IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 22, 2021, accepted May 6, 2021, date of publication May 11, 2021, date of current version December 30, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3079310

A Heuristic-Driven and Cost Effective Majority/ Minority Logic Synthesis for Post-CMOS Emerging Technology

VIPUL KUMAR MISHRA[1], MAYANK DIXIT[1], TEJALAL CHOUDHARY[1], ANURAG GOSWAMI[1], MANJIT KAUR[1], OMAR CHEIKHROUHOU[2], AND HABIB HAMAM[3], (Senior Member, IEEE)

[1]Department of Computer Science Engineering, School of Engineering and Applied Sciences, Bennett University, Greater Noida 201310, India
[2]College of Computers and Information Technology, Taif University, Taif 21944, Saudi Arabia
[3]Faculty of Engineering, University of Moncton, Moncton, NB E1A 3E9, Canada

Corresponding author: Manjit Kaur (manjitbhinder8@gmail.com)

This work was supported by the Project Taif University Researchers through the Taif University, Taif, Saudi Arabia, under Project TURSP-2020/55.

ABSTRACT Due to the physical restriction of current CMOS technology, emerging technologies that have majority logic gate as a base component are being explored. The process of transforming from boolean network to the majority logic network is called majority logic synthesis (MLS). Hence, the contributions of this work is as follows: (i) a novel heuristic-driven tabular approach for majority logic synthesis has been presented that overcomes the scalability problem of previous synthesis algorithms, (ii) a heuristic named as matching difference (MD) is proposed to guide the synthesis process, (iii) to maintain the trade-off between area and delay during majority logic synthesis a novel criterion ''cost of circuit''(CoC) has been proposed, (iv) For reduction of majority circuit delay during MLS, an extended library based on five-input and three-input majority gates is presented, and (v) a post-synthesis optimization method is proposed based on majority algebra. Based on experiments with MCNC benchmarks, it is verified that the proposed approach accomplishes an average diminution of 24% at the majority level and 31% in the majority gate count. Further, while executing a case study with quantum dot cellular automata(QCA), the proposed methodology is able to achieve an average diminution of 36% in delay and 15% in circuit cost with a penalty of 2% in the area.

INDEX TERMS Majority gate, QCA, NML, majority logic synthesis.

I. INTRODUCTION

The scaling of a transistor has major advantages, such as making transistors cheaper, faster, and energy-efficient, as described by Moore's law [1]. On the other hand, further scaling down unfolds several complexities such as doping fluctuations, higher gate leakage current, capacitive coupling, increased difficulties in lithography, and electro-migration failures due to the underlying physical confinement of current technology i.e. complementary metal oxide semiconductor (CMOS) [1]. Therefore, emerging technologies are often considered a suitable alternate for CMOS to succour Moore's law. Emerging technologies such as quantum dot cellular automata (QCA) [2]–[4], spin wave device (SWD) [5], [6], all spin logic (ASL) [7]–[9], single electron tunnelling (SET) [10], tunnelling phase logic (TPL) [11], nano magnetic logic (NML) [12]–[14],DNA strand displacement [15], [16] are just a few of possible alternative to CMOS technology [17]. There are different ways to represent the binary information in these technologies based on their physical properties. For example, in QCA, a binary information is encoded within the electron pair configuration residing in a quantum cell, whereas in NML, nano-magnets (the magnetization stored in a single domain) represents binary information that can be transmitted by utilizing the magneto dynamic interaction amidst neighbor elements.

The Majority gate acts as an essential building block in these technologies [2]. It is utilized for transforming the boolean network into a majority network along with a majority inverter. This transformation process is called majority/minority logic synthesis(MLS) [18], [19]. Due to the different gate family and structure of emerging technologies from current CMOS technology, Logic Synthesis for emerging technologies promising enough to merit

The associate editor coordinating the review of this manuscript and approving it for publication was Cheng Qian.

further investigation even if extensive research is already available for CMOS technology [20], [21]. The problem of majority logic synthesis was first addressed in the 1960s using reduced unitized tables in [18]. In another research [19], a K-map-based method was presented, while Shannon's decomposition-based principles in majority synthesis was introduced in [22].

These methods suffer from scalability problems and are not appropriate for a large network.The methods presented in [18], [19], [22] become intractable and impractical for a large network. Moreover, in [23]–[25], researchers proposed a three-cube technique created from a geometric interpretation of Boolean function for MLS. These techniques cannot handle more than three variables and were done manually. Furthermore, researchers proposed methods that can handle more than three variables presented in [26]–[29], but these techniques utilized K-map for MLS. Primary efforts for automated MLS were presented in [27], but these methods were not generating superior results in majority levels and majority gate count. In [26] and [28], the authors proposed further improvement on [27]. In another research [30], authors proposed a genetic algorithm based technique for logic optimization.

In [26], a disjoint concept was utilized for further improvement, whereas in [28] author proposed a methodology where each boolean function converts into a majority function twice: first, for original function; second, for its complement with the hope of developing a better majority circuit. But this increased the synthesis time by 2x. Furthermore, in [29], a technique which employed three input majority gate to handle four variable networks was presented. In [29], the authors first developed a majority expression lookup table (MLUT) library then mapped the function from the library. But the library generation is an exhaustive process. The drawback of the approaches [28], [29] mentioned above were exhaustive. Moreover, the methodologies presented in [26]–[29] used an old tool SIS [31] for the preprocessing of a Boolean network when a much better tool for the same operation is available and it can produce better results than the SIS tool. In another research [32], a binary decision diagram (BDD) based MLS was presented. The major drawback of the majority of these methods is that they utilized K-map for MLS. When dealing with higher input majority gates such as 5-input or 7-input during MLS using the K-map method, it becomes very intractable because the K-map method for more than four variables is a complicated task [33]. Because of this, previous approaches cannot utilize higher input majority gate.

Therefore, to utilize higher input majority gates, a novel approach for MLS is required. Moreover, the method able to generate high quality results and should not be exhaustive. Moreover, the utilization of a higher input majority gate (such as 5-input majority gate) during MLS can further enhance the majority circuit in terms of the majority level and number of majority gates, which immediately impacts the delay and area of the majority circuit. In order to conquer the shortcomings presented in the previous approaches, a heuristic-based tabular approach for majority/minority logic synthesis is proposed in this paper. Moreover, we proposed a metric Cost of Circuit (CoC) to determine the cost of the circuit based on the delay and area of the majority network. Furthermore, to decrease the majority level and gate count, the 5-input majority gate utilized in the synthesis process as primitives. The preliminary work has been published in [34]. The considerable contributions are as follows:

1) A novel cost-aware heuristic-based tabular approach for majority/minority logic synthesis is proposed.
2) A heuristic named as matching difference (MD) is presented in this paper to lead the majority synthesis process.
3) An extended library that includes 3-input and 5-input majority gates is proposed in this paper.
4) To maintain the trade-off between area and delay of the majority circuit, we have devised a novel parameter cost of the circuit(CoC).

The remainder of this paper is organized as follows: In section II, we present the background details. Motivation for the need fora novel majority logic synthesis and use of higher input majority gates are explained with an example in section III. The proposed methodology is explained with a demonstration in section IV. Results and analysis followed by the conclusion are given in section V and Section VI, respectively.

II. BACKGROUND

This section details the background which will help the user reading this article is described in this section.

A. QCA CELL

In a QCA cell, four quantum dots in a square block attached by tunnel barriers. Electrons can tunnel among the dots but can't leave the cell. A Coulomb repulsion will force the electrons to dot on opposite corners when two additional electrons are placed in the cell. Thus logic '0' and '1' can be labelled by two energetically equivalent ground state polarizations as shown in Figure 1(a) [35].

B. MAJORITY GATE

Majority gate is a basic logic device in QCA that works on the theory of majority voters. A three-input majority gate [3] is shown in Figure 1(d). It is comprised of three inputs labelled as P, Q, R, and an output. Equation 1 show operation of majority gate [35].

$$M(P, Q, R) = PQ + PS + RS \tag{1}$$

Moreover, a 5-input majority gate is shown in the Figure 1(e) [36], [37]. Equation 2 expressed the working of a 5-input majority gate [36], [37].

$$M(P, Q, R, S, T) = PQR + PQS + PQT + PRS + PRT + PST + QRS + QRT + QST + RST \tag{2}$$

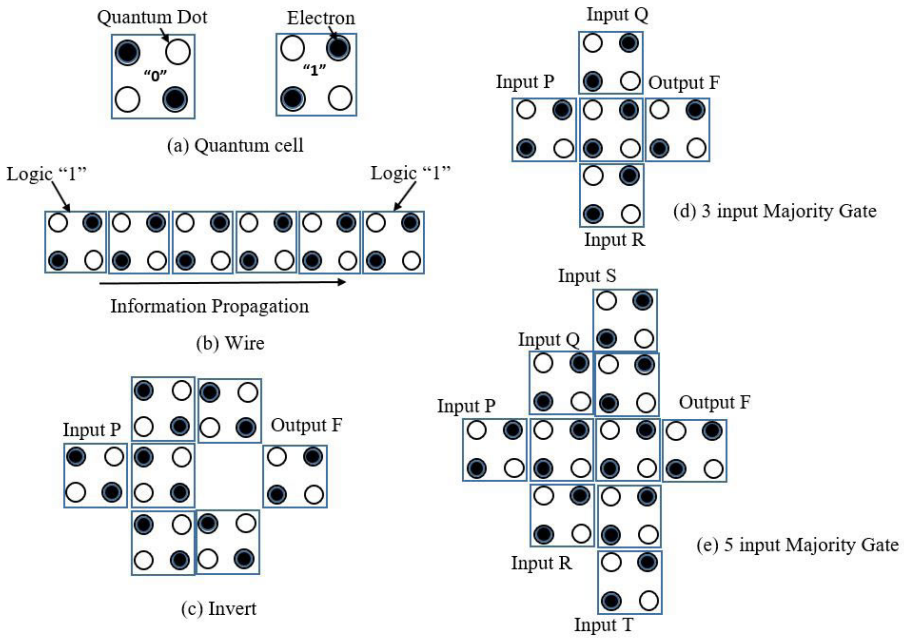


FIGURE 1. Basic QCA [35] (a) basic quantum cell, (b) wire, (c) invert gate, (d) 3 input majority gate, (e) 5 input majority gate.

