Fast Logic Synthesis for RRAM-based In-Memory Computing using Majority-Inverter Graphs

Saeideh Shirinzadeh*, Mathias Soeken*†‡, Pierre-Emmanuel Gaillardon§, Rolf Drechsler*†
*Department of Mathematics and Computer Science, University of Bremen, Germany
†Cyber-Physical Systems, DFKI GmbH, Bremen, Germany
‡Integrated Systems Laboratory, EPFL, Lausanne, Switzerland
§University of Utah, Salt Lake City, UT, USA
{saeideh,msoeken,drechsle}@cs.uni-bremen.de, pierre-emmanuel.gaillardon@utah.edu

***Abstract*—Resistive Random Access Memories (RRAMs) have gained high attention for a variety of promising applications especially the design of non-volatile in-memory computing devices. In this paper, we present an approach for the synthesis of RRAM-based logic circuits using the recently proposed *Majority-Inverter Graphs* (MIGs). We propose a bi-objective algorithm to optimize MIGs with respect to the number of required RRAMs and computational steps in both MAJ-based and IMP-based realizations. Since the number of computational steps is recognized as the main drawback of the RRAM-based logic, we also present an effective algorithm to reduce the number of required steps. Experimental results show that the proposed algorithms achieve higher efficiency compared to the general purpose MIG optimization algorithms, either in finding a good trade-off between both cost metrics or reducing the number of steps. In comparison with the RRAM-based circuits implemented by the state-of-the-art approaches using other well-known data structures the number of required computational steps obtained by our proposed MIG-oriented synthesis approach for large benchmark circuits is reduced up to factor of 26. This strong gain comes from the use of MIGs that provide an efficient and intrinsic representation for RRAM-based computing—particularly in MAJ-based realizations—and the use of techniques proposed for optimization.**

I. Introduction

While the resistive switching phenomena was known from 1960s [1], it did not gain much attention until the late 1990s. So far, various metal oxides using different materials with the resistive switching characteristics between two high and low resistance values are fabricated that are called *Resistive Random Access Memory* (RRAMs) [1]. RRAMs are of high interest due to their promising applications in non-volatile memory design [2], [3], digital and analog programmable systems [4], [5], [6], and neuromorphing computing [7]. In 1971, Chua [8] derived equations describing a forth passive circuit element from symmetry which he called *memristor*, short for memory resistor. However, some researchers claim differences between RRAMs and memristors, the resistive switching property that is used in this work is shared by both devices [9]. Since different RRAMs have already been fabricated and their functionality is proven, here we prefer to use the term RRAM.

Material Implication (IMP) can be executed by RRAMs to synthesize Boolean functions. This enables designing memories with computing capability, however, the number of computational steps is a serious drawback of implication logic [10]. Using data structures such as *Binary Decision Diagrams* (BDDs) [11] and *And-Inverter Graphs* (AIGs) [12] has been previously proposed for optimization of RRAM-based circuits. However, both approaches presented in [11], [12] require a high number of computational steps.

A novel homogeneous logic representation structure, *Majority-Inverter Graph* (MIG) was proposed in [13] that uses the majority function $M(x, y, z) = x \cdot y + x \cdot z + y \cdot z$ together with negation as the only logic operations. MIGs have a high flexibility in depth optimization that enables design of high speed logic circuits and FPGA implementations [14]. In comparison with the well-known data structures BDDs and AIGs, MIGs have experimentally shown better results in logic optimization, especially in propagation delay [13]. In particular, MIGs are highly qualified for logic synthesis of RRAM-based circuits since they can efficiently execute the built-in *resistive majority* operation in RRAMs [15].

In this paper, we propose an approach to implement fast circuits with RRAMs using MIG-based logic synthesis. In order to map MIGs to the equivalent RRAM-based circuits, we first present two realizations for majority gate: (i) a realization based on IMP that is also used by previous works using BDDs [11] and AIGs [12], and (ii) a realization that exploits the built-in resistive majority property of RRAMs denoted by MAJ. Then, we propose two MIG optimization algorithms for synthesis of RRAM-based logic circuits: (i) a multi-objective optimization algorithm to reduce the number of required RRAMs and computational steps representing area and delay of the resulting circuits, respectively, and (ii) an optimization algorithm tailored to reduce the number of computational steps as the main concern of RRAM-based logic design.

