



Cost-effective synthesis of QCA logic circuit using genetic algorithm

Amit Kumar Pramanik¹ · Mahabub Hasan Mahalat² · Jayanta Pal³ · Seyed-Sajad Ahmadpour⁴  · Bibhash Sen²

Accepted: 7 August 2022 / Published online: 13 September 2022

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract

Quantum-dot cellular automata (QCA) is a field coupling nano-technology that has drawn significant attention for its low power consumption, low area overhead, and achieving a high speed over the CMOS technology. Majority Voter (MV) and QCA Inverter (INV) are the primitive logic in QCA for implementing any QCA circuit. The performance and cost of a QCA circuit directly depend on the number of QCA primitives and their interconnections. Their optimization plays a crucial role in optimizing the QCA logic circuit synthesis. None of the previous works considered elitism in GA, all the optimization objectives (MV, INV and Level), and the redundancy elimination approach. These profound issues lead us to propose a new methodology based on Genetic algorithm (GA) for the cost-effective synthesis of the QCA circuit of the multi-output boolean functions with an arbitrary number of inputs. The proposed method reduces the delay and gate count, where the worst-case delay is minimized in terms of the level. This methodology adapts elitism to preserve the best solutions throughout the intermediate generations. Here, MV, INV, and levels are optimized according to their relative cost factor in a QCA circuit. Moreover, new methodologies are proposed to create the initial population, maintain the variations, and eliminate redundant gates. Simulation results endorse the superiority of the proposed method.

Keywords Genetic algorithm · Circuit synthesis · Quantum-dot cellular automata · Circuit optimization · Elitism

✉ Seyed-Sajad Ahmadpour
sajadahmadpour@gmail.com

Extended author information available on the last page of the article

1 Introduction

Present Complementary Metal Oxide Semiconductor (CMOS) technology approaching the saturation level in terms of power efficiency, feature size [27]. QCA technology receives tremendous attention as a worthwhile substitute to CMOS technology owing to its area efficiency and a minimal amount of power consumption [27]. QCA cells are the primary element of any QCA circuit. No current flow occurs in the QCA circuit during signal transition and propagation because of the interaction between the cells through Coulombic repulsion. The fundamental gates in QCA technology are the majority voters (MV) and inverter (INV), which are used to synthesize any complex QCA logic circuit [26, 27]. In the QCA circuit, the placement of cells to construct the logic primitives are significant in the cost-effective synthesis of the logic circuit. Therefore, reduction of logic primitives, as well as levels (discussed in Sect. 4.2), plays a crucial role in designing optimal QCA logic circuits.

Automatic cell layout and its optimization is a well-studied topic on conventional CMOS logic circuits [17, 42, 43]. A new design of multiplexer, decoder and ALU were proposed in [3–5]. The AND, OR logic-based minimal form of the sum of product (SOP) expressions or a product of sum (POS) expressions are generally utilized for the implementation of the conventional logic circuits. Unlike CMOS, in QCA, primitive boolean logic (AND/OR) cannot be generated directly; instead, 3-input MV is used to create such primitive boolean logic. Thus, it is tough to directly exploit the minimization of POS or SOP form of expression to realize the circuit in QCA.

Genetic Algorithm (GA) is one of the heuristic techniques [11, 38, 45], grounded on natural evolution theory and well recognized for the discovery of NP-hard problem resolutions [10, 21, 33]. Previously, a few mentioned works (details discussed in Sect. 3) have considered GA in the direction of the automatic generation of the optimal layout of the QCA circuit [8, 19, 20, 34, 40]. Nevertheless, each has its limitations considering the number of input variables, number of outputs, and optimization objective. Moreover, to our knowledge, none of the previous GA-based optimization methods used elitism explicitly or considered the priority of the optimization objectives to generate the QCA circuit. Elitism preserves the best solutions which can be effectively utilized in GA to improve the optimal result [1, 6, 24]. The number of majority voters (MV) and inverter (INV) have been accounted for reducing gate count, whereas level is considered to minimize the worst-case delay (represented in the number of clock zones) of a QCA circuit. However, level, MV, and INV have different cost factors in a QCA circuit. Hence, their relative priority should be considered to obtain the optimal QCA circuit [25, 37].

Motivated by the above factors, we proposed a new methodology to synthesize a cost-efficient QCA circuit using GA. In a practical scenario, the use of multi-output circuits is more general; thus, its main focus is to optimize the multi-output circuit. The salient features of the proposed methodology are summarized below.

- A priority-driven calculation approach for fitness function is proposed for the affective dimension of candidate solutions. Priority is allotted to distinct objectives (i.e., level, MV, INV), bestowing their relative cost factor in the circuit.
- The proposed methodology adopted elitism in the genetic algorithm (GA) to preserve the best solutions throughout the generations. Further, new methods are introduced to create the initial population and maintain the variations. Also, a redundancy elimination method is proposed to eliminate the possibility of any redundant gates from the final solution.
- The optimal QCA circuit for a multi-output function is attained because of the relative fitness value of the individual output functions. Previous ($i - 1$) output functions are taken into consideration for the calculation of the fitness value of the i^{th} output function.
- Some standard multi-output boolean functions are considered to check the efficiency of the proposed method. The results establish that the proposed methodology expresses substantial improvement in minimizing the level and gate count over the existing heuristic techniques.

The rest portion of the paper is structured as follows. Section 2 contains the related contextual resources, including the QCA and genetic algorithm basics required to understand the proposed methodology. Section 3 consists of a brief discussion of the related works. Details of the suggested or proposed methodology are illustrated in Sect. 4. The next section (Sect. 5) contains the simulation result related discussions and corresponding analysis. Finally, the the conclusion of the paper is drawn in Sect. 6.

2 Background materials

2.1 QCA

The vital element of QCA technology is the QCA cell. A QCA cell works based on the Coulombic repulsion between electrons [27]. It (i.e., QCA cell) consists of two (2) electrons and four (4) quantum dots confined inside it. The four dots inside the cell are placed in the four corner positions [32]. When a cell is in its most stable state, the electrons transfer in the corners because of Coulombic repulsion. In this fashion, a QCA cell feasibly expresses two probable states in a steady position. Logically represented by $P = -1$ and $P = +1$, two states are considered as equivalent to binary logic 0 and binary logic 1, respectively [31]. A QCA cell and possible states concerning electron positions are shown in Fig. 1(a) i–ii. Signal transition and propagation between the QCA cells occur without charging and discharging of voltage. A QCA gate is constructed by combining the QCA cells. Basic QCA gates are the majority voter gate (MV) and inverter (INV). The functionality of a three-input MV can be mathematically expressed by Eq. 1.

