Design of Quality-Configurable Approximate Multipliers Suitable for Dynamic Environment

Vojtech Mrazek, Zdenek Vasicek *Member, IEEE,* and Lukas Sekanina *Senior Member, IEEE,*
Brno University of Technology, Faculty of Information Technology, IT4Innovations Centre of Excellence
Brno, Czech Republic
Email: imrazek@fit.vutbr.cz, vasicek@fit.vutbr.cz, sekanina@fit.vutbr.cz

***Abstract*—In the context of approximate computing, many different design approaches have been introduced in recent years exploiting the application error resilience in order to reduce power consumption. Among others, the concept of Quality Configurable Circuits (QCC) has been proposed. QCCs allow us to dynamically modulate the degree to which their functionality is approximated with respect to the desired quality of service. Existing approaches typically construct complex approximate circuits using smaller approximate and exact components. This approach is, however, associated with a considerable area and power overhead. To mitigate this issue, we propose a holistic approach enabling to formulate the QCC design problem as a search problem that can be solved by means of evolutionary design techniques. In order to prove this concept, various quality configurable 8-bit multipliers were designed. Compared to the state-of-the-art solutions, the proposed quality configurable multipliers exhibit better tradeoffs in terms of electrical as well as quality parameters.**

I. Introduction

Improving energy efficiency of modern computing systems is the main challenge in today's digital design. Computing capabilities of mobile devices such as smartphones have grown exponentially in the past decades, however, battery technology did not follow the same evolution. Moreover, autonomy is becoming a critical issue in computer-based systems.

Many fundamentally new and different approaches have recently been introduced under the term of *approximate computing* to address the relentless rise in demand for energy-efficient systems. A good survey of existing techniques can be found, for example, in [1], [2]. The concept of approximate computing exploits the idea of accepting a certain level of inaccuracy in computations to reduce complexity and other non-functional properties (such as delay and power consumption) of digital systems. The key motivation behind the approximate computing concept is the inherent error resilience of many real-world applications. For example, Chippa et al. reported that more than 83% of runtime is spent in computations that can be approximated [3]. An open question is how to systematically approximate circuits and software in order to obtain desired quality of service (i.e. an acceptable error) with a minimum power budget.

Quality Configurable Circuits have been introduced as a natural response to the increasing demand for approximate circuits that are able to adapt their behavior and energy requirements to a variable workload. Quality Configurable Circuits are approximate circuits that can dynamically modulate the degree to which their functionality is approximated based on the desired output accuracy. For example, power consumption can be dynamically traded for accuracy depending on the state of an application. Existing approaches typically construct large approximate circuits using smaller components such as configurable 2-bit approximate multipliers [4] or maskable adders [5]. Composing a larger circuit from smaller subcircuits, however, typically introduces overhead. For example, when 8-bit approximate multipliers are composed of 2-bit multipliers, the overhead is substantial (see Section III). This paper addresses the problem of automatic design of quality configurable circuits. As a case study, 8-bit multipliers are chosen because they represent a key component of many systems. In particular, quality configurable multipliers supporting two modes of operation are considered.

II. Approximate Computing

In order to reduce the energy consumption, one could use the so-called *overscaling technique* for approximate computing. Overscaling means, that we let the supply voltage intentionally drop below a critical point that guarantees a reliable operation. As a consequence of that, the timing errors are induced on critical paths. Unfortunately, these errors often lead to large computational error [2]. In addition to that, the energy efficiency gain for this approach is relatively small. Hence the designers are gradually turning away from overscaling and the majority of the recently proposed methods resorts to *functional approximation*. The idea of functional approximation is to implement a slightly different function to the original one provided that the accuracy is kept at desired level and the power consumption or other electrical parameters are reduced adequately. Functional approximation can be introduced at various levels of circuit description.

One of the possibilities how to functionally approximate a given circuit is to make a change at the level of behavioral description, e.g., in the truth table. This idea was employed, e.g., by Kulkarni et al. who manually designed a small 2-bit approximate multiplier consisting of five gates [6], which was then employed in larger approximate multipliers. Unfortunately, the manual design represents a time consuming process which is feasible for small circuits only. The current trend is to develop fully automated functional approximation methods

978-1-5386-7753-7/18/$31.00 © 2018 IEEE

that can be integrated into computer aided design tools for digital circuits. Majority of the currently available methods are *error-oriented* in the sense that all logic optimizations leading to an approximate solution are constrained by a predefined *error criterion* [7]–[9]. The error can be expressed by various metrics such as the error magnitude, the average error magnitude or the error probability. The error-oriented approach, however, represents only one possibility of obtaining an approximate circuits design. Sekanina et al. introduced an alternative *area-oriented* approach [10], based on genetic programming, exploiting the fact that the evolutionary design is capable of constructing partially working solutions even if sufficient resources (required for creating a fully functional solution) are not available. The user can control the used area (and so power consumption which is highly correlated with occupied resources) more comfortably than by means of the error-oriented methods.

A. Automated design of approximate circuits

The approximate design methods typically employ various heuristics to identify circuit parts suitable for approximation. Probabilistic pruning is one of the first approaches that addresses the problem of approximate synthesis [11]. The goal is to remove circuit nodes or blocks and their associated wires in order to trade the exactness of computation for power, area, and delay improvement. The decision to prune a node is generally based on the significance, which is a structural parameter, and the activity (toggle count).

The Systematic methodology for Automatic Logic Synthesis of Approximate circuits (SALSA) represents another of the early approaches [7]. In order to be able to use the existing synthesis tools, the authors mapped the problem of approximate synthesis into an equivalent problem of traditional logic synthesis – don't care based optimization. A different systematic approach, denoted as Substitute-And-SIMplIfy (SASIMI), tries to identify signal pairs in the circuit that exhibit the same value with a high probability, and substitutes one for the other [9]. The substitutions introduce functional approximations and hence some redundant nodes appears. These nodes can be eliminated from the circuit which results in area and power savings.