The proposed optimization algorithms for RRAM-based logic design gain higher efficiency in comparison to the conventional MIG optimization techniques. Experiments confirm the superiority of MIGs to BDDs and AIGs in RRAM-based circuit design and the effectiveness of the proposed optimization algorithms. According to the results, the circuits represented and optimized by the proposed MIG-based synthesis approach for large benchmark functions are about 26 times faster than the RRAM-based circuits implemented by the BDD-based and AIG-based synthesis approaches presented in [11], [12].

II. Background

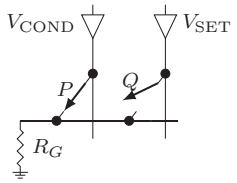
A. Logic operations for RRAM-based circuit design

We present two logic operations, IMP and MAJ, for synthesis of RRAM-based circuits.

1) Material implication (IMP): *Material Implication* (IMP) and FALSE operation, i.e. assigning the output to logic 0, are sufficient to express any Boolean function [16]. Fig. 1 shows the implementation of an IMP gate with two RRAMs that are represented by the symbols of memristors as proposed in [16]. P and Q designate two resistive switches connected to a load resistor R_G. Three voltage levels V_{SET}, V_{COND}, and V_{CLEAR} are applied to the RRAMs to execute IMP and FALSE operations by switching between low-resistance (logic 1) or high-resistance (logic 0) states.

The FALSE operation can be performed by applying V_{CLEAR} to the RRAM. The RRAM can be switched to logic 1 by applying a voltage larger than a threshold V_{SET} to its

The work has been partly supported by the University of Bremen's graduate school SyDe, funded by the German Excellence Initiative and by the Swiss National Science Foundation project number 200021 146600.

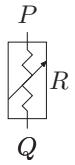


(a)

p	q	q'
0	0	1
0	1	1
1	0	0
1	1	1

(b)

Fig. 1. IMP operation. (a) Implementation of IMP using RRAMs. (b) Truth table for IMP ($q' \leftarrow p \,\mathrm{IMP}\, q = \overline{p} + q$) [16]



P	Q	R	R'
0	0	0	0
0	1	0	0
1	0	0	1
1	1	0	0

$R' = P \cdot \overline{Q}$

P	Q	R	R'
0	0	1	1
0	1	1	0
1	0	1	1
1	1	1	1

$R' = P + \overline{Q}$

Fig. 2. The intrinsic majority operation within an RRAM

voltage driver. To execute IMP, two voltage levels V_{SET} and V_{COND} are applied to the switches P and Q simultaneously. The magnitude of V_{COND} is smaller than the required threshold to change the state of the switch. However, the interaction of V_{SET} and V_{COND} can execute IMP according to the current states of the switches, such that switch Q is set to 1 if $P = 0$ and it retains its current state if $p = 1$ [16].

*2) Built-in majority operation (*MAJ*):* RRAMs are two-terminal devices which internal resistance R can be switched between two logic states 0 and 1 designating low and high resistance values, respectively. Denoting the top and bottom terminals by P and Q, the device can be switched with a negative or positive voltage V_{PQ} based on the polarity and location of dopants. Assuming the voltage levels V_{SET}, V_{CLEAR}, and V_{COND} respectively correspond to logic statements ($P = 1, Q = 0$), ($P = 0, Q = 1$), and ($P = Q$), the truth tables shown in Fig. 2 explain the next sate of the switch (R') based on P, Q, and the current state (R). The built-in majority operation described in Fig. 2 can be formally expressed as the following [15]:

$$\begin{aligned} R' &= \mathrm{MAJ}(P, Q, R) = (P \cdot \overline{Q}) \cdot \overline{R} + (P + \overline{Q}) \cdot R \\ &= P \cdot R + \overline{Q} \cdot R + P \cdot \overline{Q} \cdot \overline{R} \\ &= P \cdot R + \overline{Q} \cdot R + P \cdot \overline{Q} \cdot R + P \cdot \overline{Q} \cdot \overline{R} \\ &= P \cdot R + \overline{Q} \cdot R + P \cdot \overline{Q} \\ &= M(P, \overline{Q}, R) \end{aligned}$$

Therefore, using MAJ a majority gate for three variables x y, and z can be simply executed after preloading the RRAMs and negation of y.