$$F = PQ + QR + RP \quad (1)$$

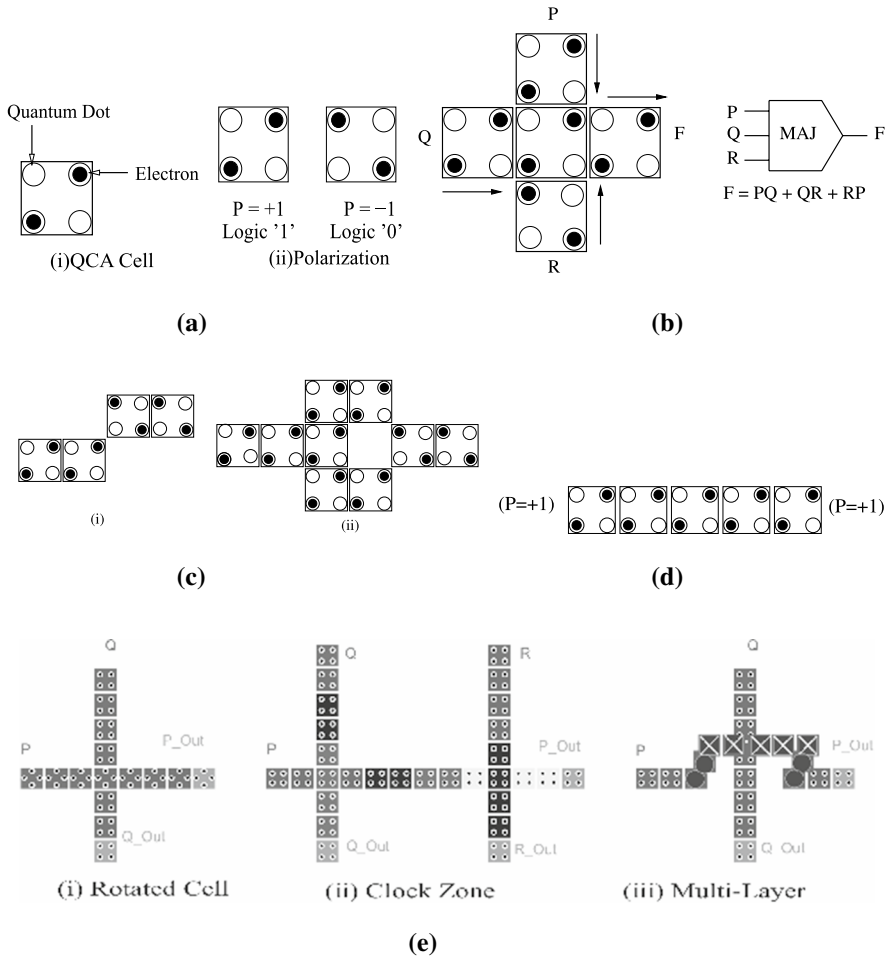


Fig. 1 QCA fundamental devices

Where P, Q, R are the inputs, and F is the output. Binary logic '1' is produced as the output of a MV when 2 out of 3 of its inputs are logic '1' and binary logic '0' for other cases [41]. Fixation of one input to a fixed polarized value of 1 or 0 leads MV to act as AND gate or the OR gate, respectively. The structure of an MV is shown in Fig. 1(b). a single-layer 5-input majority gate was proposed in [2]. Inverter (INV) is another basic gate in QCA. Logically, the INV is equivalent to the conventional NOT gate. It takes only one input and produces the complement of the input as the output. Different alternative cell orientations for QCA INV gate realization are shown in Fig. 1(c) i–ii. In a QCA circuit, the connection between the QCA gates is provided by using the QCA wire [26, 27]. A series of cells placed aside to implement a wire is shown in Fig. 1(d). Any sophisticated circuit in nanotechnology-based QCA technology can be realized by using MV and INV gates, where the

connections are made with the QCA wires [39]. But interconnecting the wires is very challenging in the fabrication step [28]. The rotated cell-based wire crossing approach uses two different types of QCA cells for crossing [7], shown in Fig. 1(e) i. Clock zone-based wire crossing shown in Fig. 1(e) ii and multi-layer based wire crossing approach uses more than one cell layer, represented in Fig. 1(e) iii. In this regard, an alternative wire crossing method based on XOR gate is reported in [14, 22].

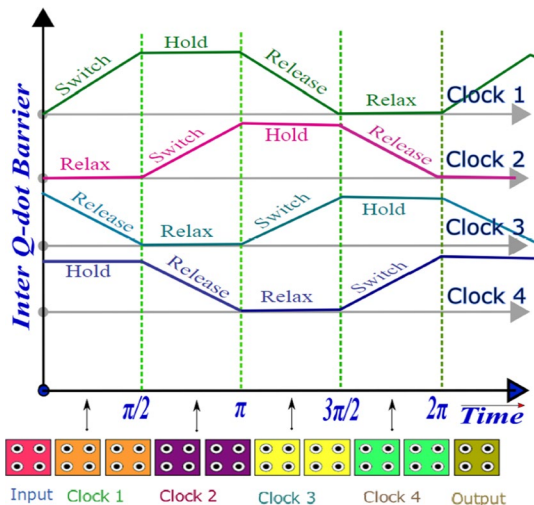
The direction and timing of the signal flow in a QCA circuit are controlled through the QCA clocking [18, 26]. QCA clocking has four (4) phases, switch phase, hold phase, release phase and relax phase. In the first phase (called as switch phase), QCA cell begins with a very low potential energy in the barrier between the dots, and the barrier gradually starts increasing.

In the hold phase QCA cell holds the barrier high. In the next phase (i.e., release phase), the strength of the barrier within the dots starts diminishing, and finally, in the last phase (termed as relax phase), the cell resets to the un-polarized state. A circuit in QCA technology is virtually split into different clock zones. A particular clock drives all the cells in the same clock zone. As a result, the more utilization of clock zones directly proportionate to the increase in the circuit delay. Figure 2 shows the phases of QCA clocking, and also the operation of a QCA wire in several clock zones.

2.2 Genetic algorithm

Genetic Algorithm (GA) is a heuristic method that is inspired by Darwin’s theory of natural evolution [10, 21, 33]. The significant steps of standard GA execution are the creation of the initial population, fitness calculation, crossover, mutation, and creation of the next-generation population. GA starts its execution by creating an initial population set of random solutions. This randomly selected population set is

Fig. 2 QCA clocking



popularly termed as a chromosome. A fitness function is used in GA to calculate the fitness value of the individual chromosome. The fitness value indicates the feasibility and defines the sustainability of a chromosome. The population set for the next generation is formed by way of the selection, crossover, followed by mutation process. (a) The selection operation is applied on parent chromosomes to select some of them based on a given criterion for the crossover. (b) The crossover function then generates new offspring chromosomes from the parent chromosomes to be used in the next generation. (c) The mutation operation is applied to preserve discrepancies in the population pool. In mutation, some portion of a chromosome is randomly altered with a specified probability to introduce new characteristics.

GA reiterates these operations several times until the targeted goal is realized or the iteration reaches the maximum step. In elitism-based GA, few chromosomes with the most suitable value are chosen to be the elite chromosomes and passed to the next generation without any alteration, which ensures the optimum result [6]. A Multi-objective genetic algorithm considers more than one objective to be optimized. Several multi-objective based genetic algorithms have been reported, suitable for different applications [12, 13, 24].

3 Related works

In previous literature, various non-GA-based methodologies were introduced to synthesize an optimal QCA circuit corresponding to a Boolean function [25, 37, 44, 46]. However, GA based optimization technique for the QCA circuit was initially introduced by [8]. A tree structure was used in this case to represent a chromosome. The basic gates like the majority voter (MV) gate or inverter (INV) gate have been represented by the internal nodes, whereas any input variables or constants were denoted by leaf nodes. This GA-based methodology used node count for measuring the fitness value. This paper [8] itself declares, due to the dissimilarities in the number of inputs in the majority and inverter, the mutation becomes complicated. New chromosomes were prepared randomly and inserted in the population set, replacing the chromosomes with the worst fitness. This methodology can be useful in reducing the number of gates only and limiting the single output boolean function. In [23], a method was developed to reduce fixed external inputs. In [35], a GA-based reduction technique is proposed taking advantage of the majority gate in both three input and five input forms in the synthesis of the QCA circuit. Also, it incorporates the mutation process, as proposed in [8]. A few methodologies of applying GA in connection with single output functional gate can be reported in [30]. In another approach, a hamming oracle-oriented assessment of evolutionary search methods is reported [29]. In this case, GA was considered without mutation or recombination. At the same time, the concept of GA was found to be applied in QCA for the reduction of wire crossing constraints in [16, 36].

A GA-based methodology was proposed to synthesize optimal QCA circuits, which can be deployed on 2-output boolean functions in [19]. It was further extended for any multi-output Boolean functions in [20]. The investigation suggests that for the case of multi-output functions separately, considering the discrete output

function may not focus on an optimal solution [19, 20]. However, the methodology in [20] can be applied to multi-output Boolean functions, which can decrease the gates count only in a QCA circuit. Utilizing multi-objective genetic programming in the synthesis of optimal QCA circuit of any multi-output boolean functions is reported in [34]. It was inevitable that considering only gate count or worst-case delay may not lead to the optimal solution. Thus, along with number of gates, the methodology to reduce worst-case delay in terms of the height of the chromosome tree (level) is also required. However, consideration of the relative priority of the optimization objectives was missing in [34]. Another technique to generate optimal QCA circuits using an evolutionary algorithm was proposed in [40]. This technique considered the number MV and level together as the optimization objective but ignored the number of INV. The stable representation of the INV contains a considerable amount of cells. Hence, it has a significant impact on circuit cell count and delay, which makes it worth considering. In contradiction with [19, 20], it did not consider the other output functions for optimizing a specific output function.

It is evident from the above discussion that some critical issues have not been explored, primarily focusing on multi-output functions. Firstly, none of the previous works considered elitism in GA, even though elitism improves the optimization performance [1, 6, 24]. Second, most of the works did not consider all the optimization objectives such as MV, INV, Level, instead focused on gate and latency reduction. Third, the consideration of relative priority of the optimization objectives, even though it was analyzed and used in the non-GA-based QCA optimization techniques [25, 37] and redundancy elimination approach. The synthesis process may generate redundant gates, but previous works did not consider the redundancy elimination approach. This literature discussion and profound issues lead us to propose a new GA-based QCA circuit optimization method.