C. QCA PRIMITIVES

By fixing one of the majority gate input HIGH (1) and LOW (0), the three-input majority gate act as a two-input OR gate and two-input AND, respectively. Equation 3 and 4 show the majority gate as "OR" and "AND" gate. Where, input "Q" (as shown in 1(d)) value set with "1" and "0" to device "OR" and "AND" gate respectively [35].

$$M(P, Q, 1) = P + Q \tag{3}$$

$$M(P, Q, 0) = PQ \tag{4}$$

In addition, a wire and invert implementation in QCA is shown in Figure 1 (*b*) and (*c*) . By placing QCA cells next to each other, they act as a QCA wire. [35].

III. MOTIVATION

This is an example of a boolean function, F = P*Q*R. To transform 'F' to the majority function 'M', two options can be considered. First, utilize only a three-input majority gate, and second, use a five-input majority gate. The implementation of function 'F' is shown in Figure 2(a) and Figure 2(b) respectively.

It is evident from Figure 2(a) and Figure 2(b) that the synthesis of 'F' needs a minimum of two 3-input majority gate with majority level 2 (i.e., delay of a two-clock cycle) and required area of 11 QCA cells, if only 3-input majority gates are used, on the other hand, If a five-input majority gate is taken into account during the synthesis of F, then a single five-input majority gate is enough for synthesizing the function F, this implementation has only one majority level (i.e., delay of a one-clock cycle) without increasing the

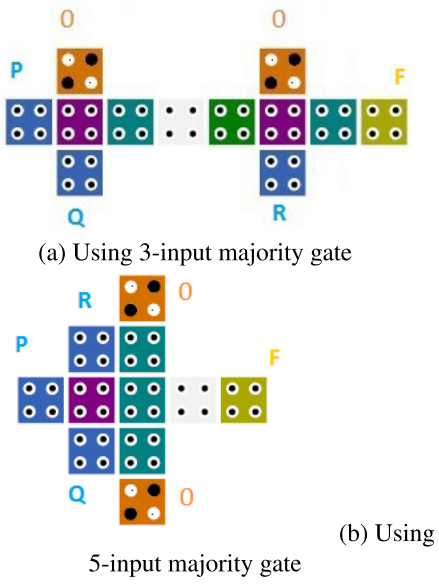


FIGURE 2. Majority synthesis of function F = P * Q * R.

area of the circuit, i.e., 11 QCA cells as an implemented function of F in QCADesigner [38]. Therefore, utilization of a five-input majority gate can reduce majority levels (i.e., directly proportionate to delay of the circuit) without any area increment.

It is evident from the previous paragraph that the utilization of a five-input majority gate or a higher input such as a 5-input majority gate [36], [37] or 7-input majority gate [39] can further optimize the circuit in terms of performance. Due to the lack of a majority logic synthesis approach that

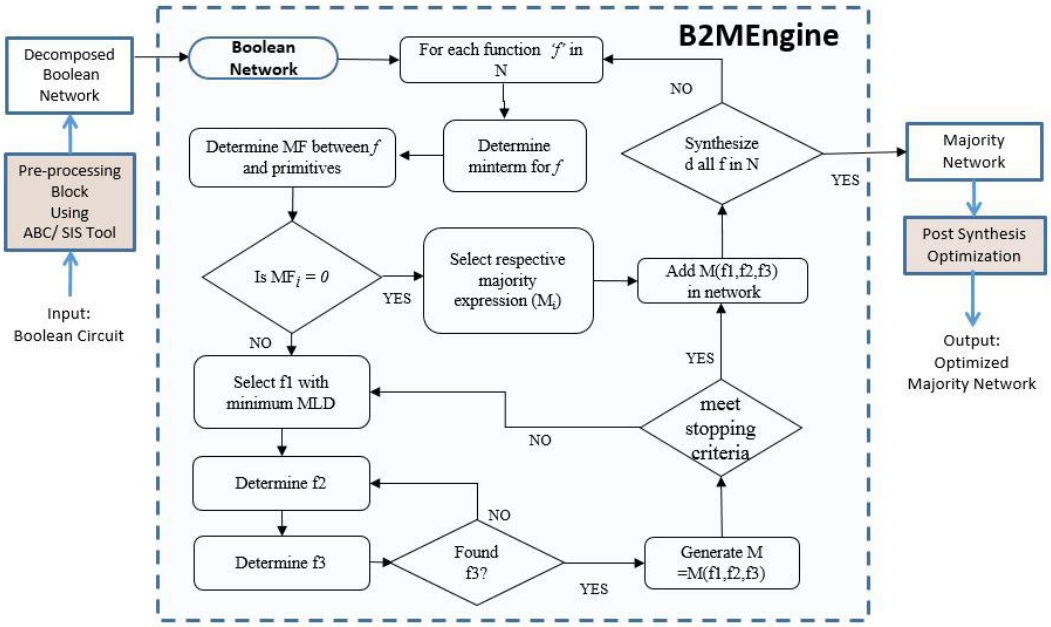


FIGURE 3. Block diagram of proposed majority logic synthesis.

utilizes a higher input majority gate, a gap is created between the available majority gate and the previous majority logic synthesis approaches because the methodologies available in the literature [26]–[29] utilized only three-input majority gate. Therefore, it gives us the motivation to investigate a new methodology that can utilize a higher input majority during MLS to overcome the reduced gap between MLS and the available majority gates.

IV. PROPOSED METHODOLOGY AND IMPLEMENTATION

A detailed description of the methodology is presented in this section.

A. OVERVIEW OF THE PROPOSED APPROACH

The block diagram in figure 3 provides an overview of the proposed MLS technique. The proposed MLS technique begins by taking Boolean network 'N' as an input, which produces an optimized majority network 'Nm' as an output. The next task is the preprocessing and decomposition of the network. Then, the proposed methodology converts the Boolean network to a majority network. After this, redundancy removal and post-synthesis optimization are performed for further improvement of the majority network. A detailed description of the proposed algorithm is presented in subsequent subsections.

B. PREPROCESSING AND DECOMPOSITION

The proposed algorithm starts with inputing the Boolean function from the user. The second is to perform simplification of the given Boolean network. The third step is to perform algebraic factorization, which gives an algebraic factor for Boolean functions have a maximum fan-in three because such a function can convert it into a majority function by using a maximum of four majority gates and two levels. [19], [26], [27]. Therefore, a Boolean network containing X Boolean function is generated that can be transformed into a majority network using a minimum X and a maximum 4*X majority network. Because of this, the preprocessing and decomposition minimize X, which is directly proportional to the number of majority gate and majority level.

The preprocessing and decomposition are performed by SIS [31] and ABC [40], [41]. Both tools are used to gain better results. The script for preprocessing and decomposition is adopted from [34]. This step produces an optimized Boolean network wherein each Boolean function fan-in is a maximum of three.

C. EXTENDED PRIMITIVES

The primitives are the Boolean functions that can be constituted of a single majority gate. The library for the MLS process are these primitives. We included a higher input primitive such as a five-input majority gate along with a three-input majority gate in our approach. The primitive library, which includes a five-Input and a three-input majority gate, is presented in Table 2 and 1 respectively. With these primitives, the proposed algorithm can be transformed by any Boolean function into a majority function.

D. PROPOSED HEURISTIC: MATCHING DIFFERENCE (MD)

A novel heuristic named as matching difference is proposed in this paper. This heuristic helps in the synthesis process in order to find the best matched majority expression for the given Boolean function. If MD = 0, then it indicates that the selected Boolean function is a primitive function. The

TABLE 1. Primitives based on three-input majority gates.

S. no.	Primitive function	Majority expression	S. no.	Primitive function	Majority expression
1	0	0	21	Y'+Z'	M(Y,0,Z)'
2	1	1	22	X'+Y	M(X',1,Y)
3	X	X	23	X'+Z	M(X',1,Z)
4	Y	Y	24	Y'+Z	M(Y',1,Z)
5	Z	Z	25	X'Y'	M(X,1,Y)'
6	X'	X'	26	X'Z'	M(X,1,Z)'
7	Y'	Y'	27	Y'Z'	M(Y,1,Z)'
8	Z'	Z'	28	X'Y	M(X',0,Y)
9	XY	M(X,0,Y)	29	X+Y'	M(X,1,Y')
10	XZ	M(X,0,Z)	30	X'Z	M(X',0,Z)
11	YZ	M(Y,0,Z)	31	X+Z'	M(X,1,Z')
12	X+Y	M(X,1,Y)	32	Y'Z	M(Y',0,Z)
13	X+Z	M(X,1,Z)	33	Y+Z'	M(Y,1,Z')
14	Y+Z	M(Y,1,Z)	34	X'Y'+X'Z'+Y'Z'	M(X,Y,Z)'
15	XY+XZ+YZ	M(X,Y,Z)	35	X'Y+X'Z+YZ	M(X',Y,Z)
16	XY'	M(X,0,Y')	36	XY'+XZ+Y'Z	M(X,Y',Z)
17	XZ'	M(X,0,Z')	37	XY'+XZ'+Y'Z'	M(X,Y',Z')
18	YZ'	M(Y,0,Z')	38	X'Y+X'Z'+YZ'	M(X',Y,Z')
19	X'+Y'	M(X,0,Y)'	39	XY+XZ'+YZ'	M(X,Y,Z')
20	X'+Z'	M(X,0,Z)'	40	X'Y'+X'Z+Y'Z	M(X',Y',Z)

TABLE 2. Primitives based on five-Input majority gates.