B. Majority-inverter graphs

The Boolean algebra of MIGs was proposed in [13]. The following set (Ω) includes the primitive transformations.

$$\Omega \begin{cases} \textbf{Commutativity} - \Omega.C \\ M(x,y,z) = M(y,x,z) = M(z,y,x) \\ \textbf{Majority} - \Omega.M \\ \begin{cases} \mathrm{if}(x = y) : M(x,y,z) = x = y \\ \mathrm{if}(x = \overline{y}) : M(x,y,z) = z \end{cases} \\ \textbf{Associativity} - \Omega.A \\ M(x,u,,M(y,u,z)) = M(z,u,M(y,u,x)) \\ \textbf{Distributivity} - \Omega.D \\ M(x,y,M(u,v,z)) = M(M(x,y,u),M(x,y,v),z) \\ \textbf{Inverter Propagation} - \Omega.I \\ \overline{M}(x,y,z) = M(\bar{x},\bar{y},\bar{z}) \end{cases}$$

It was proven in [13] that any MIG can be transformed to another logically equivalent MIG using only Ω axioms. It means that reaching a desired MIG optimized with respect to the considered cost metric is possible by applying Ω, however, the length of transformation sequence might be impractical. To solve this problem, a more advanced set of transformations derived from the basic rules in Ω was proposed in [13] which was shown by Ψ. The following set includes those axioms of Ψ that are used in this work.

$$\Psi \begin{cases} \textbf{Relevance} - \Psi.R \\ M(x,y,z) = M(x,z,z_{x/\bar{y}}) \\ \textbf{Complementary Associativity} - \Psi.C \\ M(x,u,M(y,\bar{u},z)) = M(x,u,M(y,x,z)) \end{cases}$$

Where $z_{x/\bar{y}}$ means replacing x with $\bar{y}$.

III. RRAM-based In-memory Computing Design with MIGs

This section presents the proposed realizations for majority gate with RRAMs and their corresponding MIG mapping methodology. Then, we propose MIG optimization algorithms with respect to area, i.e., the number of MIG nodes, depth, i.e., the number of levels in the graph, and the number of RRAMs and computational steps.

A. Realization of majority gate using RRAMs

1) IMP*-based realization:* The proposed IMP-based realization of a majority gate is shown in Fig. 3. It requires six RRAMs and ten sequential steps. RRAMs shown by X, Y, and Z are loaded by input variables and the remaining three RRAMs A, B, and C are required for retaining the intermediate results and the final output. The corresponding steps for executing the majority function are as follows:

01: $X = x, Y = y, Z = z$
$A = 0, B = 0, C = 0$
02: $a \leftarrow x \,\mathrm{IMP}\, a = \bar{x}$
03: $b \leftarrow y \,\mathrm{IMP}\, b = \bar{y}$
04: $y \leftarrow a \,\mathrm{IMP}\, y = x + y$
05: $b \leftarrow x \,\mathrm{IMP}\, b = \bar{x} + \bar{y}$
06: $c \leftarrow y \,\mathrm{IMP}\, c = \overline{x + y}$
07: $c \leftarrow z \,\mathrm{IMP}\, c = \overline{x \cdot z + y \cdot z}$
08: $a = 0$
09: $a \leftarrow b \,\mathrm{IMP}\, a = x \cdot y$
10: $a \leftarrow c \,\mathrm{IMP}\, a = x \cdot y + y \cdot z + x \cdot z$

During the steps shown above, the initial values stored in two out of the six RRAMS remain unchanged while the others are either cleared or used to save the outputs of the implications. In the first step, the input variables are loaded and the other RRAMs are assigned FALSE for the next operations. Another FALSE operation is also performed in step 8, to clear an RRAM which is not required anymore for inverting an intermediate result. Finally, the Boolean function representing a majority gate is executed by implying results from the seventh and ninth step.

2) MAJ*-based realization:* It is obvious that the MAJ-based majority gate can be realized with smaller number of RRAMs and computational steps due to benefiting from the discussed built-in majority property. Using MAJ, the majority gate will require only four RRAMs placed in the same structure shown in Fig. 3 such that the bottom electrodes of the switches are electrically connected via a horizontal nanowire and the switching can be done by applying the three discussed voltage levels to the top electrodes. Furthermore, the majority function can be executed within only three steps carrying out simple operations. The MAJ-based computational steps for the proposed RRAM-based realization are:

01: $X = x, Y = y, Z = z, A = 0$
02: $P_A = 1, Q_A = y, R_A = 0 \rightarrow R'_A = \bar{y}$
03: $P_Z = x, Q_Z = \bar{y}, R_Z = z \rightarrow R'_Z = M(x,y,z)$

In the first step, the initial values of input variables as well as an additional RRAM are loaded by applying V_{SET} or