4 Proposed method

As discussed earlier, the previous works did not consider either elitism or all the optimization objectives. The priority of the optimization objectives was also ignored in synthesizing the optimal QCA circuit. The proposed work has considered all these criteria to generate the optimal QCA circuit. The proposed methodology minimizes the number of levels, Majority Voter (MV) and inverter (INV), in a QCA circuit by considering their relative priority. It takes an arbitrary multi-output Boolean function as the input, where each output is viewed as a discrete function. A full Adder can be represented by two functions, the sum and the carry. The output functions are randomly ranked from 1 to the last (1, 2, 3, ..., N).

The proposed methodology starts with the generation of the initial population set of random chromosomes, where each chromosome represents a logical QCA circuit, as illustrated in the Sect. 4.1. The fitness value of each chromosome is then calculated using the fitness function defined in Sect. 4.2. A group of chromosomes having the superior fitness value of all is selected as elite chromosomes (described in Sect. 4.3) and passed into the next generation without any alteration. For the crossover, parent chromosomes are selected using tournament selection. A modified

one-point crossover function that suits the proposed methodology is applied to generate the crossover children (Sect. 4.4). After selecting the elite chromosomes and the generation of the crossover children, a new set of chromosomes is generated and added to the next-generation population pool to keep the population size constant. It is to be noted here that the normal mutation function cannot guarantee a logical QCA circuit as an output, so it cannot be applied directly. Thus, the proposed methodology considers the addition of a new set of chromosomes with the selected population pool instead of actual mutation to maintain the variation (details in Sect. 4.5). The combination of elite chromosomes, crossover children, and the new chromosome set creates the next-generation population pool. The steps involved, from the calculation of fitness value to the creation of the next-generation population pool, are repeated several times until the specified termination condition is not reached. In the case of multi-output functions, as discussed above, the process is performed for each of the output functions.

Two different fitness functions, namely 'fitness1' and 'fitness2', are introduced for the calculation of the fitness value. 'fitness1' is used for the initial output function (i.e., output with rank 1) whereas 'fitness2' is used for the rest of the output functions (output with other ranks than 1). After the termination of the whole process, it returns the best-combined chromosome set, which satisfies all the outputs. The overall glimpse of the proposed methodology is represented in a flowchart diagram in Fig. 3.

4.1 Creation of initial population

The concept of the tree structure, as reported in [8], is adopted for the representation of the QCA logic function. The internal node represents the basic gate like MV or INV, whereas the leaf node represents input variables or the constants. The traversal of the presented tree in pre-order fashion is employed here to store the corresponding QCA logic function as a chromosome. The constraints of generating a chromosome corresponding to a valid QCA circuit have not been addressed as per the literature. This paper proposes a method for preparing chromosomes corresponding to a logically correct QCA circuit with the generation of the initial population. The QCA basic gates (MV & INV), input variables, and constants are mapped with an integer value to represent a chromosome with an integer string. For example, if we have three input variables A, B & C, then the basic gates will be assigned an integer value as MV is assigned with 7, INV is assigned a value of 6. The following values are assigned to input variables like variable 'A' assigned with 5, 'B' with value 4 & 'C' with 3, and finally the binary logic values (constants) logic '1' will be assigned as 2, logic '0' assigned the value 1. A random integer collection is populated with the assigned values to generate a chromosome, and the following rules are followed in this regard, as explained below:

1. Randomly selects any number among the assigned values to the gates and input variables.

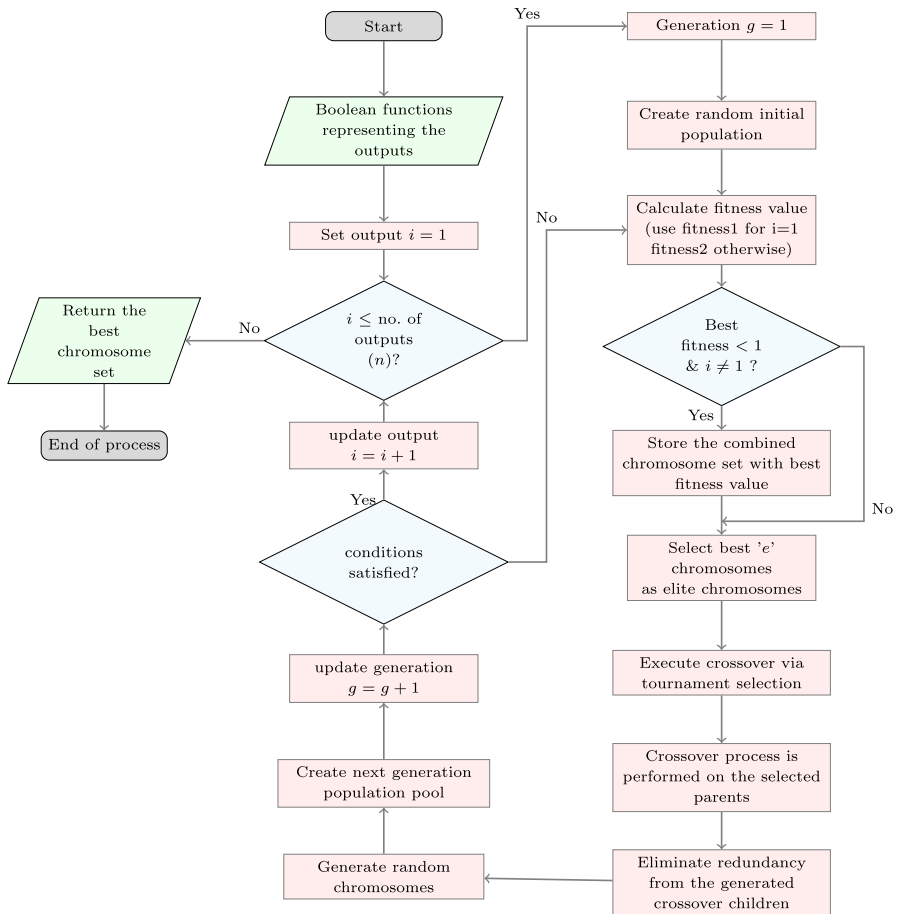


Fig. 3 Flow chart for the proposed methodology

- Some more value(s) among the mapped integers are appended depending on the selected number in the first step. If the previously selected number corresponds to MV, then three (3) more numbers are appended as MV has three inputs. Similarly, one (1) more number is appended for INV, as it takes only one input. However, no extra number gets appended for variables or constants. The number(s) to be appended is/are also selected randomly from the mapped integers.
- Step 2 is recursively applied for each appended number. If the string length exceeds the maximum specified length l (l is 2^n for n inputs function), then it will be rejected, and a new string is regenerated using the same procedure in steps 1 and 2 (l is assumed to be the maximum probable length of the string/chromosome).

The generated integer string represents a chromosome corresponding to a logically valid QCA circuit. Logically valid, in this case, indicates a circuit

representation with proper output meaning. The tree and chromosome representation of the QCA logic function $M(M(B', C, O)A, 1)$ is shown in Fig. 4. For an initial population of size n , n chromosomes are generated using the aforementioned technique and added with the initial population set. The structurally similar chromosomes in the initial population set are replaced with new chromosomes for maintaining diversity. Here, the term structurally similar means the chromosomes have similar structure representation in the tree structure.

4.2 Fitness value calculation

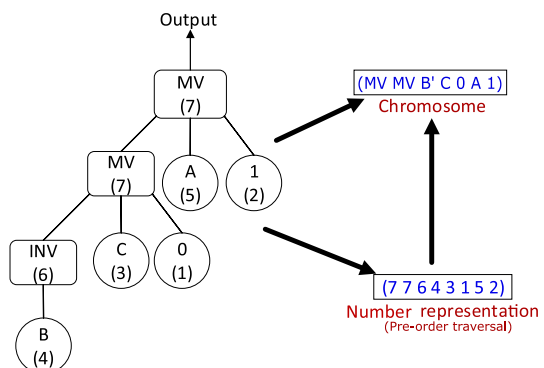
For calculating the fitness value of a particular chromosome, the respective output function (i.e., of a multi-output function) is represented in its canonical sum of product (SOP) expression. As mentioned earlier, we have used two different fitness functions to compute the fitness value efficiently. The fitness function 'fitness1' is applied for the initial output function (i.e., output function with rank 1), and 'fitness2' is used for all the remaining outputs. Details of the fitness functions are described below:

4.2.1 Fitness function (fitness1)

We have formulated the fitness function as a minimization problem; it can also be altered to a maximization problem without loss of generality. The algorithm for 'fitness1' is represented in Algorithm 1.

The initial goal of the fitness function is to estimate the suitability of a chromosome for synthesizing the output function. So, the fitness value of a chromosome is calculated as the ratio of number of total min-terms and number of min-term matches with the output function. The value one as the fitness value (i.e., $\text{temp_fit}' = 1$), points to the fact that the chromosome represents the output function.