In order to deliver better trade-offs, various more advanced approaches have been applied to obtain a more powerful method for approximate re-synthesis, in which the original logic function is replaced by a cheaper implementation [12]–[14]. The problem of approximate re-synthesis is typically mapped to a search-based design problem. An automated circuit approximation procedure is seen as a multi-objective search process in which a circuit satisfying user-defined constraints and showing desired trade-off between the quality and other electrical parameters is sought in the space of all possible implementations. A heuristic procedure (e.g. an evolutionary algorithm) that gradually modifies the original implementation is typically utilized. Among others, the following search-based approaches have been proposed. Nepal et al. introduced a technique for automated behavioral synthesis of approximate computing circuits (ABACUS) [15]. ABACUS uses a simple greedy search algorithm to modify abstract synthesis tree. In order to approximate gate-level digital circuits, Sekanina and Vasicek introduced an evolutionary approach based on Cartesian Genetic Programming (CGP) [12], [13]. The reasons for using the advanced evolutionary approach was that the population-based approach suits well in finding multiple solutions and its niche-preservation methods can be exploited to discover diverse solutions. As shown in [16], this approach is able to produce high-quality approximate circuits that are unreachable by traditional approximate techniques.

B. Approximate arithmetic circuits

The key circuit components such as small adders, multipliers, FIR and IIR filters, and DCT and FFT blocks have already been approximated [1], [7], [9], [15], [16]. We restrict our attention to the multipliers because they represent a key component of many signal processing systems.

The following techniques have been proposed in approximate implementations of multipliers [17]: (1) Approximation in generating partial products utilizing a simpler structure to generate partial products [6]. (2) Approximation in the partial product tree by ignoring some partial products or dividing partial products in to several modules and applying approximation in the less significant modules. (3) Using approximate counters or compressors in the partial product tree. (4) Composing complex approximate multipliers by means of simple approximate multipliers as shown in [6].

In addition to that, the general design methods such as SALSA or SASIMI were also employed to approximate the multipliers. Recently, an evolutionary approach was applied in the task of approximate circuit design with respect to multiple objectives [12], [16]. The circuit approximation problem was formulated as a multi-objective search problem and solved using the state-of-the art CGP method [18] combined with the NSGA-II algorithm. By means of this method a rich library consisting of the 471 high-quality Pareto optimal 8-bit approximate multipliers was generatedin [16].

Various arithmetic error metrics were employed to assess the quality of arithmetic circuits [19]. Let $O^{(i)}_{orig}$ be an output value of the fully functional circuit for an input vector i and $O^{(i)}_{approx}$ be an output value of an approximate circuit. Then, the error magnitude defined as the maximum difference between the output of approximate and precise multiplier is equal to

$$\mathrm{WCE} = \max_{\forall i} |O^{(i)}_{orig} - O^{(i)}_{approx}|. \tag{1}$$

In addition to that, the mean error magnitude and the mean relative error magnitude are typically considered

$$\mathrm{MAE} = \frac{1}{2^{n_i}} \sum_{\forall i} \left| O^{(i)}_{\mathrm{orig}} - O^{(i)}_{\mathrm{approx}} \right| \tag{2}$$

$$\mathrm{MRE} = \frac{1}{2^{n_i}} \sum_{\forall i} \frac{\left| O^{(i)}_{\mathrm{orig}} - O^{(i)}_{\mathrm{approx}} \right|}{O^{(i)}_{\mathrm{orig}}} \tag{3}$$

The WCE-oriented approximation is important not only in time-critical and dependable systems, but also in image and signal processing, where low MAE, but excessive WCE can produce unacceptable results.

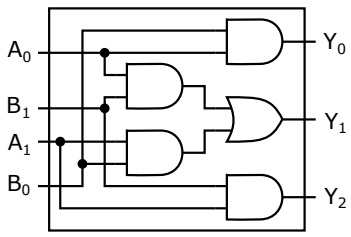
The approximation error is typically obtained using the exhaustive simulation for small multipliers [16]. For more complex instances such as 16-bit and 32-bit multipliers, techniques of formal verification have to be applied [20].

III. Quality-configurable circuits

Another class of approximate circuits are *quality configurable circuits* (QCCs) allowing the user to select one of the circuit configurations that differ in the quality of processing and power consumption. The quality configurable circuits naturally support dynamic approximation as a response to variable requirements on the quality of result [4]. For example, these circuits can be utilized in the signal processing applications that can thus benefit from an in situ dynamic adaptation of the quality of processing in response to variable requirements on the quality of result and available resources.

QCCs are typically constructed from elementary building blocks that are created in such a way that their error can be modified using several configuration bits. As the configuration bits can disconnect some preselected parts of the circuit, the power consumption can be dynamically changed depending on the actual application needs.

Several quality configurable multipliers (QCMs) have been introduced by Shafique et al. [4], [21]. A configurable 2-bit multiplier operating in two modes has been proposed. The configuration is driven by a single configuration signal. In the first mode, the multiplier works exactly as the approximate multiplier proposed by Kulkarni et al [6]. It exhibits a single erroneous output if both the input operands are equal to 3 (see Fig. 1). In this case, the value 7 is returned instead of 9. In the second mode, a correction circuit is activated (the correction circuit is shown in Fig. 2). This circuit modifies the output value of the Kulkarni's multiplier if it equals to 7. Then, the value is increased by two. This multiplier is used as a building block for construction of larger 8-bit and 16-bit multipliers.