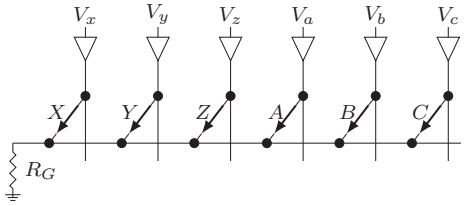


Fig. 3. Proposed realization of an IMP-based majority gate using RRAMs

TABLE I. THE COST METRICS OF MIG IMPLEMENTATION USING RRAMS

Symbol	Definition	Value
$\mathbf{N}_i$	No. of nodes in the i^{th} level of the MIG	Given
$\mathbf{C}_i$	No. of ingoing complemented edges in the i^{th} level of the MIG	Given
D	The depth of the MIG	Given
L	No. of MIG levels with ingoing complemented edges	Given
R	No. of RRAMs	$\max_{0\leq i\leq D}(K\cdot N_i + C_i)$ IMP : $K = 6$, MAJ : $K = 4$
S	No. of computational steps	$K\cdot D + L$ IMP : $K = 10$, MAJ : $K = 3$

V_{CLEAR} to their voltage divers. Step 2 executes the required NOT operation in RRAM A. This can be done with applying appropriate voltage levels V_{SET} or V_{COND} to switch A, for cases $y = 0$ and $y = 1$, respectively. In the last step, the majority function is executed by use of MAJ at RRAM Z by applying any of the three voltage levels corresponding the difference between logic states of x and $\bar{y}$.

B. Design methodology

Although both of the proposed realizations impose sequential circuit implementations, they allow a reduction in area by reusing RRAMs released from previous computations. In our proposed synthesis approach, we only consider one MIG level each time, such that the employed RRAMs to evaluate the level can be used later for the next levels. Starting from the input of the graph, the RRAMs in a level are released when all the required computational steps are done. Then, the RRAMs are reused for the upper level and this procedure is continued until the target function is evaluated. Such an implementation requires as many majority gates as the maximum number of nodes in any level of the MIG. Hence, depending on the use of IMP or MAJ in the realization, the corresponding number of RRAMs and steps for synthesizing the MIG is six or four times the number of required majority gates and ten or three times the number of levels, respectively. However, still some additional RRAMs are needed in the presence of complemented edges. Table I shows the number of RRAMs and computational steps of the resulting RRAM-based circuits. For every complemented edge in the graph a NOT gate is required. The negation can be executed by either an IMP or MAJ operation with logic 0 as shown in second step of both realizations. This will require one extra RRAM to be loaded by 0 that can be done in parallel with the data loading step and an additional step for executing the imply operation. Since the implementation starts from the input of MIG, the ingoing complemented edges of any level should be first inverted for a correct evaluation. It is obvious that the required implications for all complemented edges in a level can be executed simultaneously. In other words, the additional steps required for complemented edges are equal to the number of MIG levels with ingoing complemented edges. Similarly, the total number of RRAMs required for the synthesis of the whole graph is equal to the maximum of six (IMP) or four (MAJ) times the number of nodes in the level plus the number of ingoing complemented edges over all MIG levels.

C. MIG optimization for RRAM-based logic circuits

In general, MIG optimization is performed by applying a set of valid transformations to an existing MIG to find an equivalent MIG that is more efficient with respect to the considered cost metrics. MIG optimization in terms of area and delay aims at finding the best trade-off between the depth and the size of the graph, i.e., the number of nodes. Using RRAMs for implementation, the metrics determining area and delay depend on a combination of MIG features that some of them are not intended in conventional area and depth optimization. However, a reduction in area and especially depth might lower costs of an RRAM-based implementation. Thus, specific optimization techniques are required to find an optimum MIG with respect to the number of RRAMs and computational steps. In this section, we first present conventional area and depth optimization algorithms for standard implementation of MIGs to show why different optimization techniques are required for RRAM-based implementation. Then, we present the two proposed MIG optimization algorithms tackling the cost metrics of logic synthesis with RRAMs. The first proposed algorithm optimizes MIGs with respect to both objectives simultaneously, while the other one aims at reducing the number of computational steps, which is often regarded to be more important compared to the number of RRAMs.