Fig. 4 Representation of the QCA function $M(M(B', C, O)A, 1)$



Algorithm 1: Pseudo code for fitness1

Input: Chromosome
Output: Fitness Value
 1 $temp_fit \leftarrow \frac{\# \text{ Possible inputs}}{\# \text{ Outputs matches with target func}}$
 2 **if** $temp_fit == 1$ **then**
 3 $fitness \leftarrow 1 - \left(\frac{1}{\text{No. of gates}} + \frac{1}{\text{No. of levels}} \right)$
 4 **else**
 5 $fitness \leftarrow temp_fit$
 6 **return** $fitness$

If the chromosome successfully represents the output function (i.e., $temp_fit = 1$), then the fitness value is reevaluated to reflect the worst-case delay and the number of gates of the chromosome. The worst-case delay of a QCA circuit is directly associated with the number of required clock cycles. Here, the required clock cycles are computed in terms of the levels in the circuit. The level is defined as the number of MVs in the longest path from root to leaf node in the chromosome tree. Unlike INV, the MV gate has 3 inputs that need proper synchronization of the inputs to generate the correct output. INVs are considered for gate count but not necessarily for measuring the level.

Though the number of gates can be reduced by reducing the number of MVs and INVs in the chromosome, the area and worst-case delay are directly proportional to the level. Therefore, it is implied that the level is the most crucial factor in minimizing the circuit instead of only the gate count. So, the fitness function is assigned more priority to level implicitly than the gate count. As the level is evaluated in terms of the number of MV gates along the longest path of a QCA circuit, the level cannot be more than the number of MVs in a chromosome.

Algorithm 2: Pseudo code for fitness2

```

Input: Chromosome
Output: Fitness Value
1  $temp\_fit \leftarrow \frac{\text{No. of possible inputs}}{\text{No. of outputs matches with target function}}$ 
2 if  $temp\_fit == 1$  then
3   if fitness value is found in fitness_stored then
4      $fitness \leftarrow fitness\_stored(\text{Chromosome})$ 
5   else
6      $gates\_final \leftarrow \infty$ 
7      $level\_final \leftarrow \infty$ 
8     for  $p = 1$  to entry count in chromosomes_stored do
9        $total\_mv \leftarrow \text{unique MVs in chromosomes\_stored}(p) - \text{common MVs with current}$ 
10       $chromosome;$ 
11       $total\_inv \leftarrow \text{unique INVs in chromosomes\_stored}(p) - \text{common INVs with current}$ 
12       $chromosome;$ 
13       $total\_gates \leftarrow ((total\_inv/3) + total\_mv);$ 
14       $max\_level \leftarrow \max(\text{levels in chromosomes\_stored}(p), \text{levels in current chromosome});$ 
15      if  $(max\_level \leq level\_final)$  OR  $(max\_level < level\_final)$  AND
16       $(total\_gates < gates\_final)$  then
17         $gates\_final \leftarrow total\_gates$ 
18         $level\_final \leftarrow max\_level$ 
19         $final\_position \leftarrow p$ 
20      Common logic of current chromosome and  $chromosomes\_stored(final\_position)$  be
21      replaced by common logic with minimum no. of gates.
22       $level \leftarrow \max(\text{levels in chromosomes\_stored}(final\_position), \text{levels in current}$ 
23       $chromosome)$ 
24       $no\_of\_gates \leftarrow \text{unique gate count in chromosomes\_stored}(final\_position) + \text{gate count}$ 
25       $\text{in current chromosome} - \text{common gate count in current chromosome}$ 
26       $fitness \leftarrow 1 - (\frac{1}{no\_of\_gates} + \frac{1}{level})$ 
27       $fitness\_stored(\text{Chromosome}) \leftarrow fitness$ 
28 else
29    $fitness \leftarrow temp\_fit$ 
30 return fitness

```

In the case of a multi-output function, optimizing the individual output functions separately may not lead to the optimal solution for the whole function. Thus, while optimizing the *i* th output function previous output functions (i.e., 1 to (*ith* - 1)) should also be considered for evaluation. Initially, only ‘fitness1’ deals with a single output function as no other output functions are available. For other output functions *i* (i.e {2, 3,..., *n*}), the fitness function ‘fitness2’ will be used.

4.2.2 Calculation of fitness2

For the output functions, the fitness function is slightly altered other than the first output. The algorithm for ‘fitness2’ is represented in Algorithm 2.

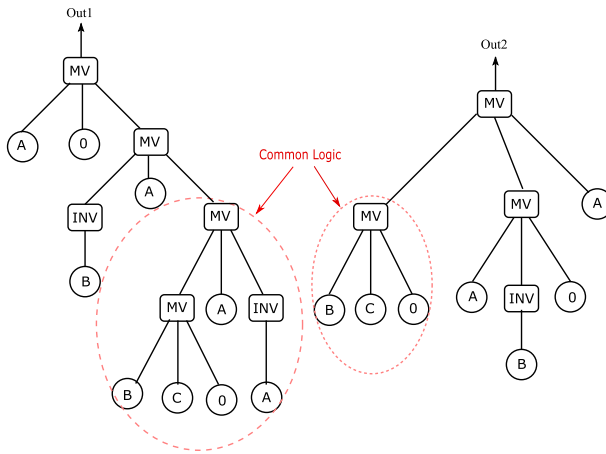
To synthesize a valid chromosome corresponding to the output function, the same steps as ‘fitness1’ are followed. The algorithm ‘fitness2’ differs from ‘fitness1’ in the next steps, where the previous output(s) is also considered to optimize the current output function. Throughout the overall process, a variable ‘fitness_stored’ is

maintained for storing the fitness of chromosomes having a lesser fitness value than 1. In the future, if it is the case when the same chromosome is used, ‘*fitness_stored*’ returns its fitness value. The computation complexity can be reduced using this strategy. If the present chromosome results in a valid circuit (e.g., ‘*temp_fit*’ = 1) and the corresponding fitness value does not exist in the ‘*fitness_stored*’, then only the steps for calculating its fitness value are executed. Matrix ‘*chromosomes_stored*’ is used to store the chromosome produced in the preceding outputs. Suppose the rank is i for the present output function. In that case, each entry in respect to the row of the ‘*chromosomes_stored*’ keeps a chromosome set till $(i - 1)$ th output starting from 1, number of unique majority, and inverter gates in the chromosome set and the maximum level. The i th output function is optimized, with respect to the previous $(i - 1)$ outputs. Any common logic between the output functions can be realized by using common gates. For selecting a chromosome set from the previous $(i - 1)$ outputs to combine with the current chromosome of i th output, two deciding factors are used. The deciding factors are named as ‘*total_gates*’ and ‘*max_level*’.

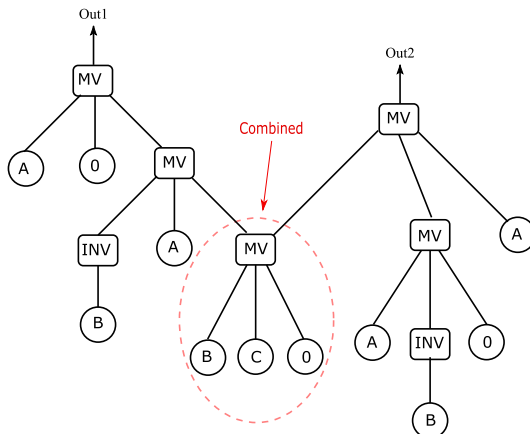
The variable ‘*total_gates*’ is calculated in terms of ‘*total_mv*’ and ‘*total_inv*’. The value ‘*total_mv*’ is determined by subtracting the number of common *MV* (i.e., between the i th output chromosome and the chromosomes of $(i - 1)$ outputs) from the number of unique *MV* in the chromosome set of $(i - 1)$ outputs and similarly ‘*total_inv*’ is determined with *INVs*. The value ‘*total_gates*’ is then calculated as the in total addition of one-third of the ‘*total_inv*’ with the ‘*total_mv*’. Here, *INV* has given one-third priority that of the *MV* because an *MV* receives three inputs to generate a single output and to synchronize, as discussed in [25].

