CGPMO+APS) was proposed here to design approximate combinational digital circuits (ACDCs). Three situations were considered in the experiments: (i) constrained multiobjective optimization (no domain information added), (ii) the design of ACDCs (no domain information added); and (iii) the design of ACDCs with information from the domain specialist.

In (i), the results indicated that CGPMO+APS was able to design more feasible circuits than the other approaches analyzed. In (ii) and (iii), the results show

that both CGPMO+APS and CGP+NSGA-II are not suitable for the construction of complex architectures such as ALUs, S8, and M8 without the introduction of the knowledge of the domain expert. On the other hand, when a conventional architecture is adopted as the initial solution, the two approaches synthesized ALUs, S8, and M8 that do not present errors with respect to the truth table and with improvement in delay or power dissipation when compared to the reference architecture. Particularly, the CGPMO+APS obtained a fully functional ALU with lower delay and power dissipation than those of the reference architecture. Also, CGPMO+APS obtained better mean values of hypervolume than those of CGP+NSGA-II.

The use of Binary Decision Diagrams to reduce the processing time and solving more complex problems are relevant research avenues.

Acknowledgments. We thanks the support provided by CNPq (312337/2017-5 and 312682/2018-2), FAPEMIG (APQ-00337-18), PPGCC, and PPGMC.

References

1. Aggarwal, S., Meher, P.K., Khare, K.: Concept, design, and implementation of reconfigurable cordic. IEEE Trans. Large Scale Integr. (VLSI) Syst. **24**(4), 1588–1592 (2016)
2. Chandrakasan, A.P., Brodersen, R.W.: Minimizing power consumption in digital CMOS circuits. Proc. IEEE **83**(4), 498–523 (1995)
3. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
4. Hrbacek, R., Mrazek, V., Vasicek, Z.: Automatic design of approximate circuits by means of multi-objective evolutionary algorithms. In: International Conference on Design and Technology of Integrated Systems in Nanoscale Era (DTIS), pp. 1–6 (2016)
5. Julio, R.O., Soares, L.B., Costa, E.A.C., Bampi, S.: Energy-efficient gaussian filter for image processing using approximate adder circuits. In: 2015 IEEE International Conference on Electronics, Circuits, and Systems (ICECS), pp. 450–453 (2015)
6. Kaufmann, P., Knieper, T., Platzner, M.: A novel hybrid evolutionary strategy and its periodization with multi-objective genetic optimizers. In: IEEE Congress on Evolutionary Computation, pp. 1–8 (2010)
7. Miller, J.F.: Cartesian Genetic Programming. Springer, Berlin (2011). https://doi.org/10.1007/978-3-642-17310-3
8. Roeva, O., Fidanova, S., Paprzycki, M.: Influence of the population size on the genetic algorithm performance in case of cultivation process modelling. In: Federated Conference on Computer Science and Information Systems, pp. 371–376 (2013)
9. Stepney, S., Adamatzky, A.: Inspired by Nature. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-67997-6
10. Vasicek, Z., Sekanina, L.: Evolutionary approach to approximate digital circuits design. IEEE Trans. Evol. Comput. **19**(3), 432–444 (2015)
11. Venkataramani, S., Sabne, A., Kozhikkottu, V., Roy, K., Raghunathan, A.: SALSA: systematic logic synthesis of approximate circuits. In: DAC Design Automation Conference, vol. 2012, p. 796–801 (2012)