1) Area optimization: The framework for area optimization given in Alg. 1 is based on conventional MIG area optimization algorithm proposed in [13]. Using *eliminate* ($\Omega.M$; $\Omega.D_{R\rightarrow L}$) some of the MIG nodes can be removed by repeatedly applying majority rule ($\Omega.M$) and distributivity from right to left ($\Omega.D_{R\rightarrow L}$) to the entire MIG. Assuming x, y, z, u and v as input variables $\Omega.D_{R\rightarrow L}$ transforms $M(M(x,y,u),M(x,y,v),z)$ to $M(x,y,M(u,v,z))$ which means the total number of nodes has decreased from three to two. In order to enable further reduction in the number of nodes, the MIG is reshaped by use of associativity axioms $\Omega.A$, $\Psi.C$, which allow to move the variables between adjacent levels. Then, *eliminate* is applied again to optimize the size of the newly arranged MIG. The area optimization algorithm can be iterated for a maximum number of cycles called *effort*. From the point of area in an RRAM-based circuit, although Alg. 1 can reduce the number of physical RRAMs by removing unnecessary nodes, it does not address the issue of complemented edges that are important in both aforementioned cost metrics.

Alg. 1 Conventional MIG area optimization (based on [13])

for (cycles = 0; cycles $<$ effort; cycles++) **do**
  $\Omega.M$; $\Omega.D_{R\rightarrow L}$;
  $\Omega.A$; $\Psi.C$;
  $\Omega.M$; $\Omega.D_{R\rightarrow L}$; } eliminate
end for

2) Depth optimization: In general, the depth of the graph is of high importance in MIG optimization to lower the latency of the resulting circuits. Alg. 2 is structurally similar to the MIG depth optimization procedure proposed in [13] with slightly shorter iterations. The depth of the MIG can be reduced by pushing the critical variable with the longest arrival time to upper levels. This can be possible by the process *push-up* shown in Alg. 2. *Push-up* includes majority, distributivity, and associativity axioms. It is obvious that the majority rule may reduce depth by removing unnecessary nodes. Applying distributivity from left to right ($\Omega.D_{L\rightarrow R}$) such that $M(x,y,M(u,v,z))$ is transformed to $M(M(x,y,u),M(x,y,v),z)$ may also result in an MIG with smaller depth. If either x or y is the critical variable with the latest arrival, distributivity cannot reduce the depth of $M(x,y,M(u,v,z))$. However, if z is the critical variable, applying $\Omega.D_{L\rightarrow R}$ will reduce the depth of MIG by

pushing z one level up. In the cases that the associativity rules ($\Omega.A$, $\Psi.C$) are applicable, the depth can be reduced by one if the axioms move the critical variable to the upper level. After performing *push-up*, the relevance axiom ($\Psi.R$) is applied to replace the reconvergent variables that might provide further possibility of depth reduction for another *push-up*.

Although Alg. 2 decreases the number of computational steps in an RRAM-based circuit, it does not aim for the issue of complemented edges. Moreover, the depth reduction by Alg. 2 is performed at a cost of area. $\Omega.D_{L\to R}$ adds one extra node to the graph. This may increase the area of the resulting RRAM-based circuit if the size of the critical level, i.e., the level with the maximum number of required RRAMs, is increased. $\Omega.A$ and $\Psi.C$ can also have a similar effect on the maximum level size by moving one node to the critical level. A simple example for this is applying $\Omega.A$ to $M(x,u,M(y,u,M(p,q,r)))$ that has a depth of three and one node in each level. The transformation results in $M(M(p,q,r),u,M(y,u,x))$ of depth two and two nodes in the lower level. Although the late arrival variable ($M(p,q,r)$) is pushed up, the number of nodes in one level, that might be the critical level, has increased from one to two. This effect is not of interest for RRAM-based implementation of MIGs, however using $\Psi.C$ might be with a positive spin in this case because of the possibility of reducing the number of complemented edges.

Alg. 2 Conventional MIG depth optimization (based on [13])

for (cycles = 0; cycles $<$ effort; cycles++) **do**
  $\Omega.M$; $\Omega.D_{L\to R}$; $\Omega.A$; $\Psi.C$; $\Psi.R$; $\Omega.M$; $\Omega.D_{L\to R}$; $\Omega.A$; $\Psi.C$; } push-up
end for

3) Multi-obtective optimization: None of the algorithms explained above suggest a solution for the issue of complemented edges that contain an important part of both cost metrics in RRAM-based circuits. Moreover, a single-objective MIG optimization algorithm considers either area or delay that leads to circuits worsened with respect to the other objective. Hence, we propose a multi-objective MIG optimization algorithm to obtain efficient RRAM-based logic circuits with a good trade-off between both objectives. The proposed multi-objective MIG optimization algorithm for RRAM-based logic design includes a combination of conventional area and depth optimization algorithms besides techniques tackling complemented edges from both aspects of area and delay. The algorithm starts with applying *push-up* to obtain a smaller depth. Then, the complemented edges are aimed by applying an extension of axiom inverter propagation from right to left ($\Omega.I_{R\to L}$) for the condition that the considered node has at least two outgoing complemented edges. The three cases satisfying this condition and their equivalent majority gates are shown below and discussed in the following considering their effect on both cost metrics.