The next factor, ‘*max_level*’ is calculated as the maximum level between the current chromosome (i.e., the chromosome of i th output) and the chromosome set from $(i - 1)$ outputs. All the individual output circuits together represent a single multi-output QCA circuit. Therefore every individual circuit should operate under the common clock. As a result, the worst-case delay of the multi-output circuit relies on the worst-case delay (in terms of a clock cycle) amongst the individual output functions. So maximum level (i.e., ‘*max_level*’) is considered to measure the delay in the worst-case situation, with the most priority when selecting a chromosome set from previous $(i - 1)$ outputs. The chromosome set from $(i - 1)$ outputs that minimize the maximum level (i.e., ‘*max_level*’) is chosen first. In the case of a tie among the chromosome sets, ‘*total_gates*’ is used to resolve the same. The minimum value of ‘*total_gates*’ is selected in this scenario. Once a chromosome set of $(i - 1)$ outputs is selected, the common logic between the current (i th) and the chosen chromosome set are substituted by the common logic with the minimum gate count. Finally, the ‘*level*’ and ‘*no_of_gates*’ are calculated in the selected chromosome set of $(i - 1)$ outputs and the current chromosome of i th output to obtain its relative fitness value. The overall performance of the minimized circuit does not completely rely on the number of cells until it increases the clock zone. However, while synthesizing the circuit, a clock-based cross-over may increase the number of clock zones (delay). Accordingly, the proposed algorithm calculates the ‘*max_level*’ parameter.

Figure 5 depicts the process of combination for a two output function, where *out1* and *out2* represent the chromosomes corresponding to the two output functions. The circled part in *out1* and *out2* depict the common logic,



(a) Chromosomes before combination



(b) Chromosomes post combination

Fig. 5 The process of combination for a two output function

$M(A, A', M(B, C, 0)) \implies (BC)$ also $M(B, C, 0) \implies (BC)$. But $M(B, C, 0)$ has less number of gates than $M(A, A', M(B, C, 0))$ so, *out1* and *out2* are combined by connecting both with $M(B, C, 0)$.

4.3 Chromosome selection

The GA approach based on elitism is applicable in the case of preserving the chromosomes with the most significant fitness value. The fitness value is considered for the sorting of the chromosomes. A few predetermined numbers of chromosomes that have better fitness from the prepared sorted list are propagated to the succeeding generation without modification. For the next phase (i.e., crossover), the selection of

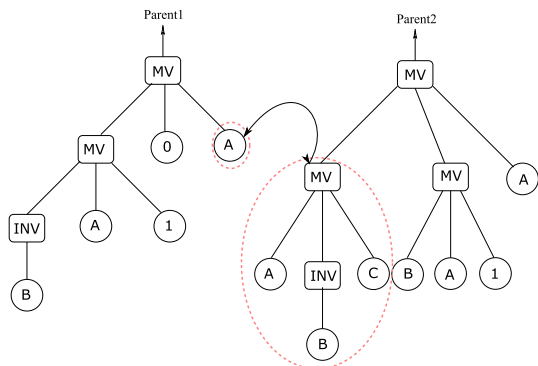
parent chromosomes is executed by the tournament selection method. The number of chromosomes as identical as the tournament size is chosen randomly. Among the selected chromosomes, the chromosome is finally picked up, having the best fitness of all. The execution of the tournament selection method is iterated until the fulfillment of the parent chromosome list for the crossover.

4.4 Crossover

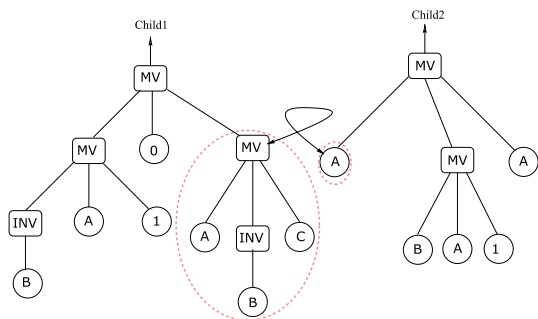
One child chromosome will be generated from a pair of parent chromosomes. Therefore, the selection of crossover children should be twice as compared to the parent chromosomes through the selection method. Two parents are randomly chosen from the selected parent chromosomes to generate a single child chromosome. The chromosomes having an unequal length make it very difficult to directly apply the conventional one-point crossover method, so a modified version of the one-point crossover is used to generate the child chromosomes.

The process of crossover operation in between the parents is illustrated by Fig. 6 (a–b). Two separate nodes from different parents (i.e., Parent1 and Parent2) are selected randomly. The chosen nodes can be internal or leaf nodes, exchanging them

Fig. 6 The process of Crossover between parents



(a) Chromosomes before crossover



(b) Chromosomes post crossover

to the corresponding sub-trees. This exchange process generates two offsprings. Lastly, the one with the best fitness value is designated as the child chromosome amongst the generated offspring.

4.5 Variation maintenance (Mutation)

As mentioned earlier, the actual mutation can not be applied here; instead, a new set of chromosomes is added with the population pool to maintain the variation only. It is due to the fact that QCA logic gates are non-uniform, and the use of MV and INV interchangeably may lead to an invalid QCA circuit [8, 29, 35]. The QCA circuit will be invalid because MV requires three inputs, whereas the INV gate requires only one input. The generation of mutation children is executed externally but not generated from the present population. The mutation children are then attached to the population pool of the next generation. A large-scale set of chromosomes is produced at random the same way as it was used to create the initial population. Then the chromosomes with the equal number of mutation children are selected randomly and redirected to the succeeding generation population pool to be considered as mutation children.

The proposed method, as given by Algorithm 3, combines two objectives into a single objective problem, which requires lesser computation than the advanced multi-objective GA. However, the proposed method might not be able to produce all the Pareto solutions, but it may produce a set of solutions with similar fitness values. If two or more solutions have the same fitness value, then the solution with the minimum level is selected. Chromosomes with the same fitness value and level, the chromosomes with fewer majority voter gates, will be chosen.

Algorithm 3: The complete pseudo code of the proposed method

```

1 Execute GA for output1 and chromosomes_stored is used to store the best chromosomes of each
  generation
2 for out = 2 to M do
3   create initial population pool of size pop
4   for gen = 1 to max_generation do
5     Compute the fitness value of individual chromosome using the function fitness2
6     if fitness value corresponds to the best chromosome < 1 then
7       append the best chromosome with the respective previous chromosomes set in the list
         temp_chromosomes_stored
8     The best chromosomes are selected as elite chromosomes (e)
9     Execute tournament selection for the selection of 2c no. of parent chromosomes
10    Crossover is applied to generate c no. of child chromosomes
11    Utilize redundancy_elimination function upon every child chromosomes generated
12    Mutation operation is performed to produce (pop - (e + c)) no. of chromosomes
13    if 1 > best fitness AND no alteration in the best fitness throughout thresh_generation
       no. of generations then
14      if out == M then
15        return the best chromosome produced and the respective previous chromosomes set
16      else
17        chromosomes_stored ← temp_chromosomes_stored
18        delete the values in temp_chromosomes_stored
19        inner for loop terminated

```

Here, one additional step called ‘redundancy elimination’ is added after crossover to eliminate any redundant offspring generated. It removes redundant inverter gates or replaces majority voters or some input with equivalent variable or constant. For example, $M(1',A,B)$ can be replaced by $M(0,A,B)$ as both the functions have similar output; later, one does not require an INV. The MV gate $M(A,A,1)$ can be replaced by a variable A. In this algorithm, the outer loop executes several times from 2nd to the last output, whereas the output function for the i th output with respect to $(i - 1)$ outputs are generated by the inner loop. The most crucial combination of the i th output chromosome with the $(i - 1)$ output chromosomes is stored in ‘chromosomes_stored’.

The variable ‘chromosomes_stored’ is modified after the computation of each output. The inner loop of the algorithm repeat until the maximum generation or no enhancement is observed in best fitness throughout the ‘thresh_generation’ number of consecutive generations after generation of the correct chromosome. Beyond the final output, the outer loop terminates and results in the combined best chromosome, satisfying all the outputs.

5 Simulation and results analysis

5.1 Simulation environment

The proposed method is implemented using MATLAB. The simulation environment and design constraints are shown in Table 1.

The proposed methodology uses 200 chromosomes and 5000 generations for population size and maximum generation, respectively. From the current population pool, 20 best chromosomes (i.e., 10% of population size) are selected as elite chromosomes. Initially, 70% of the next generation chromosomes are produced through crossover, which excludes the elite chromosomes. The rest of the chromosomes are created through the process of mutation. The tournament selection method of size 3 has been utilized to select parent chromosomes to create new offspring.

For the generation of the first justified chromosome in respect to the SOP expression, the rate of mutation is dropped by assuming a greater crossover rate of 80%. If a valid solution is produced well in advance than the maximum generation, a dynamic method is applied to conclude the GA. Periodic checking is performed after generating the first valid chromosome in respect to the SOP expression. The GA stops the execution if the best fitness value does not improve throughout 300 consecutive generations.