S. no.	Primitive function	Majority expression	S. no.	Primitive function	Majority expression
1	X+Y+Z	M(X,Y,Z,1,1)	17	X+(Y*Z)	M(X,X,1,Y,Z)
2	X+Y+Z'	M(X,Y,Z',1,1)	18	X+(Y*Z')	M(X,X,1,Y,Z')
3	X+Y'+Z	M(X,Y',Z,1,1)	19	X+(Y'*Z)	M(X,X,1,Y',Z)
4	X+Y'+Z'	M(X,Y',Z',1,1)	20	X+(Y'*Z')	M(X,X,1,Y',Z')
5	X'+Y+Z	M(X',Y,Z,1,1)	21	X'+(Y*Z)	M(X',X',1,Y,Z)
6	X'+Y+Z'	M(X',Y,Z',1,1)	22	X'+(Y*Z')	M(X,X,0,Y',Z)'
7	X'+Y'+Z	M(X',Y',Z,1,1)	23	X'+(Y'*Z)	M(X,X,0,Y,Z')'
8	X'+Y'+Z'	M(X,Y,Z,0,0)'	24	X'+(Y'*Z')	M(X,X,0,Y,Z)
9	X*Y*Z	M(X,Y,Z,0,0)	25	X*(Y+Z)	M(X,X,0,Y,Z)
10	X*Y*Z'	M(X,Y,Z',0,0)	26	X*(Y+Z')	M(X,X,0,Y,Z')
11	X*Y'*Z	M(X,Y',Z,0,0)	27	X*(Y'+Z)	M(X,X,0,Y',Z)
12	X*Y'*Z'	M(X,Y',Z',0,0)	28	X*(Y'+Z')	M(X,X,0,Y',Z')
13	X'*Y*Z	M(X',Y,Z,0,0)	29	X'*(Y+Z)	M(X',X',0,Y,Z)
14	X'*Y*Z'	M(X',Y,Z',0,0)	20	X'*(Y+Z')	M(X,X,1,Y',Z)'
15	X'*Y'*Z	M(X',Y',Z,0,0)	21	X'*(Y'+Z)	M(X,X,1,Y,Z')'
16	X'*Y'*Z'	M(X,Y,Z,1,1)'	22	X'*(Y'+Z')	M(X,X,1,Y,Z)'

expression of MD is given in equation 5.

$$MD\,(P, Q) = max \begin{cases} |P| - |P \cap Q| \\ |Q| - |P \cap Q| \end{cases} \tag{5}$$

where P and Q are the minterms set, MD(P,Q) is the MD between P and Q, $|P|$ length of P. During the synthesis process, MD gives suggestions of the most suitable majority function. The lower value of MD indicates the most suitable, and the higher value indicates the least suitable majority function.

E. PROPOSED METHODOLOGY FOR BOOLEAN NETWORK TO MAJORITY NETWORK TRANSFORMATION

After pre-processing and decomposition, a Boolean network that includes *'N'* functions has maximum fan-in three. The next step of the proposed methodology is to synthesize a Boolean network to a majority network by transforming Boolean functions to majority functions respectively. The proposed methodology for transforming the majority function from the Boolean function is shown in Figure 1, and demonstration of the proposed methodology with an example is given in Figure 4.

The proposed algorithm first chooses a Boolean function *f* from the Boolean network 'N' then finds the minterms of the selected function *f*. After this, algorithm calculates matching difference (MD) between the primitives p_i(given in Table 1 and Table 2) and *f* using equation 5. If *f* has MD = 0, then respective majority function picked from Table 1 or Table 2 and append it into the majority network 'Nm'. If MD = 0 with respect to all primitive function and Once algorithm found that the selected function is not a primitive function, then an array 'Indicator' size of 2^n is created. with zero initial values (i.e. step 9 in algorithm 1). After the algorithm identifies f_1, f_2, f_3 as described in subsequent sections.

1) IDENTIFY f_1

To find f1, the algorithm first identifies a function f_x with minimum MD from the given primitives as shown in Table 1. In case of tie in MD among primitive function, based on the step 10 of the proposed algorithm, choose the function f_x, with minimum cost (The cost of a function can be calculated using algorithm 2). using algorithm 3 update the Indicator array after selecting f1.

2) IDENTIFY f_2

After updating the Indicator array as step 11 in the proposed algorithm, determine f_2. To accomplish this, the proposed algorithm determines a set of primitive which minterms must cover all indexes with the value of 0 and must not contain the minterms equal to the indexes with a value of -1 as shown in step 12 to 15 of Algorithm 1. Then choose the functions with minimum MD and lowest cost from set 'G' as f_2, as described in step 16. Then, update the Indicator array by the UpdateIndicator algorithm. The example is given in Figure 4.

3) IDENTIFY f_3

After updating the Indicator as shown in step 17 of the proposed algorithm, the next is to find f_3. The proposed algorithm determines a set of primitive which minterms must cover all indexes with indicator value one and must not cover the minterms equal to the indexes with indicator value -1, which is represented In Algorithm 1 from step 18 to 20. In case of empty set H restart the process from step 16 to 21 with another p_i from set G. Then selct one of the functions from set 'H' as f_3 with the lowest cost, as depicted from step 23. The demonstration example also shown in Figure 4. After determining f_1, f_2, f_3, create a majority function as step 24. Then repeat step 9 to step 24 until the stopping criteria is satisfied (explained in section 4.4.4). Finally append majority gate $M\,(f_1, f_2, f_3)$ in the majority network.

4) STOPPING CRITERIA

The stopping criteria of the proposed algorithm are as follows:

1) *if* $(CoC_n \geq CoC_{n-1})$
2) *if* $(n \geq T)$

where n is current iteration count and T is maximum iteration count for a function. The stopping criteria ensure that a better solution should not be missed by algorithm. In addition, this process should also not run for a longer time like exhaustive search.

This should be repeated on the whole Boolean network that is not converted into the majority network.

Algorithm 1 Proposed Algorithm for Boolean to Majority Transformation

$N \leftarrow$ Perform preprocessing and decomposition on given Boolean*Network*
for k = 1 to B **do** { where B = number of Boolean function in Boolean network N}
  Find minterm for f_k: $f^m = minterm(f_k)$
  for i = 1 to J **do** { where J = number of primitives}
    $d_i = MD\,(f^m, p_i)$
  end for
  if $MF_i = 0$ **then**
    Append $(MajorityFunction_i)$ to Majority Network
  else
    while stopping criterion met **do**
      Create an array *Indicator* 'I' size of 2^n, where n is 3
      $f_1 = min\,(cost\,(min\,(MD\,(p_i))))$ WHERE $p_i \in P$
      UPDATECOUNTER $\left(f^m, f_1^m\right)$
      $X_{-1} = \{\, i \;\; where\; C\,[i] = -1\}$
      $X_0 = \{\, i \;\; where\; C\,[i] = 0\}$
      $Y = \{X_0 \cap f^m\}$
      $G = \left\{p_i \mid (p_i \neq f_1) \wedge \left(p_i^m \cap X_{-1} = \phi\right) \wedge \left(Y - p_i^m = \phi\right)\right\}$
      $f_2 = min\,(cost\,(min\,(MD\,(p_i))))$ WHERE $p_i \in G$
      UPDATECOUNTER $\left(f^m, f_2^m\right)$
      $X_{-1} = \{\, i \;\; where\; I\,[i] = -1\}$
      $X_1 = \{\, i \;\; where\; I\,[i] = 1\}$
      $Z = \{X_1 \cap f^m\}$
      $H = \left\{p_i \mid \left(\left(p_i^m \cap X_{-1} = \phi\right) \wedge p_i \neq f_1 \neq f_2\right) \wedge (Z - p_i = \phi)\right\}$
      if $H = \phi$ then repeat step from 16 to 21 with next p_i
      $f_3 = min\,(cost\,(p_i))$ where $p_i \in H$
      GENERATE $M\,(f_1, f_2, f_3)$
    end while
    Append $M\,(f_1, f_2, f_3)$ *toMajorityNetwork*
  end if
end for

Algorithm 2 Majority Function Cost

procedure COST(*Mexp*)
  $M \leftarrow$ NUMBER OF MAJORITY GATES IN(*Mexp*)
  $I \leftarrow$ NUMBER OF INVERTERS IN(*Mexp*)
  $Area \leftarrow M * 10 + C_I$
  if $M > 1$ **then**
    $D_M = 2$
  else
    $D_M = 1$
  end if
  if $I > 0$ **then**
    $D_I = 1$
  else
    $D_I = 0$
  end if
  $Delay \leftarrow D_M + D_I$
  $Cost = w_1 \left(\frac{Delay}{maxDelay}\right) + w_1 \left(\frac{Area}{maxArea}\right)$
  return *Cost*
end procedure=0

F. POST-SYNTHESIS OPTIMIZATION

After analyzing the majority network generated by the MLS algorithm, it was found that further optimization of the generated majority network is possible. To achieve this, the proposed algorithm performed post-synthesis optimization utilizing majority algebra rules published in [42]. The rules used for this is given in equation 6.