$$M(\bar{x},\bar{y},\bar{z}) = \overline{M}(x,y,z) \quad (1)$$

$$\overline{M}(\bar{x},\bar{y},z) = M(x,y,\bar{z}) \quad (2)$$

$$M(\bar{x},\bar{y},z) = \overline{M}(x,y,\bar{z}) \quad (3)$$

In the first case, the ingoing complemented edges of the gate are decreased from three to zero, while one complement attribute is moved to the upper level, i.e., the level including the output of the gate. Assuming that the current level, i.e., the level including the ingoing edges, is the critical level with the maximum number of required RRAMs, this case is favorable for area optimization. However, if the upper level is the critical level, the number of required RRAMs will increase by only one. Similar scenarios exist for the two other cases, although the last case might be less interesting because the number of complemented edges in both levels is changed equally by one. That means a penalty of one is possible as the cost for a reduction of one, while transformations (1) and (2) may result in RRAM reductions of three and two, respectively.

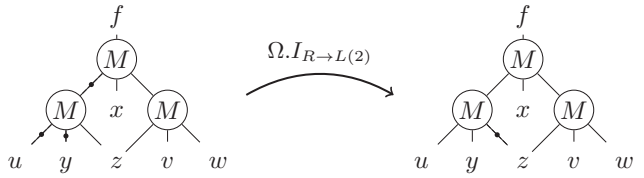


Fig. 4. Applying an extension of $\Omega.I_{R\to L}$ to reduce the extra RRAMs and steps caused by complemented edges

To reduce the number of computational steps, the number of levels possessing complemented edges should be reduced. Depending on the presence of complemented edges by other gates in both levels, the two first transformations given above might reduce or increase the number of steps or even leave it unchanged. Case (1) is beneficial if the upper level already has complement edges and also the transformation removes all the complemented edges from the current level. It might be also neutral if none of the levels are going to be improved to a complement-free level. The worst case occurs when moving the complement attribute to the upper level increments the number of levels with complement edges. Similar arguments can be made for the remaining cases. However, case (2) is more favorable because it never adds a level with complemented edges and case (3) can not be advantageous because it can never release a level from complemented edges.

Fig. 4 shows a simple MIG that is applicable to transformation (2) ($\Omega.I_{R\to L(2)}$). The transformation has released one level of the MIG from the complement attribute (black dot), which results in a smaller number of computational steps. Furthermore, as a result of removing one complemented edge from the critical level, the required number of RRAMs is decreased by one.

Alg. 3 Mult-objective optimization for RRAM costs

for (cycles = 0; cycles $<$ effort; cycles++) **do**
  $\Omega.M_{L\to R}$; $\Omega.D_{L\to R}$; $\Omega.A$; $\Psi.C$; $\Omega.I_{R\to L(1-3)}$; $\Omega.M_{L\to R}$; $\Omega.D_{L\to R}$; $\Omega.A$; $\Psi.C$; $\Omega.A$; $\Omega.D_{R\to L}$; } push-up
end for

After applying inverter propagation for the aforementioned conditions ($\Omega.I_{R\to L(1-3)}$), the MIG is also reshaped and more chances for reducing the depth might be created. Thus, *push-up* is applied to the entire MIG again to reduce the number of steps as much as possible. In the last step, the number of RRAMs are reduced to make a trade-off between both objectives. Applying $\Omega.A$, some of changes by *push-up* that have increased the maximum level size can be undone. Finally, distributivity from right to left ($\Omega.D_{R\to L}$) is applied to the graph to reduce the number of nodes in levels.