5.2 Realization using regular clocking

To investigate the efficiency of the method one four input , two output function (F_1) is realized using USE [9] clocking scheme and one three input , three output function (F_2) is realized using RES [15] clocking schemes. Table 2 represents the realization using USE and RES clocking schemes. The count of majority voter (MV), inverter (INV), cell count, and latency to realize function F_1 using only USE clocking scheme are 5, 2, 157, 2.25, respectively. However, the count of MV, INV, cell count, and latency to realize function F_1 using the USE clocking scheme and GA are 3, 2, 100, 1.25, respectively. Similarly, when F_2 is realized only using RES, the MV, INV, cell count, and latency count are 3, 1, 103, 1.5, respectively. But when F_2 is realized only using RES, the MV, INV, cell count, and latency count are 2, 1, 27, 0.5, respectively. So, the MV, INV, cell count, and latency for both functions are reduced after applying GA.

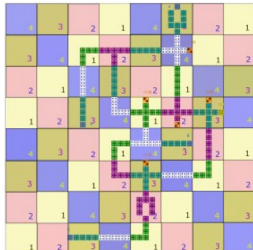
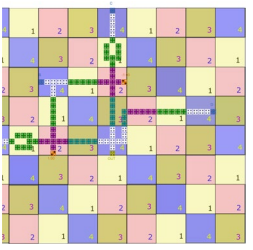
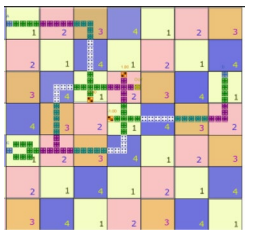
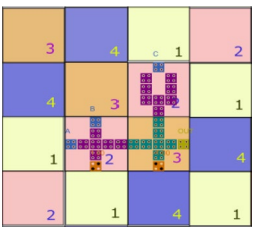
5.3 Result analysis

To scrutinize the performance of the proposed method taking into consideration the multi-output function. In this regard, a comparison is performed with the existing methods proposed in [34] and [40]. The comparison is based on the best available

Table 1 Simulation environment and design constraints

Simulation software:	MATLAB R2015a
Maximum generations:	5000
Population size:	200 chromosomes
Number of elite chromosomes:	10% of the population size
Selection method:	Tournament selection (size 3)
Crossover probability:	Initially 0.7 afterward 0.8
Mutation probability:	Not applicable

Table 2 Realization of functions using regular clocking scheme

Function	Clocking scheme	Before GA	After GA
$F_1 = \sum(m_1, m_3, m_4, m_5, m_7, m_{12}, m_{13}, m_{15})$	USE [9]		
$F_2 = \sum(m_2, m_4, m_6)$	RES [15]		

related data in the literature. The comparisons are shown based on the data provided in [34] and [40]. This work mainly focused on multi-output functions, keeping away from comparison with single output functions. The simulation results and comparisons are presented in Tables 3, 4, 5, 6 and 7 with the best-suited value found in the literature. m_0, m_1, m_2, \dots represents the min-terms of the output functions. The common parts in the circuit synthesis using the proposed method are underlined.

The overall performance is evaluated based on the total number of gates (TG) and the maximum level. Maximum level (‘Max level’) is determined as the maximum number of levels among all the individual output functions (i.e., F_1, F_2, \dots, F_n) and the total number of gates (TG) is calculated as the summation of total MV (TMV) and total INV (TINV) gates. ‘TG’ and ‘max level’ are the significant parameters to optimize a QCA circuit. The area utilization for a QCA circuit synthesis relies on the number of gates. An increase in the gate count signifies a significant increase in the area of a QCA circuit. It also involves increasing the average delay of the circuit.

In [8, 19, 20, 34], TG is considered as a comparison metric. In [40], only MV is considered instead of TG; however, we also mentioned the number of MV for comparison. As discussed in the proposed method, when multiple-output functions are combined, all the output functions have to work under the influence of the same clocking scheme. Therefore, it is the maximum clock cycle for an individual output function on which the clock cycle of the combined circuit depends. The clock cycle is measured in terms of level; thus, ‘max level’ is taken as another comparison metric.

In Tables 3 and 5, for the 3 input/2 output and 3 input/4 output function, the proposed method does not show significant improvement. However, the proposed

Table 3 Comparison for three (3) input and two (2) output function(s)

Method	functions	Circuit	NMV	NINV	Levels	CMV	CINV	TMV	TINV	TG	Max level
[40]	$p_1 = \sum m_0, m_2, m_4, m_7$	$p_1 = M(M(A, \bar{B}, C), M(1, A, C), M(1, \bar{B}, \bar{C}))$	4	3	2	1	1	$(7 - 1) = 6$	$(5 - 1) = 4$	10	2
	$p_2 = \sum m_0, m_2, m_3, m_4$	$p_2 = M(M(1, \bar{B}, \bar{C}), M(0, A, B), 0)$	3	2	2						
Proposed	$p_1 = \sum m_0, m_2, m_4, m_7$	$p_1 = M(M(\underline{B}, 0, \underline{C}), M(A, C', 1), M(A, C, B'))$	4	2	2	1	0	$(7 - 1) = 6$	4	10	2
	$p_2 = \sum m_0, m_2, m_3, m_4$	$p_2 = M(M(\underline{B}, 0, A), M(0, \underline{B}, C), C')$	3	2	2						

NMV Number of MV, N/INV Number of INV gate, CMV Number of common MV, C/INV Number of common INV gate, TMV Total MV, T/INV Total INV gate, TG TMV+TINV

Table 4 Comparison for four (4) input and two (2) output function(s)

Method	Functions	Circuit	NMV	NINV	Levels	CMV	CINV	TMV	TINV	TG	Max level
[34]	$p_1 = \sum m_0, m_2, m_6, m_{12}, m_{13}, m_{14}$ $p_2 = \sum m_1, m_3, m_4, m_5, m_7, m_{12}, m_{13}, m_{15}$	$p_1 = M(M(\bar{D}, M(B, A, 1), M(A, B, C)), M(A, 1, C), M(1, B, A))$ $p_2 = M(M(M(B, A, 1), D), M(A, B, C), M(A, C, 1), B)$	6	5	3	3	3	(11 - 3)=8	(8 - 3)=5	13	3
Proposed	$p_1 = \sum m_0, m_2, m_6, m_{12}, m_{13}, m_{14}$ $p_2 = \sum m_1, m_3, m_4, m_5, m_7, m_{12}, m_{13}, m_{15}$	$p_1 = M(M(M(B, C, 1), C, \bar{D}), M(B, 1, \bar{A}), M(A, 0, M(0, B, \bar{C})))$ $p_2 = M(M(\bar{C}, 0, B), D, M(1, \bar{A}, B))$	6	4	3	2	2	(9 - 2)=7	(6 - 2)=4	11	3
Improvement(%)			3	2	2					15.38	same

NMV Number of MV, N/INV Number of INV gate, CMV Number of common MV, C/INV Number of common INV gate, TMV Toal MV, T/INV Total INV gate, TG TMV+TINV

Table 5 Comparison for three (3) inputs and four (4) output function(s)

Method	Minterms	Circuit	NMV	NINV	Levels	CMV	CINV	TMV	TINV	TG	Max level
[34]	$p_1 = \sum m_1, m_4, m_5, m_7$	$p_1 = M(\bar{B}, A, C)$	1	1	1	3	4	(12 - 3)=9	(10 - 4)=6	15	3
	$p_2 = \sum m_3, m_4, m_6$	$p_2 = M(M(A, \bar{1}, \bar{C}), \bar{A}, M(M(C, B, A), 1, A))$	4	3	3						
	$p_3 = \sum m_0, m_2, m_5, m_6$	$p_3 = M(M(\bar{1}, B, C), M(A, 1, C), M(C, B, A))'$	4	3	2						
	$p_4 = \sum m_4, m_6, m_7$	$p_4 = M(B, M(A, \bar{B}, C), M(A, \bar{1}, \bar{C}))$	3	3	2						
Proposed	$p_1 = \sum m_1, m_4, m_5, m_7$	$p_1 = M(\bar{B}, A, C)$	1	1	1	4	2	(13 - 4)=9	(8 - 2)=6	15	3
	$p_2 = \sum m_3, m_4, m_6$	$p_2 = M(M(A, C, 0), M(0, \bar{C}, A), M(B, C, 0))$	4	2	2						
	$p_3 = \sum m_0, m_2, m_5, m_6$	$p_3 = M(M(M(B, 0, C), A, \bar{C}), B, M(M(C, 0, A), A, B))'$	5	4	3						
Improvement(%)	$p_4 = \sum m_4, m_6, m_7$	$p_4 = M(M(0, \bar{C}, A), A, M(B, C, 0))$	3	1	2					Same	Same