$$\Omega = \begin{cases} Commutativity: \\ M_3\,(P, Q, R) = M_3\,(Q, P, R) = M_3\,(R, Q, P) \\ Majority: \\ \begin{cases} if\;(P = Q) : M_3\,(P, Q, R) = P = Q \\ if\;\left(P = Q'\right) : M_3\,(P, Q, R) = R \end{cases} \\ M_3\,(P, P, 1) = P \\ Assocativity: \\ M_3\,(P, S, M_3\,(Q, S, R)) = \\ M_3\,(R, S, M_3\,(Q, S, P)) \\ Distributivity: \\ M_3\,(P, Q, M_3\,(S, T, R)) = \\ M_3\,(M_3\,(P, Q, S)\,, M_3\,(P, Q, T)\,, R) \\ InverterPropogation: \\ M_3\,(P, Q, R)' = M_3\left(P', Q', R'\right) \end{cases} \tag{6}$$

An example of post-synthesis optimization is shown in Figure 5. As evident from Figure 5, post-synthesis

Demonstration of proposed approach:

Consider function f= a'b'c' + abc' + ab'c +a'bc

Step 1: Find minterms for function f

Minterms of f = {0, 3, 5, 6}

Step 2- Find MD between primitives and function

Majority function	Minterms	MD
M(a, b, c)	{3, 5, 6, 7}	1
M(a, b', c')	{0, 4, 5, 6}	1
M(a', b, c')	{0, 2, 3, 6}	1
M(a', b', c)	{0, 1, 3, 5}	1
a	{4, 5, 6, 7}	2
b	{2, 3, 6, 7}	2
.....		

Note- Due to lack of space we show only few primitives and their corresponding MD values for better understanding.

Step 3- Generate counter table

Index	0	1	2	3	4	5	6	7
count	0	0	0	0	0	0	0	0

Step 4- Select f1

We have selected M(a, b, c) as f1. In order to selection of f1, the first condition is minimum MLD and second condition is minimum cost. As shown in step 2, we have four choices for f1 based on MLD but cost of M(a, b, c) is minimum i.e. 10 whereas, the cost of M(a, b', c'), M(a', b, c'), M(a', b', c) are 12,12,12, respectively.

Cost = 10* #majority gate + inverter

Cost (M(a, b, c))= 10*1 +0 = 10

Similarly, cost of other majority function can be calculated.

Update counter table

Index	0	1	2	3	4	5	6	7
count	0	0	0	1	0	1	1	-1

Step 5- Select f2

α = = {7}, β = = {0, 1, 2, 4}

$\lambda = \beta \cap f$ = {0}

The functions which satisfy conditions $f2 \cap \alpha = \phi and \lambda - f2 = \phi$ are M(a, b', c'), M(a', b, c'), M(a', b', c). All three options have same MLD and same cost; thus, we select M(a, b', c') randomly.

Update counter table

Index	0	1	2	3	4	5	6	7
count	1	0	0	1	-1	2	2	-1

Step 6- Select f3

α = = {4, 7}, γ = ={0, 3}

The functions which satisfy conditions $f3 \cap \alpha = \phi and \gamma - f3 = \phi$ are M(a', b, c'), M(a', b', c), a'. the cost of function are 12, 12, 1, respectively. Thus, we select a' as f3.

Finally, the majority expression for f is M= M(f1, f2, f3)

FIGURE 4. Demonstration of proposed methodology.

optimization can further optimize the majority circuit. The methodology is able to reduce one majority level, one inverter, and one majority gate using majority algebra. We have chosen a simple example; but we can achieve more reduction in the number of majority gates and majority levels for a complex circuit

TABLE 3. Area and delay [44], [45].

	Area (μm^2)	Delay (ns)
Majority3	$4.0 * 10^{-3}$	$1.4 * 10^{-2}$
Majority5	$7.64 * 10^{-3}$	$1.4 * 10^{-2}$
Inverter	$1.2 * 10^{-3}$	$4 * 10^{-3}$

G. REDUNDANCY ELIMINATION

It is observed that there are multiple redundant nodes present in the present majority network, either in the form of similar majority gates or complement of a majority gate. A unidirectional graph with respect to the majority network is created to eliminate such redundancy. Next, a redundant node or

TABLE 4. Comparative results between the proposed methodology and previous works in terms of the majority count and the majority level when the majority level has a higher priority than the majority gate count.

	Proposed approach				(23)(2007)		(24)[(2010)		(25)(2015)	
Benchmark	Level	#Maj5	#Maj3	Total	Level	#Maj	Level	#Maj	Level	#Maj
Alu2	10	153	55	208	18	356	18	340	16	347
Apex6	6	250	177	427	17	701	17	662	17	662
B1	2	0	7	7	3	9	2	7	2	6
C8	6	22	69	91	7	115	7	112	7	112
cc	4	18	11	29	5	44	5	43	5	43
cht	3	33	84	117	4	120	4	120	4	120
Cm42a	2	8	5	13	2	21	2	18	2	18
Cm85a	5	3	20	23	7	34	6	26	6	19
Cm150a	9	0	46	46	9	46	9	46	6	37
Cm151a	4	11	4	15	7	42	7	23	7	20
Cm152a	6	0	21	21	6	21	6	21	6	17
Cm162a	5	15	16	31	7	46	7	41	7	41
Cm163a	4	13	13	26	7	42	7	38	7	32
cmb	3	15	4	19	4	44	4	28	4	26
Cu	3	12	17	29	7	46	7	40	7	39
decod	2	4	20	24	3	28	3	28	3	28
Example2	7	58	143	201	10	259	10	247	9	241
Frg1	7	71	30	101	18	111	18	105	17	102
Frg2	10	259	192	451	15	672	14	582	13	600
I1	2	12	10	22	6	41	6	36	6	35
lal	5	15	42	57	8	95	8	82	8	82
Majority	3	2	2	4	4	6	4	6	4	5
mux	5	25	3	28	9	46	9	46	6	37
Pcle	5	23	18	41	8	67	8	62	8	62
Pcler8	3	28	24	52	10	90	9	80	8	90
Pm1	3	14	7	21	6	45	6	35	6	32
sct	4	11	26	37	6	72	6	65	6	65
Term1	7	51	67	118	13	174	11	106	10	89
Ttt2	6	54	29	83	11	154	11	145	10	144
X1	6	127	54	181	11	320	11	264	22	253
X2	4	16	10	26	7	42	7	37	6	36
9symml	8	68	82	150	12	216	10	47	12	45
Average	**4.96**	**43.46**	**40.87**	**84.34**	**8.34**	**128.90**	**8.09**	**110.56**	**8.03**	**108.90**
	Average Improvement%				**40.44**	**34.56**	**38.61**	**23.71**	**38.13**	**22.55**

Demonstration of post-synthesis optimization
Consider the majority function
$M\left(M\left(x,y,z\right)',M\left(x,y,1\right)',z'\right)$

$M\left(M\left(x,y,z\right)',M\left(x,y,1\right)',z'\right)$ applying inverter prorogation
$\Rightarrow M\left(M\left(x,y,z\right),M\left(x,y,1\right),z\right)'$ applying Distributivity
$\Rightarrow M\left(x,y,M\left(z,1,z\right)\right)'$ applying Commutativity and Majority
$\Rightarrow M\left(x,y,z\right)'$ Final optimized majority function

FIGURE 5. Demonstration of post-synthesis optimization.

subgraph is determined, and finally, the redundant node for sub-graph with lower levels sub-graph or node are eliminated as shown in Figure 6.

V. RESULT AND ANALYSIS

A comparative analysis between previous methods [27]–[29] and the proposed approach is presented in this section. For comparative analysis, 38 benchmarks circuit from MCNC benchmark [43] are tested. These results are shown in two subsections. The first subsection presents comparative results for number of majority gates and majority levels. The second subsection presents a test case for QCA and comparative results presented for area, delay, and cost of circuit. The delay and area value of the majority gate in QCA technology is given in Table 3.

Algorithm 3 Update Counter

```
procedure UPDATE COUNTER(f_i^m, f^m)
    for ALL x ∈ f_i^m do
        if x ∈ f^m then
            C[x]++
        else
            C[x]−−
        end if
    end for
end procedure=0
```

where f^m and f_i^m are minterms of f and f_i respectively.

A. COMPARATIVE ANALYSIS OF THE NUMBER OF MAJORITY GATE AND MAJORITY LEVEL

The experiment was conducted to analyze the proposed algorithm twofold: First, when the majority level has a higher priority than a majority gate count, it implies that a circuit delay has a higher priority than area. Second, when the majority gate count has a higher priority than the majority level denotes that the circuit area has a higher priority

TABLE 5. Comparative results between the proposed methodology and previous algorithm in terms of the majority count and majority level the when the majority count has a higher priority than majority-level optimization.