D. Step optimization

Due to the importance of latency in logic synthesis, and the issue of sequential implementation in RRAM-based circuits, we propose an MIG optimization algorithm for reducing the number of computational steps. In the proposed step optimization algorithm, two axioms of inverter propagation are applied to the MIG after *push-up*. First, only the axiom presented by case (1), i.e., the base rule of inverter propagation from right to left

($\Omega.I_{R\rightarrow L}$), is applied to the entire MIG to lower the number of levels with complemented edges. Since the transformation moves one complement attribute to the upper level, it might create new inverter propagation candidates for the all three discussed cases if the upper level already has one or two ingoing complemented edges. Hence, we apply $\Omega.I_{R\rightarrow L(1-3)}$ again to ensure maximum coverage of complemented edges. Although case (3) can not reduce the number of steps, it is not excluded from $\Omega.I_{R\rightarrow L(1-3)}$ due to its effect on balancing the levels' sizes. Finally, *push-up* is applied to the MIG to reduce the depth more if new opportunities are generated. It should be noted that the number of computational steps is mainly determined by the MIG depth. In fact, in the worst case caused by complemented edges, the total number of steps would be equal to seven times the number of levels, i.e., the MIG depth. Nonetheless, we show the efficiency of our proposed step optimization algorithm in the following section.

Alg. 4 Step optimization

for (cycles = 0; cycles $<$ effort; cycles++) **do**
  $\Omega.M_{L\rightarrow R}$; $\Omega.D_{L\rightarrow R}$; $\Omega.A$; $\Psi.C$;
  $\Omega.I_{R\rightarrow L}$;
  $\Omega.I_{R\rightarrow L(1-3)}$;
  $\Omega.M_{L\rightarrow R}$; $\Omega.D_{L\rightarrow R}$; $\Omega.A$; $\Psi.C$;
  } push-up
end for

IV. Experimental Results

A. Experimental setup

To have a comprehensive performance assessment and comparison, experiments are carried out over a benchmark set including 25 Boolean functions from ISCAS89 [17] and LGsynth91 [18] with a number of input variables from 7 to 135, and the number of cycles (effort) is set to 40 in all experiments. The run-time of each proposed algorithm for the whole benchmark set is less than 3 seconds.

B. Optimization results

The experimental results of the presented algorithms are shown in Table II. Due to the lack of space, only results of the proposed algorithms are given for both realizations. As expected, the smallest values for the number of RRAMs and computational steps belong to the MAJ-based realization. The step optimization for the MAJ-based realization has resulted in MIGs with the smallest number of steps that is almost one forth of the obtained value by the depth optimization algorithm performed on the IMP-based realization. Even considering the results of the step optimization on the IMP-based realization, this reduction is obvious in comparison with the results of conventional depth optimization. This proves that the employed techniques to reduce the complemented edges have been effective and the proposed step optimization algorithm satisfies requirements of fast RRAM-based circuit implementations.

The proposed algorithm for RRAM costs optimization performed on the MIGs using the IMP-based realization has reduced the sum of the number of steps by 35.39%, i.e., the major drawback of sequential implementation has been effectively lowered. Furthermore, the proposed multi-objective optimization algorithm achieves 30.43% smaller number of steps compared to the conventional depth optimization. The proposed multi-objective algorithm for the MAJ-based realization achieves the smallest number of required RRAMs over other algorithms as well as maintaining a quite small number of computational steps. Sum of the RRAM counts by the proposed multi-objective algorithm is almost 19.78% lower than the same value by the proposed algorithm for step optimization at a cost of 21.09% increase in the sum of the number of steps which confirms the good trad-off and high efficiency in the resulting circuits.

C. Comparison with existing approaches using BDD and AIG

Table III shows the comparison of results of the proposed multi-objective MIG optimization algorithm for RRAM-based logic circuits obtained by both proposed realizations with the results by two previous works using BDD-based [11] and AIG-based [12] synthesis. Both works exploit optimization to lower the number of RRAMs and computational steps. According to Table III, the sum of the number of computational steps by our proposed MIG-based synthesis approach for the MAJ-based realization is almost 8 times smaller than the corresponding value obtained by BDD-based synthesis [11] at a fair cost of 57.42% increase in the total number of RRAMs. This is mostly due to the fact that MIGs have the privilege of benefiting from the built-in majority property of RRAMs. Although the ratio of the number of steps between the BDD-based approach in [11] and the proposed MIG-based approach scales down to 4.5 for the IMP-based realization, it can be still regarded as a noticeable superiority of MIGs in synthesis of RRAM-based circuits. This is especially obvious for larger functions. For example, the numbers of steps for the largest functions in the benchmark set *apex6* and *x3* with 135 inputs obtained by the proposed MIG optimization algorithm are equal to 121 and 99 for the IMP-based realization and 44 and 44 for the MAJ-based realization. While, the same values obtained by BDD-based synthesis exceed 1000 steps . More precisely, the number of required steps obtained by the MAJ-based realization for both functions is 26.5 times smaller than the corresponding result by the BDD-based approach at a low cost increase of 32.2% in the number of RRAMs. In other words, synthesis of RRAM-based circuits with BDDs for large Boolean functions might be too costly or even impractical due to the high number of computational steps, whereas the resulting circuits by MIG-based synthesis still remain efficient and quite fast.