B \ A	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	4	6
3	0	3	6	7

Fig. 1. 2-bit approximate multiplier proposed by Kulkarni in [6]

Yang et al. utilized a different approach [5]. In order to obtain a configurable 8-bit multiplier, a carry-maskable one-bit adder was employed to generate the final product. In addition to that, the partial product tree of the multiplier was approximated by a tree compressor. This architecture enables to flexibly select the length of the carry propagation to satisfy the accuracy requirements. The shorter length implies lower switching activity which results in the reduction of power consumption and delay.

As the technology nodes shrink, the static power constitutes the predominant part of the total energy consumption. The straightforward way to reduce the static power is to turn off the power supply of the blocks that are currently inactive. In general, this technique, called *power gating*, can be applied to reduce dynamic as well as static power as it can be used for the so called logic isolation. A logic block (island) is assigned its own supply voltage from a virtual V_{DDV} rail. When the block is active, the V_{DDV} is connected to the power supply voltage V_{DD}. When the block is inactive, the switches are turned off allowing V_{DDV} to float and gradually sink toward 0. There are various ways in which a portion of logic can be isolated [22]. The simplest approach would be to use latches or AND/OR gates at the inputs of the island to prevent switching activity within it. Similarly, multiplexers can be inserted at the island outputs.

The power gating principle is employed in the work of Jain et al. [22]. The idea is to identify portions of logic in the circuit that consume significant power, but contribute only minimally to the output accuracy, and to isolate them in a quality-aware manner, i.e., to suppress the selected logic portions from being evaluated provided the output satisfies the desired quality requirement. One or more approximate modes of the circuit operation are then enabled by isolating the identified logic (using multiplexers, latches or power gating cells). Recently, configurable multipliers with two approximation modes based on four approximate 4:2 compressors were proposed by Akbari et al. [23]. In the approximate mode, only the approximate part is active whereas the supplementary part along with some components of the approximate part is invoked in the exact operating mode. The power gating technique was used to turn OFF the unused components of the approximate part. In the exact operating mode, tristate buffers are utilized to disconnect the outputs of the approximate part from the primary outputs. Each of the proposed compressors has its own level of accuracy in the approximate mode as well as different delays and power dissipations in the approximate and exact modes.

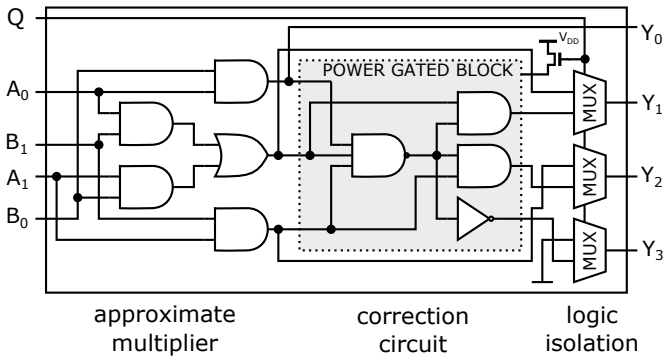


Fig. 2. Quality configurable approximate 2-bit multiplier

Let us investigate a common approach to designing QCM. Fig. 2 depicts the structure of a configurable 2-bit multiplier implemented using the power gating technique. The multiplier consists of two blocks. While the first block is always active, power gating is employed to control the second block. In the approximate mode, the approximate 2-bit multiplier is enabled only. In the second mode, the correction circuit, as proposed by Shafique et al., is activated. In this mode, the QCM operates as an exact 2-bit multiplier. Multiplexers are employed to isolate undefined accurate output values in the approximate quality mode. The parameters of the 2-bit QCM synthesized using Synopsys DC with 45nm FreePDK are given in Table I. Compared to the common multiplier implemented using Verilog star operator, the 2-bit QCM exhibits some overhead when we consider area, power, and delay. In the approximate mode, however, the electrical parameters (i.e. power and delay) are significantly improved. The 2-bit QCM in the approximate mode consumes 36% less power compared to the accurate multiplier.

TABLE I
PARAMETERS OF 2-BIT AND 8-BIT EXACT AND CONFIGURABLE APPROXIMATE MULTIPLIERS ACCORDING TO [4], [21].

Multiplier	operating mode	*Power* μW	*Delay* ns	*Area* μm^2	MAE	WCE
2-bit exact	-	3.8	0.15	19	0%	0%
2-bit QCM	approx.	2.4	0.09	12*	13%	13%
	exact	4.7	0.21	23	0%	0%
8-bit exact	-	428.3	1.25	727	0%	0%
8-bit QCM	approx.	483.0	1.51	1197*	1.4%	22%
	exact	516.4	1.60	1337	0%	0%

* The value is the area of the approximate part only.

The small 2-bit QCMs can be used as a building block for construction of larger configurable multipliers. We can employ a divide-and-conquer strategy for synthesizing a $2n$-bit multiplier from four n-bit multipliers. The operands are divided into four n-bit chunks (each operand has a lower and higher part) that are independently processed using four multipliers whose outputs are reduced using two adders with one n-bit and one $2n$-bit operand each. This approach can be applied recursively. It means that sixteen 2-bit QCMs are required to implement a single 8-bit QCM. Depending on the application, a coarse grained as well as a fine grained configuration can be supported. In the first case, all 2-bit QCMs share the same configuration bit. In the second case, every 2-bit QCM has a dedicated configuration bit. The parameters of the exact as well as configurable 8-bit multiplier supporting two approximate modes are given in Table I. Although the Kulkarni's 2-bit approximate multiplier is very effective, the recursive construction introduces a substantial overhead because it uses a large amount of accurate adders for summing the partial products. In the approximate mode, the QCM consumes even about 13% more power than the accurate multiplier. Switching from the exact mode to the approximate mode reduces the power by 6% only.