	Proposed approach				(23)(2007)		(24)[(2010)		(25)(2015)	
Benchmark	Level	#Maj5	#Maj3	Total	Level	#Maj	Level	#Maj	Level	#Maj
Alu2	17	123	79	202	18	356	18	340	16	347
Apex6	6	250	177	427	17	701	17	662	17	662
B1	2	0	7	7	3	9	2	7	2	6
C8	6	22	69	91	7	115	7	112	7	112
cc	4	18	11	29	5	44	5	43	5	43
cht	4	39	42	81	4	120	4	120	4	120
Cm42a	2	8	5	13	2	21	2	18	2	18
Cm85a	5	3	20	23	7	34	6	26	6	19
Cm150a	9	0	46	46	9	46	9	46	6	37
Cm151a	7	0	22	22	7	42	7	23	7	20
Cm152a	6	0	21	21	6	21	6	21	6	17
Cm162a	6	8	24	32	7	46	7	41	7	41
Cm163a	6	14	10	24	7	42	7	38	7	32
cmb	3	15	4	19	4	44	4	28	4	26
Cu	3	12	17	29	7	46	7	40	7	39
decod	2	4	20	24	3	28	3	28	3	28
Example2	9	62	128	190	10	259	10	247	9	241
Frg1	11	49	12	61	18	111	18	105	17	102
Frg2	11	207	166	373	15	672	14	582	13	600
I1	2	12	10	22	6	41	6	36	6	35
lal	5	15	42	57	8	95	8	82	8	82
Majority	3	2	2	4	4	6	4	6	4	5
mux	9	0	46	46	9	46	9	46	6	37
Pcle	7	15	28	43	8	67	8	62	8	62
Pcler8	3	28	24	52	10	90	9	80	8	90
Pm1	3	14	7	21	6	45	6	35	6	32
sct	4	11	26	37	6	72	6	65	6	65
Term1	10	26	58	84	13	174	11	106	10	89
Ttt2	6	54	29	83	11	154	11	145	10	144
X1	8	125	31	156	11	320	11	264	22	253
X2	5	13	9	22	7	42	7	37	6	36
9symml	9	9	30	39	12	216	10	47	12	45
Average	**6.03**	**36.18**	**38.18**	**74.37**	**8.34**	**128.90**	**8.09**	**110.56**	**8.03**	**108.90**
	Average Improvement%				**27.71**	**42.30**	**25.48**	**32.73**	**24.90**	**31.70**

TABLE 6. Comparative results between proposed methodology and previous algorithm in terms of area, delay, and CoC.

	Delay (us)				Area (um2)				CoC			
	Prop	**(23)**	**(24)**	**(25)**	**Prop**	**(23)**	**(24)**	**(25)**	**Prop**	**(23)**	**(24)**	**(25)**
	Delay has highest Priority											
Avg	0.067	0.116	0.113	0.11	0.493	0.515	0.442	0.435	0.685	0.899	0.820	0.793
% imp		41.95	40.15	40.38		4.2	-11.72	-13.39		23.76	16.45	13.53
	Area has highest Priority											
Avg	0.082	0.116	0.113	0.11	0.427	0.515	0.442	0.435	0.692	0.899	0.820	0.793
% imp		29.21	27.03	27.31		17.02	3.23%	1.78		23.06	15.68	12.73
	CoC has Highest Priority											
Avg	0.072	0.116	0.113	0.113	0.442	0.515	0.442	0.435	0.668	0.899	0.820	0.793
% imp		38.2	36.29	36.54		14.19	-0.08	-1.57		25.65	18.52	15.67

than circuit delay. Table 4 presents comparative results with [27]–[29] and the proposed algorithm when the majority level has higher priority than the majority gate count. It is observable from Table 4 that the proposed algorithm was able to reduce on average 40%, 38%, and 38% in respect of the majority level and average 34%, 23%, 22% in respect of the total number majority gates counts as contrast to [27]–[29] respectively. Moreover, Table 5 presents comparative results with [27]–[29] and the proposed algorithm when majority gate count has a higher priority than majority level. It is clearly shown in Table 5 that the proposed algorithm is able to achieve an average reduction of 42%, 32%, 31% in the majority gate counts and 27%, 25% and 24% reduction with respect to the majority level as contrast to [27]–[29] respectively.

B. COMPARATIVE ANALYSIS OF DELAY, AREA, AND CoC

This section presents the result of a test case where we analyze the performance of the proposed algorithm in consideration of QCA as a base technology. A comparative analysis of delay, area, and CoC is presented in this subsection. The estimation of delay, area, and CoC is performed using equation 7, 8, and 9 respectively. *Note- to simplify, we did not*

TABLE 7. Comparative results between the proposed methodology and previous works in terms of delay, area, and CoC when delay has the highest priority.

	Delay (us)				Area (um2)				CoC			
	Prop	**(23)**	**(24)**	**(25)**	**Prop**	**(23)**	**(24)**	**(25)**	**Prop**	**(23)**	**(24)**	**(25)**
Alu2	0.14	0.252	0.252	0.224	1.3828	1.424	1.36	1.388	0.622788	0.855289	0.839321	0.790752
Apex6	0.084	0.238	0.238	0.238	2.608	2.804	2.648	2.648	0.641521	1	0.972183	0.972183
B1	0.028	0.042	0.028	0.028	0.024	0.036	0.028	0.024	0.666667	1	0.722222	0.666667
C8	0.084	0.098	0.098	0.098	0.4432	0.46	0.448	0.448	0.607969	0.6793	0.672992	0.672992
cc	0.056	0.07	0.07	0.07	0.1808	0.176	0.172	0.172	0.833333	0.903392	0.89233	0.89233
cht	0.042	0.056	0.056	0.056	0.5868	0.48	0.48	0.48	0.607676	0.651678	0.651678	0.651678
CM42a	0.028	0.028	0.028	0.028	0.0808	0.084	0.072	0.072	0.959091	0.977273	0.909091	0.909091
cm85a	0.07	0.098	0.084	0.084	0.1028	0.136	0.104	0.076	0.626681	0.853148	0.692848	0.607274
Cm150a	0.07	0.126	0.126	0.084	0.1908	0.184	0.184	0.148	0.777778	0.98218	0.98218	0.721174
Cm151a	0.056	0.098	0.098	0.098	0.0996	0.168	0.092	0.08	0.464131	0.794372	0.578953	0.544939
Cm152a	0.084	0.084	0.084	0.084	0.084	0.084	0.084	0.068	0.903846	0.903846	0.903846	0.826923
Cm162a	0.07	0.098	0.098	0.098	0.178	0.184	0.164	0.164	0.838745	0.997835	0.943723	0.943723
Cm163a	0.056	0.098	0.098	0.098	0.1508	0.168	0.152	0.128	0.731341	0.996454	0.949173	0.878251
cmb	0.042	0.056	0.056	0.056	0.13	0.176	0.112	0.104	0.621782	0.835644	0.677228	0.657426
Cu	0.042	0.098	0.098	0.098	0.1592	0.184	0.16	0.156	0.554457	0.893162	0.84188	0.833333
decode	0.028	0.042	0.042	0.042	0.1104	0.112	0.112	0.112	0.69459	0.866492	0.866492	0.866492
Example2	0.098	0.14	0.14	0.126	1.0128	1.036	0.988	0.964	0.827196	0.988127	0.965511	0.904203
Frg1	0.098	0.252	0.252	0.238	0.6596	0.444	0.42	0.408	0.694444	0.836568	0.818375	0.781501
Frg2	0.14	0.21	0.196	0.182	2.7364	2.688	2.328	2.4	0.833333	0.991156	0.892043	0.871866
I1	0.028	0.084	0.084	0.084	0.1312	0.164	0.144	0.14	0.566667	1	0.939024	0.926829
lal	0.07	0.112	0.112	0.112	0.282	0.38	0.328	0.328	0.60002	0.887439	0.834421	0.834421
Majority	0.042	0.056	0.056	0.056	0.0232	0.024	0.024	0.02	0.677083	0.8125	0.8125	0.760417
mux	0.07	0.126	0.126	0.126	0.202	0.184	0.184	0.148	0.708333	0.830446	0.830446	0.741337
Pcle	0.07	0.112	0.112	0.112	0.2468	0.268	0.248	0.248	0.751334	0.976529	0.940967	0.940967
Pcler8	0.042	0.14	0.126	0.112	0.3088	0.36	0.32	0.36	0.578889	1	0.894444	0.9
Pm1	0.042	0.084	0.084	0.084	0.1344	0.18	0.14	0.128	0.623333	1	0.888889	0.855556
sct	0.056	0.084	0.084	0.084	0.1876	0.288	0.26	0.26	0.659028	1	0.951389	0.951389
Term1	0.098	0.182	0.154	0.14	0.6556	0.696	0.424	0.356	0.70431	0.933333	0.671264	0.58908
Ttt2	0.084	0.154	0.154	0.14	0.5264	0.616	0.58	0.576	0.612077	0.897112	0.873904	0.825871
X1	0.084	0.154	0.154	0.308	1.1812	1.28	1.056	1.012	0.59777	0.75	0.6625	0.895313
X2	0.056	0.098	0.098	0.084	0.1616	0.168	0.144	0.144	0.690524	0.920842	0.860721	0.789293
9symml	0.112	0.168	0.14	0.168	0.8448	0.864	0.188	0.18	0.670707	0.772727	0.336069	0.376894
AVERAGE	**0.06783**	**0.1168**	**0.1133**	**0.11375**	**0.49395**	**0.5156**	**0.4421**	**0.4356**	**0.68588**	**0.8995**	**0.8208**	**0.7931**

consider interconnect area and delay during the delay and area calculation The delay and area value of majority gates (3-input and 5-input) are shown in table 3.

$$Area = (N_{M5} * A_{M5}) + (N_{M3} * A_{M3}) \tag{7}$$

$$Delay = Level * D_M \tag{8}$$

$$CoC = w_1 \left(\frac{Delay}{maxDelay} \right) + w_1 \left(\frac{Area}{maxArea} \right) \tag{9}$$

where N_{M3} is the total 3-input majority gates, A_{M3} is the area of a 3-input majority gate, N_{M5} is the total 5-input majority gates, and A_{M3} is the area of a 5-input majority gate, where D_M is the delay of a majority gate.