Table 6 Comparison for three (3) input and three (3) output function(s)

Method	Minterms	Circuit	NMV	NINV	Levels	CMV	CINV	TMV	TINV	TG	Max level
[34]	$p_1 = \sum m_2, m_4, m_6$	$p_1 = M(B, M(A, B, 1), M(M(1, C, \bar{B}), 1, A))'$	4	3	3	4	4	(12 - 4)=8	(10 - 4)=6	14	3
	$p_2 = \sum m_0, m_1, m_3, m_6$	$p_2 = M(M(1, A, C), M(\bar{A}, B, \bar{C}), M(A, B, 1))'$	4	3	2						
	$p_3 = \sum m_0, m_3, m_6$	$p_3 = M(M(\bar{A}, B, \bar{C}), \bar{1}, M(M(1, C, \bar{B}), 1, A))$	4	4	3						
Proposed	$p_1 = \sum m_2, m_4, m_6$	$p_1 = M(M(L, B, A), 0, \bar{C})$	2	1	2	3	1	(10 - 3)=7	(6 - 1)=5	12	2
	$p_2 = \sum m_0, m_1, m_3, m_6$	$p_2 = M(M(C, 0, A), M(C, A, B), M(L, B, A))$	4	2	2						
	$p_3 = \sum m_0, m_3, m_6$	$p_3 = M(M(C, B, A), M(1, A, B), M(B, \bar{A}, \bar{C}))$	4	3	2						
Improvement(%)										16.67	33.33

NMV Number of MV, N/INV Number of INV gate, CMV Number of common MV, C/INV Number of common INV gate, TMV Toal MV, T/INV Total INV gate, TG TMV+TINV

Table 7 Comparison for four (4) inputs and four (4) output function(s)

Method	Minterms	Circuit	NMV	NINV	Levels	CMV	CINV	TMV	TINV	TG	Max level
[34]	$P_1 = \sum m_3, m_4, m_7, m_{15}$	$P_1 = M(M(D, M(C, B, A)', B), M(C, \bar{1}, D), \bar{D})$	4	3	3	8	6	(17 - 8)=9	(14 - 6)=8	17	4
	$P_2 = \sum m_1, m_3, m_4, m_9, m_{13}, m_{15}$	$P_2 = M(M(D, M(C, B, A)', B), M(\bar{1}, M(\bar{D}, A, 1), B), \bar{B})$	5	4	3						
	$P_3 = \sum m_3, m_6, m_7, m_{11}, m_{13}, m_{14}, m_{15}$	$P_3 = M(M(\bar{1}, M(\bar{D}, A, 1), B), C, D)$	3	2	3						
	$P_4 = \sum m_2, m_6, m_{10}, m_{11}, m_{14}$	$P_4 = M(\bar{1}, C, M(M(C, \bar{1}, D), M(D, M(C, B, A)', B), \bar{D})')$	5	5	4						
[40]	$P_1 = \sum m_3, m_4, m_7, m_{15}$	$P_1 = M(M(C, 1, D)'), M(\bar{A}, B, D), M(C, 0, D)$	4	2	2	3	2	(15 - 3)=12	(7 - 2)=5	17	3
	$P_2 = \sum m_1, m_3, m_4, m_9, m_{13}, m_{15}$	$P_2 = M(M(D, 0, M(A, \bar{B}, D)), M(\bar{A}, B, D), M(B, C, D)')$	5	3	3						
	$P_3 = \sum m_3, m_6, m_7, m_{11}, m_{13}, m_{14}, m_{15}$	$P_3 = M(M(B, C, D), 0, M(1, A, C))$	3	0	2						
	$P_4 = \sum m_2, m_6, m_{10}, m_{11}, m_{14}$	$P_4 = M(M(C, \bar{D}, 0), C, M(\bar{A}, B, D))$	3	2	2						
Proposed	$P_1 = \sum m_3, m_4, m_7, m_{15}$	$P_1 = M(B, M(C, D, 0), M(A, B, M(B, C, D)))'$	4	1	3	5	1	(15 - 5)=10	(5 - 1)=4	14	3
	$P_2 = \sum m_1, m_3, m_4, m_9, m_{13}, m_{15}$	$P_2 = M(B, M(A, B, M(B, C, D))), M(A, D, 0), \bar{B}, D)$	5	2	3						
	$P_3 = \sum m_3, m_6, m_7, m_{11}, m_{13}, m_{14}, m_{15}$	$P_3 = M(C, M(B, C, D), M(A, D, 0))$	3	0	2						
	$P_4 = \sum m_2, m_6, m_{10}, m_{11}, m_{14}$	$P_4 = M(C, \bar{D}, M(\bar{B}, M(A, D, 0), 0))$	3	2	3						
Improvement(%)			17.64 same								

NMV Number of MV, NINV Number of INV gate, CMV Number of common MV, CINV Number of common INV gate, TMV Total MV, TINV Total INV gate, TG TMV+TINV

method outperforms the other methods considering the functions presented in Tables 4, 6, and 7. Table 4 shows an improvement of 15.38% considering the total number of gates for the 4 input/ 2 output function. The proposed method shows a significant improvement of 16.67% in the total number of gates, and also 33.33% improvement in case of maximum levels for 3 input/ 3 output function (Table 6). 17.64 % improvement is achieved for the 4 input/ 4 output function in terms of the total number of gates (Table 7). The proposed method utilizes a priority-driven optimization methodology where utmost priority is given to level followed by MV and INV. For example, in Table 7, it may record a higher total majority voters (TMV) count in the proposed method than that of [34], the proposed method shows an improvement in terms of the maximum level.

Also, it is evident from the simulation results that the proposed method does not generate redundant gates, while some of the previous circuits contain obvious redundancy. For example, in Table 4, considering [34] the function F_1 produced two redundant INV gates and also the function F_2 produced one redundant INV gate, whereas the proposed method does not contain any redundant gates (MV, INV) (Table 4). The proposed method takes extra care to remove this type of redundancy from the logic level, which can be seen from all the tables (Table 3, 4, 5, 6 and 7).

6 Conclusion

In this paper, an elitism-based genetic algorithm is applied to synthesize optimal QCA circuits of any arbitrary multi-output boolean function, which improves the quality of the solution by preserving the best chromosomes throughout the generations. The proposed methodology uses the relative priority of the objectives where maximum priority has been given to reducing the level, followed by reducing the MV and INV and the proper placement of QCA primitives in each level in the QCA circuit. Moreover, the process of eradication of redundancy is executed to avoid the possibility of any redundant gate in the final outcome. The comparison results demonstrated that the proposed method achieved a 33.33% improvement in terms of maximum levels for 3 input/ 3 output, whereas 17.64% enhancement is attained in terms of the number of gates for the 4 input/ 4 output function. The simulation is performed till function with 4 inputs & 4 outputs without observing any loss in generality. The proposed method can also be utilized for the case of multi-output functions, where input variables can be of an arbitrary number.

Acknowledgements This work is sponsored by the Young Faculty Research Fellowship (YFRF) of Visvesvaraya Ph.D. scheme through the grant number MLA/MUM/GA/ 10(37)B.