IV. EVOLUTIONARY DESIGN OF QUALITY CONFIGURABLE CIRCUITS

In order to approximate digital circuits, various approaches have been proposed. In this work, we employ CGP [18]. CGP especially differs from other GP branches in (i) the solution representation and (ii) the search mechanism. CGP can easily handle constraints given on candidate circuits, the method is naturally multi-objective and high-quality approximate circuits have already been obtained with CGP [13], [16].

This section introduces the idea of the proposed design method, the utilized evolutionary algorithm and the construction of the fitness function for the design of quality configurable approximate circuits.

A. Representation of digital circuits

From a hardware designer point of view, every candidate circuit is represented as a netlist containing a constant number of components. These components are (virtually) organized in a two-dimensional grid of n_c columns and n_r rows. Type of components depends on the level of abstraction used in modeling, where logic gates and RT level components are naturally supported. Every component has up to n_a inputs and n_b outputs. We restrict ourselves to single output components (i.e. $n_b = 1$).

A unique address is assigned to all primary inputs and to the output of all components to define an addressing system enabling circuit topologies to be specified. The primary inputs are labeled $0 \dots n_i - 1$, where n_i denotes the number of primary inputs. The outputs of components are labeled $n_i, n_i + 1, \dots, n_i + n_c \cdot n_r - 1$. No feedback connections are allowed in the basic version of CGP. Each component is represented using $n_a + 1$ integers in the netlist, where n_a integers specify destination addresses for its inputs and one integer is a pointer to the table Γ defining all supported functions. A component placed in the j-th column can obtain its input

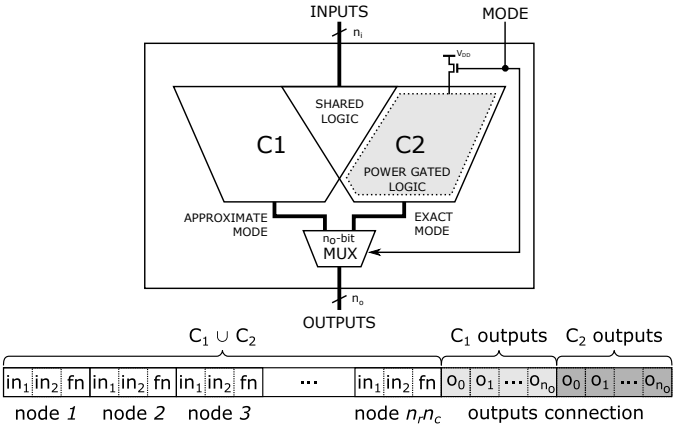


Fig. 3. Representation of the quality configurable circuits supporting two approximate modes. The chromosome consists of three parts. In the first part, array of CGP nodes is encoded. This part is followed by n_o integers defining the outputs of the QCC in the first mode. The last part contains n_o integers specifying the outputs of QCC in the second mode.

values either from primary inputs or from the components placed in the previous columns.

The whole circuit is then represented using a string consisting of $n_c \cdot n_r \cdot (n_a + 1) + n_o$ integers, where n_o denotes the number of primary outputs. The last n_o integers specify the indexes where the primary outputs are connected to. In addition to that, logic constants ('0' and '1') are allowed to be directly connected to the primary outputs in approximate circuit design. Our goal is to design a quality configurable circuit supporting two approximate modes. One of the possibilities how to solve this design problem is to extend the common CGP encoding and append additional n_o integers specifying the outputs in the second mode of operation (see Fig. 3).

For example, Fig. 4a shows a gate level 4 input / 4 output QCC consisting of 9 gates and having 3 logic levels on the longest input-output path. This circuit is represented in the CGP grid with $n_c = 3$ and $n_r = 3$ and the outputs of its components are labeled $4 \ldots 12$. There is a single unused component (gate having index 10).

The main feature of the CGP encoding is that while the number of nodes is constant, the size of the represented circuits is variable (from 0 to $n_c \cdot n_r$ components can be involved) as some components can remain disconnected. This redundancy has been identified as a crucial property of the efficient search in the space of digital circuits [18].

B. Fitness Function

The considered QCC shown in Fig. 3 consists of two subcircuits and works as follows. In the approximate mode, circuit C1 is active. The remaining logic is switched off. In the exact mode, power gated logic is activated. It means that both circuits C1 and C2 are active. The QCC is optimized for the scenario in which the approximate mode is active for a predominant part of the whole runtime. This is the reason, why C1 is always powered. For applications where the switching time is not critical, it would be more efficient to apply power gating also on a part of C1 that is not shared between modes, i.e. to $\mathrm{C1} \setminus \mathrm{C2}$, where C1 denotes the set of gates of circuit C1, C2 denotes the set of gates of circuit C2 and $\setminus$ is the set difference.

The goal of the evolution is to find a circuit showing the highest possible power reduction when it operates in the approximate mode. In addition to that, we are looking for a QCM whose electrical parameters are almost identical with the accurate multiplier. In order to simplify the problem we exploit the fact that power consumption is typically highly correlated with occupied resources [10]. It means that we can base the search on the optimization of the area on the chip. The problem is formulated as follows. Let ε be the maximum acceptable worst-case error. Then, the fitness value of a candidate QCC is calculated as

$$fitness = -\begin{cases} 2|\mathrm{C1}| + |\mathrm{C2} \setminus \mathrm{C1}| & \text{if } \mathrm{WCE}(C1) < \varepsilon \wedge \\ & \mathrm{WCE}(C2) = 0 \\ \infty, & \text{otherwise,} \end{cases}$$

where $|\mathrm{C1}|$ denotes the area of the subcircuit C1 and $|\mathrm{C2} \setminus \mathrm{C1}|$ denotes the area of the power gated logic. The goal of the evolutionary algorithm is to maximize the fitness and thus minimize the area.