The comparative study was performed in three cases. In the first case, when the delay optimization has the highest priority. The second case when area optimization has the highest priority, and the third case when CoC has the highest priority, in which we maintained a trade-off between are and delay by

TABLE 8. Comparative results between the proposed methodology and previous works in terms of delay, area, and CoC when area has the highest priority.

	Delay (us)				Area (um2)				CoC			
	Prop	**(23)**	**(24)**	**(25)**	**Prop**	**(23)**	**(24)**	**(25)**	**Prop**	**(23)**	**(24)**	**(25)**
Alu2	0.238	0.252	0.252	0.224	1.2508	1.424	1.36	1.388	0.784298	0.855289	0.839321	0.790752
Apex6	0.084	0.238	0.238	0.238	2.608	2.804	2.648	2.648	0.641521	1	0.972183	0.972183
B1	0.028	0.042	0.028	0.028	0.024	0.036	0.028	0.024	0.666667	1	0.722222	0.666667
C8	0.084	0.098	0.098	0.098	0.4432	0.46	0.448	0.448	0.607969	0.6793	0.672992	0.672992
cc	0.056	0.07	0.07	0.07	0.1808	0.176	0.172	0.172	0.833333	0.903392	0.89233	0.89233
cht	0.056	0.056	0.056	0.056	0.4644	0.48	0.48	0.48	0.643498	0.651678	0.651678	0.651678
CM42a	0.028	0.028	0.028	0.028	0.0808	0.084	0.072	0.072	0.959091	0.977273	0.909091	0.909091
cm85a	0.07	0.098	0.084	0.084	0.1028	0.136	0.104	0.076	0.626681	0.853148	0.692848	0.607274
Cm150a	0.07	0.126	0.126	0.084	0.1908	0.184	0.184	0.148	0.777778	0.98218	0.98218	0.721174
Cm151a	0.098	0.098	0.098	0.098	0.088	0.168	0.092	0.08	0.567615	0.794372	0.578953	0.544939
Cm152a	0.084	0.084	0.084	0.084	0.084	0.084	0.084	0.068	0.903846	0.903846	0.903846	0.826923
Cm162a	0.084	0.098	0.098	0.098	0.1568	0.184	0.164	0.164	0.852814	0.997835	0.943723	0.943723
Cm163a	0.084	0.098	0.098	0.098	0.1464	0.168	0.152	0.128	0.861196	0.996454	0.949173	0.878251
cmb	0.042	0.056	0.056	0.056	0.13	0.176	0.112	0.104	0.621782	0.835644	0.677228	0.657426
Cu	0.042	0.098	0.098	0.098	0.1592	0.184	0.16	0.156	0.554457	0.893162	0.84188	0.833333
decode	0.028	0.042	0.042	0.042	0.1104	0.112	0.112	0.112	0.69459	0.866492	0.866492	0.866492
Example2	0.126	0.14	0.14	0.126	0.9832	1.036	0.988	0.964	0.913249	0.988127	0.965511	0.904203
Frg1	0.154	0.252	0.252	0.238	0.4204	0.444	0.42	0.408	0.624234	0.836568	0.818375	0.781501
Frg2	0.154	0.21	0.196	0.182	2.2372	2.688	2.328	2.4	0.775452	0.991156	0.892043	0.871866
I1	0.028	0.084	0.084	0.084	0.1312	0.164	0.144	0.14	0.566667	1	0.939024	0.926829
lal	0.07	0.112	0.112	0.112	0.282	0.38	0.328	0.328	0.60002	0.887439	0.834421	0.834421
Majority	0.042	0.056	0.056	0.056	0.0232	0.024	0.024	0.02	0.677083	0.8125	0.8125	0.760417
mux	0.126	0.126	0.126	0.126	0.184	0.184	0.184	0.148	0.830446	0.830446	0.830446	0.741337
Pcle	0.098	0.112	0.112	0.112	0.226	0.268	0.248	0.248	0.839349	0.976529	0.940967	0.940967
Pcler8	0.042	0.14	0.126	0.112	0.3088	0.36	0.32	0.36	0.578889	1	0.894444	0.9
Pm1	0.042	0.084	0.084	0.084	0.1344	0.18	0.14	0.128	0.623333	1	0.888889	0.855556
sct	0.056	0.084	0.084	0.084	0.1876	0.288	0.26	0.26	0.659028	1	0.951389	0.951389
Term1	0.14	0.182	0.154	0.14	0.4296	0.696	0.424	0.356	0.641954	0.933333	0.671264	0.58908
Ttt2	0.084	0.154	0.154	0.14	0.5264	0.616	0.58	0.576	0.612077	0.897112	0.873904	0.825871
X1	0.112	0.154	0.154	0.308	1.074	1.28	1.056	1.012	0.601349	0.75	0.6625	0.895313
X2	0.07	0.098	0.098	0.084	0.1348	0.168	0.144	0.144	0.694818	0.920842	0.860721	0.789293
9symml	0.126	0.168	0.14	0.168	0.1884	0.864	0.188	0.18	0.313573	0.772727	0.336069	0.376894
AVERAGE	**0.0826**	**0.1168**	**0.1133**	**0.1137**	**0.4278**	**0.5156**	**0.4421**	**0.4356**	**0.6921**	**0.8995**	**0.8208**	**0.7931**

providing equal weight to area and delay by w1 = 0.5 and w2 = 0.5 in equation 9. Table 6 presents comparative results. Table 6 presents average area, average delay, and average CoC values of all tested benchmarks area, delay, and Coc. *Detailed result is presented in Table 7, 8 and 9*. As shown in Table, 6 in the first case of the experiment, it was found that the proposed algorithm was able to achieve an average reduction of delay by 41.9%, 40.1%, and 40.3% compare to [27]–[29]. But to achieve this reduction in delay, we have to pay area overhead of 11.7%, and 13.39% with respect to [28], [29]. Whereas, in the second case proposed algorithm is able to achieve an average reduction in delay by 29.21%, 27.03%, and 27.31% compare to [27]–[29] along with the reduction in the area by 17.02%, 3.23% and 1.78% as compare to [27]–[29]. Moreover, in the third case, gave equal weight to area and delay, which means CoC has the highest priority then proposed algorithm was able to achieve an average reduction in delay by 38.2%, 36.29%, and 36.5% compared to [27]–[29] with the penalty of the area by 0.08% and 1.57% with respect to [28], [29].

TABLE 9. Comparative results between the proposed methodology and previous works in terms of delay, area, and CoC when CoC has the highest priority.

	Delay (us)				Area (um2)				CoC			
	Prop	**(23)**	**(24)**	**(25)**	**Prop**	**(23)**	**(24)**	**(25)**	**Prop**	**(23)**	**(24)**	**(25)**
Alu2	0.14	0.252	0.252	0.224	1.3828	1.424	1.36	1.388	0.622788	0.855289	0.839321	0.790752
Apex6	0.084	0.238	0.238	0.238	2.608	2.804	2.648	2.648	0.641521	1	0.972183	0.972183
B1	0.028	0.042	0.028	0.028	0.024	0.036	0.028	0.024	0.666667	1	0.722222	0.666667
C8	0.084	0.098	0.098	0.098	0.4432	0.46	0.448	0.448	0.607969	0.6793	0.672992	0.672992
cc	0.056	0.07	0.07	0.07	0.1808	0.176	0.172	0.172	0.833333	0.903392	0.89233	0.89233
cht	0.042	0.056	0.056	0.056	0.5868	0.48	0.48	0.48	0.607676	0.651678	0.651678	0.651678
CM42a	0.028	0.028	0.028	0.028	0.0808	0.084	0.072	0.072	0.959091	0.977273	0.909091	0.909091
cm85a	0.07	0.098	0.084	0.084	0.1028	0.136	0.104	0.076	0.626681	0.853148	0.692848	0.607274
Cm150a	0.07	0.126	0.126	0.084	0.1908	0.184	0.184	0.148	0.777778	0.98218	0.98218	0.721174
Cm151a	0.056	0.098	0.098	0.098	0.0996	0.168	0.092	0.08	0.464131	0.794372	0.578953	0.544939
Cm152a	0.084	0.084	0.084	0.084	0.084	0.084	0.084	0.068	0.903846	0.903846	0.903846	0.826923
Cm162a	0.07	0.098	0.098	0.098	0.178	0.184	0.164	0.164	0.838745	0.997835	0.943723	0.943723
Cm163a	0.056	0.098	0.098	0.098	0.1508	0.168	0.152	0.128	0.731341	0.996454	0.949173	0.878251
cmb	0.042	0.056	0.056	0.056	0.13	0.176	0.112	0.104	0.621782	0.835644	0.677228	0.657426
Cu	0.042	0.098	0.098	0.098	0.1592	0.184	0.16	0.156	0.554457	0.893162	0.84188	0.833333
decode	0.028	0.042	0.042	0.042	0.1104	0.112	0.112	0.112	0.69459	0.866492	0.866492	0.866492
Example2	0.098	0.14	0.14	0.126	1.0128	1.036	0.988	0.964	0.827196	0.988127	0.965511	0.904203
Frg1	0.154	0.252	0.252	0.238	0.4204	0.444	0.42	0.408	0.624234	0.836568	0.818375	0.781501
Frg2	0.154	0.21	0.196	0.182	2.2372	2.688	2.328	2.4	0.775452	0.991156	0.892043	0.871866
I1	0.028	0.084	0.084	0.084	0.1312	0.164	0.144	0.14	0.566667	1	0.939024	0.926829
lal	0.07	0.112	0.112	0.112	0.282	0.38	0.328	0.328	0.60002	0.887439	0.834421	0.834421
Majority	0.042	0.056	0.056	0.056	0.0232	0.024	0.024	0.02	0.677083	0.8125	0.8125	0.760417
mux	0.07	0.126	0.126	0.126	0.202	0.184	0.184	0.148	0.708333	0.830446	0.830446	0.741337
Pcle	0.07	0.112	0.112	0.112	0.2468	0.268	0.248	0.248	0.751334	0.976529	0.940967	0.940967
Pcler8	0.042	0.14	0.126	0.112	0.3088	0.36	0.32	0.36	0.578889	1	0.894444	0.9
Pm1	0.042	0.084	0.084	0.084	0.1344	0.18	0.14	0.128	0.623333	1	0.888889	0.855556
sct	0.056	0.084	0.084	0.084	0.1876	0.288	0.26	0.26	0.659028	1	0.951389	0.951389
Term1	0.14	0.182	0.154	0.14	0.4296	0.696	0.424	0.356	0.641954	0.933333	0.671264	0.58908
Ttt2	0.084	0.154	0.154	0.14	0.5264	0.616	0.58	0.576	0.612077	0.897112	0.873904	0.825871
X1	0.084	0.154	0.154	0.308	1.1812	1.28	1.056	1.012	0.59777	0.75	0.6625	0.895313
X2	0.07	0.098	0.098	0.084	0.1348	0.168	0.144	0.144	0.694818	0.920842	0.860721	0.789293
9symml	0.126	0.168	0.14	0.168	0.1884	0.864	0.188	0.18	0.313573	0.772727	0.336069	0.376894
AVERAGE	**0.0721**	**0.1168**	**0.1133**	**0.1137**	**0.4424**	**0.5156**	**0.4421**	**0.43562**	**0.6688**	**0.8995**	**0.8208**	**0.7931**