Data Availability The authors declare that the data supporting the findings of this study are available within the article.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Abraham A, Jain L (2005) Evolutionary multiobjective optimization. *Evolutionary Multiobjective Optimization* pp 1–6
2. Ahmadpour SS, Mosleh M (2021) Ultra-efficient adders and even parity generators in nano scale. *Comput Electr Eng* 96:107548
3. Ahmadpour SS, Mosleh M, Heikalabad SR (2020) The design and implementation of a robust single-layer qca alu using a novel fault-tolerant three-input majority gate. *J Supercomput* 76:10155–10185
4. Ahmadpour SS, Mosleh M, Asadi MA (2021) The development of an efficient 2-to-4 decoder in quantum-dot cellular automata. *Iran J Sci Technol Trans Electr Eng* 45:391–405
5. Ahmadpour SS, Mosleh M, Heikalabad SR (2022) Efficient designs of quantum-dot cellular automata multiplexer and ram with physical proof along with power analysis. *J Supercomput* 78:1672–1695
6. Ahn CW, Ramakrishna RS (2003) Elitism-based compact genetic algorithms. *IEEE Trans Evolut Comput* 7(4):367–385. <https://doi.org/10.1109/TEVC.2003.814633>
7. Bhanja S, Ottavi M, Lombardi F, Pontarelli S (2006) Novel designs for thermally robust coplanar crossing in qca. In: 2006 Design, Automation and Test in Europe, IEEE Computer Society, vol 1, pp 6
8. Bonyadi MR, Azghadi SMR, Rad NM, Navi K, Afjei E (2007) Logic optimization for majority gate-based nanoelectronic circuits based on genetic algorithm. In: 2007 International Conference on Electrical Engineering, pp 1–5, <https://doi.org/10.1109/ICEE.2007.4287307>
9. Campos CAT, Marciano AL, Neto OPV, Torres FS (2015) Use: a universal, scalable, and efficient clocking scheme for qca. *IEEE Trans Comput-Aided Des Integr Circuits Syst* 35(3):513–517
10. Changdar C, Mahapatra G, Pal RK (2014) An efficient genetic algorithm for multi-objective solid travelling salesman problem under fuzziness. *Swarm Evolut Comput* 15:27–37. <https://doi.org/10.1016/j.swevo.2013.11.001>
11. Chu PC, Beasley JE (1997) A genetic algorithm for the generalised assignment problem. *Comput Operations Res* 24(1):17–23
12. Coello CAC, Lamont GB, Van Veldhuizen DA et al (2007) *Evolutionary algorithms for solving multi-objective problems*, vol 5. Springer, Berlin
13. Deb K (2015) *Multi-Objective Evolutionary Algorithms*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 995–1015. doi: https://doi.org/10.1007/978-3-662-43505-2_49
14. Dysart TJ, Kogge PM (2007) Probabilistic analysis of a molecular quantum-dot cellular automata adder. In: 22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT 2007), IEEE, pp 478–486
15. Goswami M, Mondal A, Mahalat MH, Sen B, Sikdar BK (2019) An efficient clocking scheme for quantum-dot cellular automata. *Int J Electron Lett* pp 1–14
16. Gunter AK (2019) Design and investigation of genetic algorithmic and reinforcement learning approaches to wire crossing reductions for pnml devices
17. Hen CK (2011) Design and development of automated digital circuit structure base on evolutionary algorithm method. *Int J Electron Comput Commun Technol* 2(1):1–8
18. Hennessy K, Lent CS (2001) Clocking of molecular quantum-dot cellular automata. *J Vac Sci Technol B Microelectron Nanometer Struct Process Meas Phenomena* 19(5):1752–1755
19. Houshmand M, Khayat SH, Rezaei R (2009) Genetic algorithm based logic optimization for multi- output majority gate-based nano-electronic circuits. In: 2009 IEEE International Conference on Intelligent Computing and Intelligent Systems, vol 1, pp 584–588, doi: <https://doi.org/10.1109/ICICISYS.2009.5357775>
20. Houshmand M, Saleh RR, Houshmand M (2011) Logic Minimization of QCA Circuits Using Genetic Algorithms, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 393–403. doi: https://doi.org/10.1007/978-3-642-20505-7_35
21. Jong KAD, Spears WM (1989) Using genetic algorithms to solve np-complete problems
22. khosroshahy MB, Daliri MS, Abdoli A, Navi K, Bagherzadeh N (2016) A 3d universal structure based on molecular-qca and cnt technologies. *J Mol Struct* pp 86–95
23. Khosroshahy MB, Moaiyeri MH, Angizi S, Bagherzadeh N, Navi K (2017) Quantum-dot cellular automata circuits with reduced external fixed inputs. *Microprocess Microsyst* 50:154–163

24. Konak A, Coit DW, Smith AE (2006) Multi-objective optimization using genetic algorithms: a tutorial. *Reliab Eng Syst Safe* 91(9):992–1007
25. Kong K, Shang Y, Lu R (2010) An optimized majority logic synthesis methodology for quantum-dot cellular automata. *IEEE Trans Nanotechnol* 9(2):170–183. <https://doi.org/10.1109/TNANO.2009.2028609>
26. Lent CS, Tougaw PD (1997) A device architecture for computing with quantum dots. In: *Proceedings of the IEEE*
27. Lent CS, Tougaw PD, Porod W, Bernstein GH (1993) Quantum cellular automata. *Nanotechnology* 4(1):49
28. Liu P, Ni J, Chu Z (2022) Wire-crossings optimization based on majority-of-five and xor-of-three primitives in qca. *Int J Theor Phys* 61(3):1–22
29. Machado RS, Castellanos J, Lahoz-Beltra R (2016) Evolutionary synthesis of qca circuits: a critique of evolutionary search methods based on the hamming oracle. *Int J Inf Technol Knowl* 10(3):203–215
30. Neto OPV, Pacheco MAC, Barbosa CRH (2007) Neural network simulation and evolutionary synthesis of qca circuits. *IEEE Trans Comput* 56(2):191–201
31. Orlov A, Amlani I, Bernstein G, Lent C, Snider G (1997) Realization of a functional cell for quantum-dot cellular automata. *Science* 277(5328):928–930
32. Pal J, Bhattacharjee S, Saha AK, Dutta P (2019) Study on temperature stability and fault tolerance of adder in quantum-dot cellular automata. In: *2019 5th International Conference on Signal Processing, Computing and Control (ispcc)*, IEEE, pp 69–74
33. Potvin JY (1996) Genetic algorithms for the traveling salesman problem. *Ann Operations Res* 63(3):337–370
34. Rezaee R, Houshmand M, Houshmand M (2013) Multi-objective optimization of qca circuits with multiple outputs using genetic programming. *Genet Program Evol Mach* 14(1):95–118. <https://doi.org/10.1007/s10710-012-9173-6>
35. Roohi A, Menbari B, Shahbazi E, Kamrani M (2013) A genetic algorithm based logic optimization for majority gate-based qca circuits in nanoelectronics. *Quant Matter* 2(3):219–224
36. Roohi A, Thapliyal H, DeMara R (2015) Wire crossing constrained qca circuit design using bilayer logic decomposition. *Electron Lett* 51(21):1677–1679
37. Sarvaghad-Moghaddam M, Orouji AA, Houshmand M (2017) A multi-objective synthesis methodology for majority/minority logic networks. *J Comput Electron* 16(1):162–179. <https://doi.org/10.1007/s10825-016-0938-7>
38. Sastry K, Goldberg DE, Kendall G (2014) *Genetic Algorithms*, Springer US, Boston, MA, pp 93–117. doi: https://doi.org/10.1007/978-1-4614-6940-7_4
39. Sridharan K, Pudi V (2015) *Design of arithmetic circuits in quantum dot cellular automata nanotechnology*, vol 599. Springer, Berlin
40. Tehrani MA, Navi K, Kia-kojooi A (2013) Multi-output majority gate-based design optimization by using evolutionary algorithm. *Swarm Evolut Comput* 10:25–30. <https://doi.org/10.1016/j.swevo.2012.12.002>
41. Tougaw PD, Lent CS (1994) Logical devices implemented using quantum cellular automata. *J Appl Phys* 75(3):1818–1825
42. Vasicek Z, Sekanina L (2016) Evolutionary design of complex approximate combinational circuits. *Genet Program Evol Mach* 17(2):169–192. <https://doi.org/10.1007/s10710-015-9257-1>
43. Vijayakumari C, Mythili P, James RK, Kumar CA (2015) Genetic algorithm based design of combinational logic circuits using universal logic modules. *Procedia Comput Sci* 46:1246–1253
44. Walus K, Schulhof G, Jullien GA, Zhang R, Wang W (2004) Circuit design based on majority gates for applications with quantum-dot cellular automata. In: *Conference Record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers, 2004.*, vol 2, pp 1354–1357 Vol.2, doi: <https://doi.org/10.1109/ACSSC.2004.1399374>
45. Whitley D (1994) A genetic algorithm tutorial. *Stat Comput* 4(2):65–85
46. Zhang R, Walus K, Wang W, Jullien GA (2004) A method of majority logic reduction for quantum cellular automata. *IEEE Trans Nanotechnol* 3(4):443–450

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

Authors and Affiliations

Amit Kumar Pramanik¹ · Mahabub Hasan Mahalat² · Jayanta Pal³ · Seyed-Sajad Ahmadpour⁴  · Bibhash Sen²

Amit Kumar Pramanik
amit.pramanik21@gmail.com

Mahabub Hasan Mahalat
mahabubhasan.mahalat@gmail.com

Jayanta Pal
jayantapal@tripurauniv.ac.in

Bibhash Sen
bibhash.sen@cse.nitdgp.ac.in

¹ Department of CSE, Dumka Engineering College, Dumka, Jharkhand, India

² Department of CSE, NIT Durgapur, Durgapur, West Bengal, India

³ Department of IT, Tripura University, Suryamaninagar, India

⁴ Department of Computer Engineering, Faculty of Engineering and Natural Sciences, Kadir Has University, Istanbul, Turkey