C. Proposed Seeding Strategies

CGP employs a simple search method. Firstly, the initial population P is created. In the approximate circuit design, the initial population is typically seeded by an original circuit ought to be approximated. The next step consists in the evaluation of candidate circuits using the fitness function. Each member of P then receives the so-called fitness score and the highest-scored individual becomes a new parent of the next population. From this parent, λ candidate solutions are generated using mutation operator. The termination criterion is given by the number of iterations. The mutation operator modifies up to h randomly chosen integers of the chromosome. Their new values are generated randomly, but it is guaranteed that the new values are valid. One mutation can affect either the gate function, gate input connection, or primary output connection.

Two seeding strategies are considered in this paper. Firstly, the initial population is seeded by a single candidate circuit that consists of a single instance of the original circuit. If the goal is to design a quality configurable n-bit multiplier, the CGP encodes a netlist with the exact multiplier. The exact multiplier has $2n$ primary outputs. The CGP uses $n_o = 2 \cdot 2n$ primary outputs. The first n_o outputs are connected to the

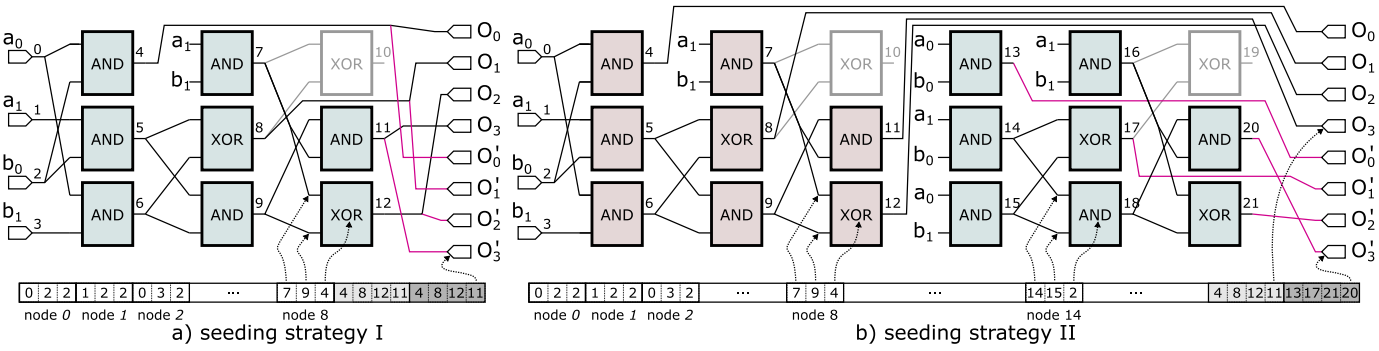


Fig. 4. Encoding of the initial QCMs created using a 2-bit multiplier and their netlists. O_i are the outputs of subcircuit C1, O_i' are the outputs of C2.

outputs of the multiplier and the same applies for the second n_o outputs. Both output parts share the same logic and the CGP is forced to separate this logic. This approach is denoted as *seeding strategy I* (SS-I) and it is illustrated in Fig. 4a. Secondly, the initial population is seeded by a single candidate circuit that consists of two identical instances of the original circuit. The first n_o outputs are connected to the first instance and the second n_o outputs are connected to the second instance. The CGP is forced to find the largest common part. This approach is denoted as *seeding strategy II* (SS-II) and it is illustrated in Fig. 4b.

V. Experimental Setup

The CGP parameters were initialized as follows. We employed $\lambda = 6$ individuals in the population, the number of mutations was $h = 5$, and the number of rows $n_r = 1$. When the first seeding strategy is employed, the number of columns equals to the number of components of the exact conventional multiplier used to seed the initial population. The number of columns was two-times bigger when the second seeding strategy is employed. Each CGP node has two inputs and one output (i.e. $n_a = 2$, $n_b = 1$) and could implement one of the following eight functions: Γ = {BUF (buffer), INV (inverter), AND, OR, XOR, NAND, NOR, XNOR}. The evolution was executed for 1 hour.

Various 8-bit multipliers were employed to seed the initial population. In particular, Ripple-Carry Array multiplier, two Carry-Save Array multipliers and three different Wallace Tree architectures. The array multiplier offers the lowest speed but occupies the smallest area on a chip compared to the other variants and especially the most expensive Wallace tree multiplier. Wallace-tree adder multiplier is the fastest known architecture which sums the partial products using multiple levels of carry-save adders.

The evolutionary design was repeated for 12 target arithmetic errors $\varepsilon \in \{0.01\%, 0.02\%, 0.05\%, 0.1\%, 0.2\%, 0.5\%, 1\%, 2\%, 5\%, 10\%\ 15\%\ 20\%\}$. Considering two seeding strategies, 12 target errors and 6 types of initial multiplier architectures, there are 144 different configurations in total. For each configuration, 10 independent evolutionary runs were executed.

We obtained more than one thousand of different 8-bit QCMs. All QCMs were synthesized by means of Synopsys Design Compiler using 45nm technology in order to determine power and delay parameters for each operation mode.

VI. Results

A. Search algorithm

In the first experiment, we investigated the impact of the seeding strategy on the size of resulting circuits. Fig. 5 shows the number of CGP nodes (i.e. the circuit size) in the exact and approximate multiplier mode for several target worst case errors. Box plots are constructed from all 60 independent CGP runs executed for each error level. A general observation is that the number of CGP nodes decreases with ε for both SS-I and SS-II. However, the impact on the circuits size is almost negligible in SS-I. In SS-II, the size of circuits in the exact mode is doubled with respect to SS-I for low ε. This undesirable property becomes eliminated for higher error levels. In the approximate mode, the size of circuits produced by SS-II is significantly reduced with increasing ε.