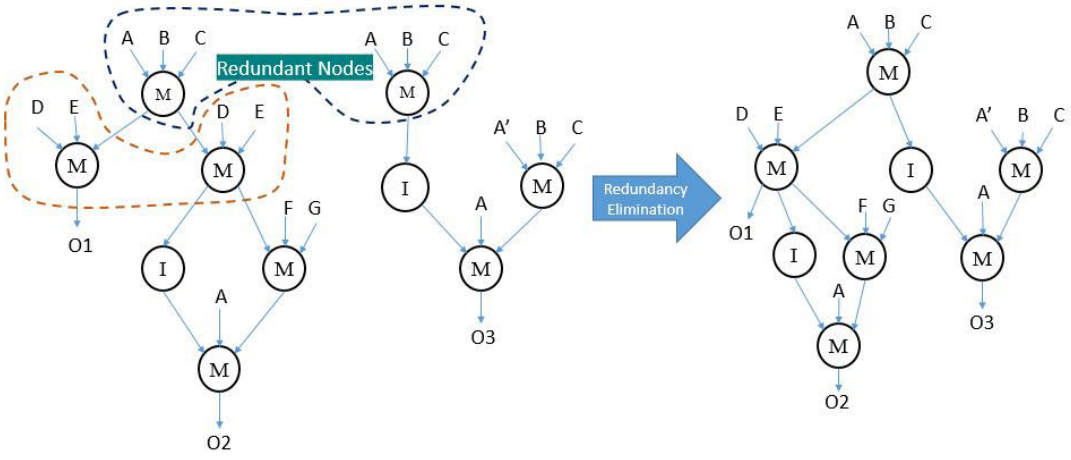


FIGURE 6. Example of redundancy elimination.

VI. CONCLUSION AND FUTURE SCOPE

The paper presented a novel heuristic-based cost-aware majority logic synthesis algorithm. A novel heuristic 'MD' was introduced to steer the MLS method. Furthermore, an extended library of majority function is also presented in this paper. Moreover, a novel post-synthesis optimization method was also demonstrated in this paper. Results showed that the proposed algorithm achieved a reduction of 22% (average) in the majority gate count and a reduction of 24% (average) in the majority level compared to recent work. Moreover, in the case of QCA technology, the proposed algorithm gained a reduction of 36% (average) in delay and a reduction of 15% (average) in CoC while paying area overhead of 2% as compared to previous methods. Near future we want to develop circuit for various image processing kernels for image encryption [46], [47], image fusion [48]based on majority logic for various applications.

REFERENCES

[1] *Semiconductor Industries Association Roadmap*. Accessed: Jan. 10, 2021. [Online]. Available: http://www.itrs2.net/

[2] F. Ciontu, C. Cucu, and B. Courtois, "Application-specific architecture for quantum cellular automata," in *Proc. 2nd IEEE Conf. Nanotechnol.*, Aug. 2002, pp. 351–354.

[3] A. O. Orlov, I. Amlani, R. K. Kummamuru, R. Ramasubramaniam, G. Toth, C. S. Lent, G. H. Bernstein, and G. L. Snider, "Experimental demonstration of clocked single-electron switching in quantum-dot cellular automata," *Appl. Phys. Lett.*, vol. 77, no. 2, pp. 295–297, Jul. 2000.

[4] A. O. Orlov, I. Amlani, G. Toth, C. S. Lent, G. H. Bernstein, and G. L. Snider, "Experimental demonstration of a binary wire for quantum-dot cellular automata," *Appl. Phys. Lett.*, vol. 74, no. 19, pp. 2875–2877, May 1999.

[5] S. Klingler, P. Pirro, T. Brächer, B. Leven, B. Hillebrands, and A. V. Chumak, "Design of a spin-wave majority gate employing mode selection," *Appl. Phys. Lett.*, vol. 105, no. 15, Oct. 2014, Art. no. 152410.

[6] L. Amaru, P.-E. Gaillardon, S. Mitra, and G. De Micheli, "New logic synthesis as nanotechnology enabler," *Proc. IEEE*, vol. 103, no. 11, pp. 2168–2195, Nov. 2015.

[7] M. G. Mankalale and S. S. Sapatnekar, "Optimized standard cells for all-spin logic," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 13, no. 2, pp. 1–22, Mar. 2017.

[8] B. Behin-Aein, D. Datta, S. Salahuddin, and S. Datta, "Proposal for an all-spin logic device with built-in memory," *Nature Nanotechnol.*, vol. 5, no. 4, pp. 266–270, Apr. 2010.

[9] M. P. Kostylev, A. A. Serga, T. Schneider, B. Leven, and B. Hillebrands, "Spin-wave logical gates," *Appl. Phys. Lett.*, vol. 87, no. 15, Oct. 2005, Art. no. 153501.

[10] T. Oya, T. Asai, T. Fukui, and Y. Amemiya, "A majority-logic device using an irreversible single-electron box," *IEEE Trans. Nanotechnol.*, vol. 2, no. 1, pp. 15–22, Mar. 2003.

[11] H. A. H. Fahmy and R. A. Kiehl, "Complete logic family using tunneling-phase-logic devices," in *Proc. 11th Int. Conf. Microelectron. (ICM)*, 2000, pp. 153–156.

[12] S. A. Wolf, J. Lu, M. R. Stan, E. Chen, and D. M. Treger, "The promise of nanomagnetics and spintronics for future logic and universal memory," *Proc. IEEE*, vol. 98, no. 12, pp. 2155–2168, Dec. 2010.

[13] R. P. Cowburn and M. Welland, "Room temperature magnetic quantum cellular automata," *Science*, vol. 287, no. 5457, pp. 1466–1468, Feb. 2000.

[14] S. Breitkreutz, J. Kiermaier, I. Eichwald, X. Ju, G. Csaba, D. Schmitt-Landsiedel, and M. Becherer, "Majority gate for nanomagnetic logic with perpendicular magnetic anisotropy," *IEEE Trans. Magn.*, vol. 48, no. 11, pp. 4336–4339, Nov. 2012.

[15] J. Zhu, L. Zhang, S. Dong, and E. Wang, "Four-way junction-driven DNA strand displacement and its application in building majority logic circuit," *ACS Nano*, vol. 7, no. 11, pp. 10211–10217, Nov. 2013.

[16] A. K. George and H. Singh, "Three-input majority gate using spatially localised DNA hairpins," *Micro Nano Lett.*, vol. 12, no. 3, pp. 143–146, Mar. 2017.

[17] D. E. Nikonov and I. A. Young, "Overview of beyond-CMOS devices and a uniform methodology for their benchmarking," *Proc. IEEE*, vol. 101, no. 12, pp. 2498–2533, Dec. 2013.

[18] S. B. Akers, "Synthesis of combinational logic using three-input majority gates," in *Proc. 3rd Annu. Symp. Switching Circuit Theory Log. Design (SWCT)*, 1962, pp. 149–158.

[19] H. S. Miller and R. O. Winder, "Majority-logic synthesis by geometric methods," *IRE Trans. Electron. Comput.*, vol. EC-11, no. 1, pp. 89–90, Feb. 1962.

[20] G. D. Hachtel and F. Somenzi, *Logic Synthesis and Verification Algorithms*. New York, NY, USA: Springer, 2007.

[21] G. De Micheli, *Synthesis and Optimization of Digital Circuits*. New York, NY, USA: McGraw-Hill, 1994.

[22] S. Muroga, *Threshold Logic and Its Applications*. New York, NY, USA: Wiley, 1971.

[23] R. Zhang, K. Walus, W. Wang, and G. A. Jullien, "A method of majority logic reduction for quantum cellular automata," *IEEE Trans. Nanotechnol.*, vol. 3, no. 4, pp. 443–450, Dec. 2004.

[24] K. Walus, G. Schulhof, G. A. Jullien, R. Zhang, and W. Wang, "Circuit design based on majority gates for applications with quantum-dot cellular automata," in *Proc. Conf. Rec. 38th Asilomar Conf. Signals, Syst. Comput.*, vol. 2, 2004, pp. 1354–1357.

[25] Z. Huo, Q. Zhang, S. Haruehanroengra, and W. Wang, "Logic optimization for majority gate-based nanoelectronic circuits," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2006, p. 4.

[26] S. Rai, "Majority gate based design for combinational quantum cellular automata (QCA) circuits," in *Proc. 40th Southeastern Symp. Syst. Theory (SSST)*, Mar. 2008, pp. 222–224.

[27] R. Zhang, P. Gupta, and N. K. Jha, "Majority and minority network synthesis with application to QCA-, SET-, and TPL-based nanotechnologies," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 7, pp. 1233–1245, Jul. 2007.

[28] K. Kong, Y. Shang, and R. Lu, "An optimized majority logic synthesis methodology for quantum-dot cellular automata," *IEEE Trans. Nanotechnol.*, vol. 9, no. 2, pp. 170–183, Mar. 2010.

[29] P. Wang, M. Y. Niamat, S. R. Vemuru, M. Alam, and T. Killian, "Synthesis of majority/minority logic networks," *IEEE Trans. Nanotechnol.*, vol. 14, no. 3, pp. 473–483, May 2015.

[30] M. R. Bonyadi, S. M. R. Azghadi, N. M. Rad, K. Navi, and E. Afjei, "Logic optimization for majority gate-based nanoelectronic circuits based on genetic algorithm," in *Proc. Int. Conf. Electr. Eng.*, Apr. 2007, pp. 1–5.

[31] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A. Sangiovanni-Vincentelli, "SIS: A system for sequential circuit synthesis," Dept. EECS, Univ. California, Berkeley, CA, USA, Tech. Rep., 1992.

[32] L. Amarú, P.-E. Gaillardon, and G. De Micheli, "BDS-MAJ: A BDD-based logic synthesis tool exploiting majority logic decomposition," in *Proc. 50th Annu. Design Autom. Conf.*, 2013, pp. 1–6.

[33] S. B. Z. Vransic, *Fundamentals of Digital Logic With VHDL Design*, 2nd ed. New York, NY, USA: McGraw-Hill, 2005, pp. 315–724.

[34] V. K. Mishra and H. Thapliyal, "Heuristic based majority/minority logic synthesis for emerging technologies," in *Proc. 30th Int. Conf. VLSI Design 16th Int. Conf. Embedded Syst. (VLSID)*, Jan. 2017, pp. 295–300.

[35] P. D. Tougaw and C. S. Lent, "Logical devices implemented using quantum cellular automata," *J. Appl. Phys.*, vol. 75, no. 3, pp. 1818–1825, 1994.

[36] R. Akeela and M. D. Wagh, "A five-input majority gate in quantum-dot cellular automata," in *Proc. NSTI Nanotech*, vol. 2, 2011, pp. 978–981.

[37] K. Navi, S. Sayedsalehi, R. Farazkish, and M. R. Azghadi, "Five-input majority gate, a new device for quantum-dot cellular automata," *J. Comput. Theor. Nanosci.*, vol. 7, no. 8, pp. 1546–1553, Aug. 2010.

[38] K. Walus, T. J. Dysart, G. A. Jullien, and R. A. Budiman, "QCADesigner: A rapid design and simulation tool for quantum-dot cellular automata," *IEEE Trans. Nanotechnol.*, vol. 3, no. 1, pp. 26–31, Mar. 2004.

[39] S. S. Farahani, R. Zarhoun, M. H. Moaiyeri, and K. Navi, "An efficient CNTFET-based 7-input minority gate," 2013, *arXiv:1303.2175*. [Online]. Available: http://arxiv.org/abs/1303.2175

[40] R. Brayton and A. Mishchenko, "ABC: An academic industrial-strength verification tool," in *Proc. Int. Conf. Comput. Aided Verification*. Pacific Grove, CA, USA: Springer, 2010, pp. 24–40.

[41] (2010). *ABC: A System for Sequential Synthesis and Verification (2007)*. [Online]. Available: http://www.eecs.berkeley.edu/alanmi/abc

[42] L. Amaru, P.-E. Gaillardon, A. Chattopadhyay, and G. De Micheli, "A sound and complete axiomatization of majority- logic," *IEEE Trans. Comput.*, vol. 65, no. 9, pp. 2889–2895, Sep. 2016.

[43] S. Yang, "Logic synthesis and optimization benchmarks user guide: Version 3.0," Microelectron. Center North Carolina, 1991, pp. 502–508.
[44] C. S. Lent and P. D. Tougaw, "A device architecture for computing with quantum dots," *Proc. IEEE*, vol. 85, no. 4, pp. 541–557, Apr. 1997.
[45] K. Navi, R. Farazkish, S. Sayedsalehi, and M. R. Azghadi, "A new quantum-dot cellular automata full-adder," *Microelectron. J.*, vol. 41, no. 12, pp. 820–826, Dec. 2010.
[46] M. Kaur, D. Singh, and R. S. Uppal, "Parallel strength Pareto evolutionary algorithm-II based image encryption," *IET Image Process.*, vol. 14, no. 6, pp. 1015–1026, May 2020.
[47] M. Kaur, D. Singh, and V. Kumar, "Color image encryption using minimax differential evolution-based 7D hyper-chaotic map," *Appl. Phys. B, Lasers Opt.*, vol. 126, no. 9, pp. 1–19, Sep. 2020.
[48] M. Kaur and D. Singh, "Multi-modality medical image fusion technique using multi-objective differential evolution based deep neural networks," *J. Ambient Intell. Hum. Comput.*, vol. 12, no. 2, pp. 2483–2493, Feb. 2021.

VIPUL KUMAR MISHRA received the Ph.D. degree from the Indian Institute of Technology Indore, India, in 2015. He was a Postdoctoral Scholar with the University of Kentucky, from 2015 to 2016. He is currently an Assistant Professor with the Department of Computer Science and Engineering, Bennett University, India. His research interests include machine learning, deep learning, neural network optimization, and design space exploration. He is currently working to make deep learning model faster, smaller, and power efficient. He is also working in applied deep learning in computer vision.

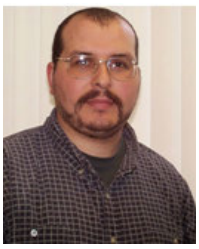
MAYANK DIXIT is currently a Research Scholar with the Department of Computer Science Engineering, Bennett University, Greater Noida, India, and also working as an Assistant Professor with Department of Computer Science and Engineering, Galgotias College of Engineering and Technology, Greater Noida. Formerly, he has worked with companies like Hewlett Packard and Tata Consultancy Services for around six years in the area of IT infrastructure and automation. His research interest includes various object extraction from remote sensing images.

TEJALAL CHOUDHARY received the Bachelor of Engineering (B.E.) degree in computer science and engineering from the Government Engineering College, Jabalpur, India, in 2006, and the Master of Engineering (M.E.) degree in computer engineering (specialization in software engineering) from the Institute of Engineering and Technology (IET), Devi Ahilya Vishwavidyalaya, Indore, India, in 2014. He is currently pursuing the Ph.D. degree with Bennett University, Greater Noida, India. His research interests include computer vision, machine learning, deep learning, and model compression.

ANURAG GOSWAMI is currently an Assistant Professor with the Department of Computer Science and Engineering, Bennett University, India. His research interest includes improving software quality via empirical evaluation. He is also involved in research that enables reduction of the skill gap between academia and industry. He has collaborated with multiple researchers across the globe and is currently researching in the area of machine learning to understand that factors where machine learning could help in the improvement of software quality.

MANJIT KAUR received the Master of Engineering degree in information technology from Panjab University, Chandigarh, India, in 2011, and the Ph.D. degree in image processing from the Thapar Institute of Engineering and Technology, Patiala, India, in 2019. She is currently working as an Assistant Professor with the School of Engineering and Applied Sciences, Bennett University, Greater Noida, India. She has published more than 27 SCI/SCIE indexed articles. Her research interests include wireless sensor networks, digital image processing, and meta-heuristic techniques.

OMAR CHEIKHROUHOU received the B.S., M.S., and Ph.D. degrees in computer science from the National School of Engineers of Sfax, in March 2012. His Ph.D. deals with the security in wireless sensor networks and more precisely in "Secure Group Communication in Wireless Sensor Networks." He is currently an Assistant Professor with the College of Computer and Information Technology, Taif, Saudi Arabia. He is also a member of CES Laboratory (computer and embedded system), National School of Engineers, University of Sfax. He has several publications in several high-quality international journals and conferences. His research interests include wireless sensor networks, cybersecurity, edge computing, blockchain, and multi-robot system coordination. He has received some awards, including the Governor Prize from the Governor of Sfax, in 2005.



HABIB HAMAM (Senior Member, IEEE) received the B.Eng. and M.Sc. degrees in information processing from the Technical University of Munich, Germany, in 1988 and 1992, respectively, and the Ph.D. degree in physics and applications in telecommunications from the Université de Rennes I conjointly with France Telecom Graduate School, France, in 1995. He obtained a Postdoctoral Diploma "Accreditation to Supervise Research in Signal Processing and Telecommunications" from the Université de Rennes I, in 2004. He was a Canada Research Chair holder in Optics in Information and Communication Technologies, from 2006 to 2016. He is currently a Full Professor with the Department of Electrical Engineering, Université de Moncton. His research interests include optical telecommunications, wireless communications, diffraction, fiber components, RFID, information processing, data protection, COVID-19, and deep learning. He is a Senior Member of OSA, and a registered Professional Engineer in New-Brunswick. He is the Editor-in-Chief of CIT-Review and an Associate Editor of the *IEEE Canadian Review*.