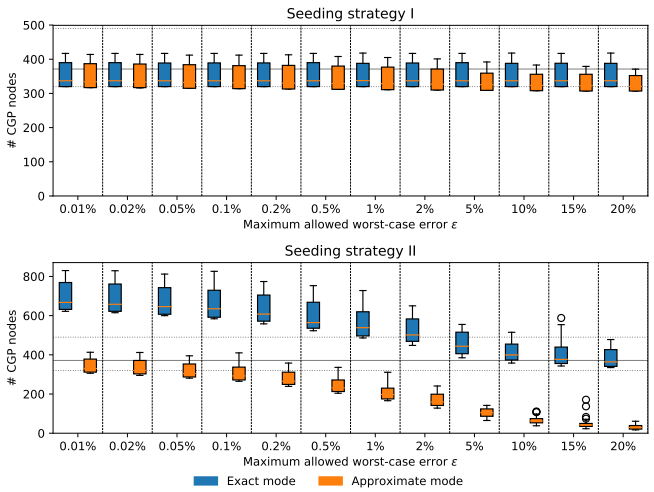


Fig. 5. The number of CGP nodes (circuit size) of the logic utilized in the exact mode ($|C1 \cup C2|$) and approximate mode ($|C1|$) for several target worst case errors. Box plots are constructed from 10 independent CGP runs seeded with 6 different exact multipliers.

B. Results of synthesis

As SS-II leads to solutions significantly different in the exact and approximate mode in terms of the number of CGP nodes (i.e. in the circuit size and power consumption), we will analyze only the circuits evolved with SS-II. For a further analysis we randomly selected three QCMs (with WCE = 0.019%, 1.979% and 9.914% in the approximate mode) that are denoted A1, A2 and A3 in Table II. These circuits were compared with exact multipliers (Verilog star operator, Dadda multiplier, composite multiplier constructed using 2x2 bit multipliers), approximate multiplier (again, with a composite structure) and various QCMs reported in [23]. Table II shows that the proposed QCMs exhibit very good properties especially in the approximate mode. In the exact mode, the selected multipliers exhibit parameters that are comparable with the exact multiplier built from 2-bit multipliers. The chosen QCMs consume about 30% more power compared to the optimal exact multiplier. It is not fair, however, to compare these multipliers directly. Firstly, the power gating introduces some overhead. Secondly, the logic that is used in the approximate mode is intentionally active even in the exact mode because the QCMs are optimized for applications where the approximate mode is active most of the time. In the approximate mode, the QCMs exhibit substantially better parameters compared to the approximate multipliers based on 2-bit approximate components. While the QCM A1 exhibits by two orders of magnitude better MAE (one order in the magnitude for MRE) compared to the QCM constructed using 2-bit multipliers, it

TABLE II
PARAMETERS OF THE PROPOSED QUALITY CONFIGURABLE 8-BIT MULTIPLIERS AND VARIOUS EXACT AND APPROXIMATE MULTIPLIERS.

MULTIPLIER	APPROXIMATE MODE						EXACT MODE			
	MAE	*MRE*	*Area*	*Delay*	*Power*	*PDP*	*Area*	*Delay*	*Power*	*PDP*
Exact (Verilog star operator)	-	-	-	-	-	-	100.0%	100.0%	100.0%	100.0%
Exact (Dadda) [23]	-	-	-	-	-	-	124.3%	102.7%	102.1%	104.8%
Exact (built from exact 2×2)	-	-	-	-	-	-	159.8%	122.4%	112.0%	137.1%
Approx. (built from approx. 2×2 [6])	899.6	3.2%	89.4%	112.0%	83.2%	93.3%	-	-	-	-
QCM (built from approx. 2×2 [6])	899.6	3.2%	152.3%	120.8%	112.0%	135.3%	170.0%	128.0%	119.8%	153.3%
QCM DQ4:2C$_1$ [23]	3540.0	36.9%	48.7%	46.7%	30.8%	14.4%	125.3%	104.0%	101.7%	105.7%
QCM DQ4:2C$_2$ [23]	2400.0	29.3%	40.2%	61.3%	29.2%	17.9%	127.4%	105.3%	102.4%	107.8%
QCM DQ4:2C$_3$ [23]	3220.0	32.8%	65.2%	65.3%	61.0%	39.9%	128.4%	105.3%	102.6%	108.1%
QCM DQ4:2C$_4$ [23]	1390.0	8.1%	75.4%	70.7%	64.5%	45.6%	132.4%	106.7%	103.1%	109.9%
QCM DQ4:2C$_{mixed}$ [23]	1460.0	11.9%	52.7%	69.3%	42.5%	29.4%	127.9%	105.3%	102.1%	107.6%
Proposed QCM A1 (WCE= 0.019%)	4.3	0.2%	92.2%	89.6%	97.2%	87.1%	124.5%	79.5%	129.4%	102.8%
Proposed QCM A2 (WCE= 1.979%)	329.5	16.7%	45.0%	74.7%	36.8%	27.5%	135.8%	101.7%	128.7%	130.9%
Proposed QCM A3 (WCE= 9.914%)	1668.2	35.4%	16.7%	40.4%	9.3%	3.7%	112.3%	139.4%	105.9%	147.6%

occupies the area comparable to the area of non-configurable Kulkarni's approximate multiplier. The MAE of DQ4 QCMs is very high. It is thus natural to expect higher power reduction. Interestingly, QCM A3 with comparable MAE overcomes these QCMs not only in the power but also in delay.

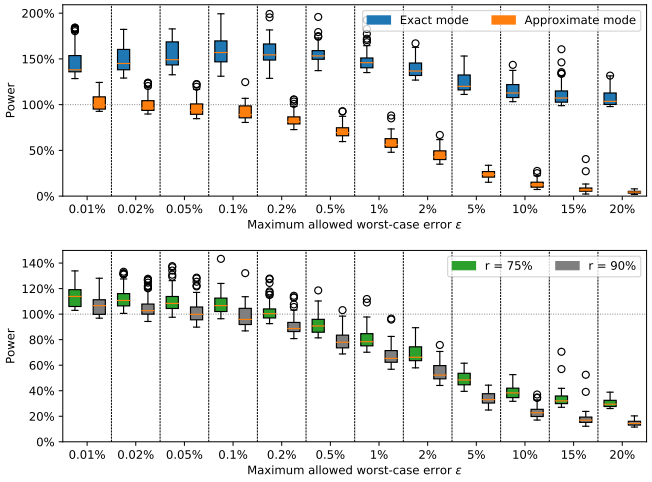


Fig. 6. Power consumption of evolved QCMs used a) in the static mode (exact and approximate mode is evaluated separately) and b) in dynamic mode where the ratio between approximate and exact mode is $r = 75\%$ and $r = 90\%$. The results are given separately for each level of WCE.

Fig. 6 shows how the power consumption depends on the circuit mode and the maximum allowed WCE. While the power follows the expected trend for the approximate mode, the exact mode exhibits undesired variability and the QCMs do not show any benefit. When we evaluate the QCMs in the intended dynamic scenario (see the bottom part of Fig. 6 showing the average power for various ratios between duration of the approximate and exact mode), there are QCMs that can achieve nearly 10% power reduction even when a negligible error is introduced (see the situation for WCE=0.05%). As evident, the power decreases with the increasing error. In addition to that, spending more time in the approximate mode leads to lower average power consumption. For 1% error and 90% ratio in favor of approximate mode, for example, the best evolved QCM requires only 60% of the power of the exact 8-bit multiplier.

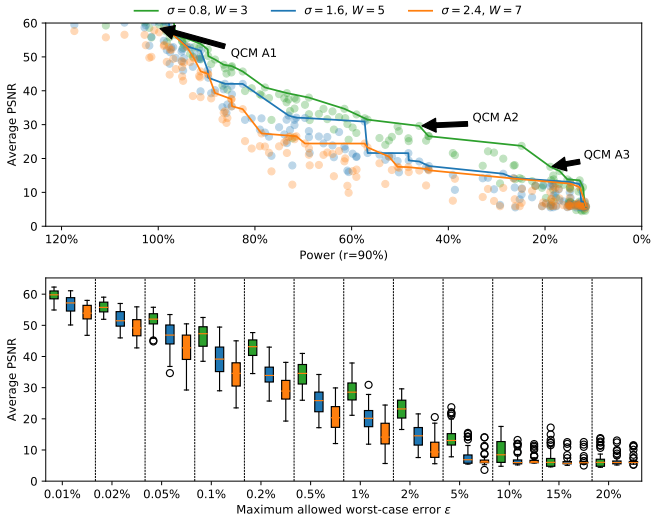


Fig. 7. Average PSNR representing the quality of filtering w.r.t power consumption (top) and WCE (bottom) of QCMs employed in Gaussian image filters. Local Pareto fronts are depicted using solid lines for three analyzed cases.

C. QCM in Image Filtering

The proposed QCMs are evaluated as building blocks of image filters developed for Gaussian noise elimination. Each QCM is used to create three Gaussian noise filters with kernels $W \times W$ pixels, where $W = 3$ (standard deviation $\sigma = 0.8$), 5 ($\sigma = 1.6$) and 7 ($\sigma = 2.4$). For example, 49 QCMs are needed for composing a filter with the 7x7 kernel. In total, over 4000 different image filters were created and then evaluated using a set of 25 benchmark images with the size of 384 x 256 pixels[1]. The quality of filtering was determined

[1]http://www.fit.vutbr.cz/vasicek/imagedb

as a peak signal to noise ratio (PSNR) between the exact Gaussian filter and approximate Gaussian filter composed of QCMs configured in the approximate mode. Fig. 7 shows the average PSNR obtained by filtering of 25 images w.r.t power consumption and WCE of QCMs employed in selected Gaussian image filters in which the approximate mode is active in 90% of time. Their power consumption is given relatively to the power consumption of the exact Gaussian noise filter. The proposed filters show expected behavior — the quality of filtering is positively corelated with power consumption of individual QCMs. The best results are obtained with the 3x3 pixel kernel because larger kernels lead to the loss of detail in filtered images. For obtaining a reasonable image quality (which roughly corresponds with PSNR $>$ 30 dB) the QCMs should exhibit WCE better than 2%. This situation is also depicted in Fig. 8 in which three different filters containing QCMs A1, A2 and A3 are compared using a single image. While the difference between the output of the exact filter and the filter employing A1 is invisible, A2 leads to a small reduction of a dynamic range. The quality of filtering using QCM A3 is not acceptable. We have already seen that QCMs from [23] have significantly higher MAE in comparison with the proposed QCMs. Because MAE is highly positively correlated with WCE, DQ4 circuits are not competitive in this task.

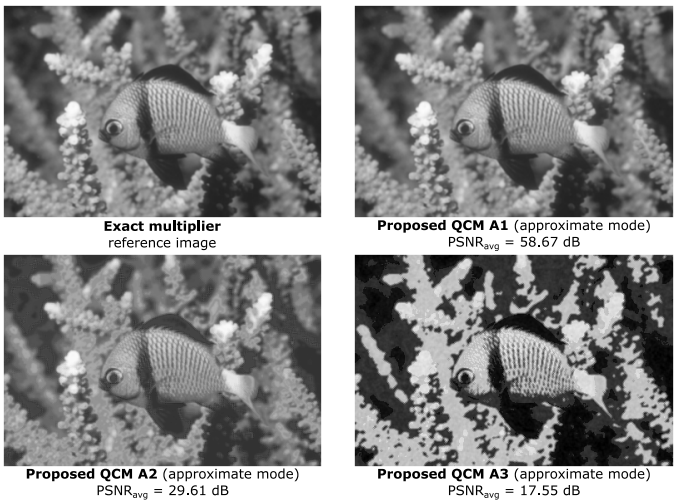


Fig. 8. The images obtained using exact and three approximate Gaussian image filters with the 3x3 kernel and $\sigma = 0.8$. The approximate filters are composed of the QCMs whose parameters are given in Table II.

VII. Conclusion

We developed a new method for the design of quality configurable circuits. The method is based on CGP in which the exact and approximate versions of the circuit are evolved together. The method was experimentally evaluated in the design of QCMs operating in two modes. A detailed analysis revealed that evolved QCMs provide better parameters than the state of the art QCMs. Proposed QCMs were employed in quality configurable Gaussian noise filters in which the approximate mode of operation is dominating. We showed that a small loss in the quality of filtering leads to a significant power consumption reduction. Our future work will be devoted to applying the proposed methodology in the design of other quality configurable circuits.

Acknowledgment

This work was supported by The Ministry of Education, Youth and Sports of the Czech Republic INTER-COST project LTC18053.

References

[1] S. Mittal, "A survey of techniques for approximate computing," *ACM Comput. Surv.*, vol. 48, no. 4, p. 62:162:33, 2016.
[2] Q. Xu, T. Mytkowicz, and N. S. Kim, "Approximate computing: A survey," *IEEE Design Test*, vol. 33, no. 1, pp. 8–22, Feb 2016.
[3] V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Analysis and characterization of inherent application resilience for approximate computing," in *DAC'13*, 2013.
[4] M. Shafique, W. Ahmad, R. Hafiz, and J. Henkel, "A low latency generic accuracy configurable adder," in *DAC'15*, 2015.
[5] T. Yang, T. Ukezono, and T. Sato, "A low-power high-speed accuracy-controllable approximate multiplier design," in *ASP-DAC'18*, 2018.
[6] P. Kulkarni, P. Gupta, and M. D. Ercegovac, "Trading accuracy for power in a multiplier architecture," *J. Low Power Electronics*, vol. 7, no. 4, pp. 490–501, 2011.
[7] S. Venkataramani, A. Sabne *et al.*, "SALSA: systematic logic synthesis of approximate circuits," in *DAC'12*, 2012, pp. 796–801.
[8] D. Shin and S. K. Gupta, "Approximate logic synthesis for error tolerant applications," in *DATE'10*, 2010, pp. 957–960.
[9] S. Venkataramani, K. Roy, and A. Raghunathan, "Substitute-and-simplify: a unified design paradigm for approximate and quality configurable circuits," in *DATE'13*, 2013, pp. 1367–1372.
[10] L. Sekanina and Z. Vasicek, "Approximate circuits by means of evolvable hardware," in *2013 IEEE Int. Conf. on Evolvable Systems*, ser. SSCI'13, 2013, pp. 21–28.
[11] A. Lingamneni, C. Enz *et al.*, "Energy parsimonious circuit design through probabilistic pruning," in *DATE'11*, 2011.
[12] Z. Vasicek and L. Sekanina, "Evolutionary design of approximate multipliers under different error metrics," in *DDECS'14*, 2014.
[13] ——, "Evolutionary approach to approximate digital circuits design," *IEEE Trans. Evol. Comput*, vol. 19, no. 3, pp. 432–444, 2015.
[14] A. Lotfi, A. Rahimi, A. Yazdanbakhsh, H. Esmaeilzadeh, and R. K. Gupta, "Grater: An approximation workflow for exploiting data-level parallelism in FPGA acceleration," in *DATE'2016*, 2016, pp. 1279–1284.
[15] K. Nepal, Y. Li *et al.*, "ABACUS: A technique for automated behavioral synthesis of approximate computing circuits," in *DATE'14*, 2014.
[16] R. Hrbacek, V. Mrazek, and Z. Vasicek, "Automatic design of approximate circuits by means of multi-objective evolutionary algorithms," in *DTIS'16*, 2016.
[17] H. Jiang, C. Liu, N. Maheshwari, F. Lombardi, and J. Han, "A comparative evaluation of approximate multipliers," in *IEEE/ACM Int. Symp. on Nanoscale Architectures*, 2016.
[18] J. F. Miller, *Cartesian Genetic Programming*. Springer-Verlag, 2011.
[19] Z. Vasicek, "Relaxed equivalence checking: a new challenge in logic synthesis," in *2017 IEEE Int. Symp. Design and Diagnostics of Electronic Circuits Systems (DDECS)*, 2017, pp. 1–6.
[20] M. Ceska, J. Matyas, V. Mrazek, L. Sekanina, Z. Vasicek, and T. Vojnar, "Approximating complex arithmetic circuits with formal error guarantees: 32-bit multipliers accomplished," in *ICCAD'17*, 2017.
[21] M. Shafique, R. Hafiz, S. Rehman *et al.*, "Invited: Cross-layer approximate computing: From logic to architectures," in *DAC'16*, 2016.
[22] S. Jain, S. Venkataramani, and A. Raghunathan, "Approximation through logic isolation for the design of quality configurable circuits," in *DATE'16*, 2016.
[23] O. Akbari, M. Kamal *et al.*, "Dual-quality 4:2 compressors for utilizing in dynamic accuracy configurable multipliers," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 4, pp. 1352–1361, April 2017.