The results of AIG-based synthesis [12] is given for a different set of Boolean functions including smaller circuits with input variables from 3 to 16. Since the number of required RRAMs for the benchmark set are not given in [12], here we can only compare with respect to the number of computational steps. The total number of steps by the proposed MIG optimization algorithm for the MAJ-based and IMP-based realizations are respectively 7.1 and 2.57 times smaller than the same value obtained in [12]. Furthermore, the AIG-based synthesis approach proposed in [12] fails to keep the number of computational steps at a reasonable value when the number of inputs increases. As shown in Table III, the approach proposed in [12] requires 1172 and 1564 computational steps, respectively, for functions *sym10_d* and *t481_d* with 10 and 16 input variables. While using our MIG optimization algorithm, both functions can be synthesized with only 72 or 187 steps for the MAJ-based or IMP-based realizations, respectively.

V. Conclusion

We presented an approach for MIG-based synthesis of Boolean functions implemented with RRAMs using two different realizations. We proposed MIG optimization algorithms to reduce the number of RRAMs and computational steps addressing the area and delay of the resulting circuits, respectively. Experimental results show that the proposed algorithms have successfully fulfilled the aims of optimization that are either finding a trade-off between both objectives or minimizing the number of computational steps. Especially, our proposed approach gains high quality performance with respect to the number of steps that is known to be the major cost metric in RRAM-based circuit design.

References

[1] H.-S. P. Wong, H. Lee, S. Yu, Y. Chen, Y. Wu, P. Chen, B. Lee, F. T. Chen, and M. Tsai, "Metal-oxide RRAM," *Proc. IEEE*, vol. 100, no. 6, pp. 1951–1970, 2012.

TABLE II. THE NUMBER OF REQUIRED RRAMS AND STEPS OBTAINED BY THE OPTIMIZATION ALGORITHMS

Benchmark	Inputs	Area-IMP		Depth-IMP		RRAM costs-IMP		RRAM costs-MAJ		Step-IMP		Step-MAJ	
		R	S	R	S	R	S	R	S	R	S	R	S
5xp1_90	7	170	110	213	110	199	99	149	36	264	77	182	28
alu4_98	14	1542	286	1858	242	2160	176	1370	72	2461	165	1717	56
apex1	45	2647	241	3399	187	3676	165	2343	56	4335	121	2972	44
apex2	39	355	275	583	231	531	143	358	56	653	132	435	47
apex4	9	3854	198	4122	176	4728	143	2820	64	5340	132	3602	48
apex5	117	1240	275	1757	143	1482	141	1053	47	1975	98	1286	35
apex6	135	1097	198	1277	143	1652	121	1018	44	1742	99	1191	36
apex7	49	300	176	389	143	408	132	277	48	526	121	348	44
b9	41	252	99	252	88	252	87	168	32	252	66	168	28
clip_124	9	256	132	276	121	312	110	217	40	380	99	275	36
cm150a	21	132	99	132	99	147	77	95	32	132	88	90	32
cm162a	14	90	99	90	77	90	86	60	30	90	66	65	24
cm163a	16	102	77	102	77	102	76	68	27	102	66	68	24
cordic_138	23	199	164	242	132	189	121	134	48	229	99	162	39
misex1_178	8	101	77	128	66	111	66	76	24	130	55	94	20
misex3_180	14	1547	253	2118	231	2207	165	1444	67	2621	143	1762	52
parity	16	224	176	224	176	216	132	152	53	216	154	152	48
seq_201	41	2032	308	2566	242	3189	153	1970	64	3551	132	2498	60
t481_208	16	102	209	168	132	148	142	90	52	188	110	123	40
table5	17	1598	286	2719	231	2630	154	1723	64	3393	142	2252	52
too_large	38	315	341	512	264	510	164	322	64	587	121	392	48
x1	51	442	164	736	110	569	99	435	36	711	77	509	28
x2	10	66	88	92	77	66	76	46	26	94	66	68	24
x3	135	1075	198	1363	143	1729	99	1008	44	1787	99	1201	36
x4	94	570	121	591	88	599	77	391	28	694	66	563	24
$\sum$	979	20308	4650	25909	3729	27902	3004	**17787**	1154	32453	2594	22175	**953**

TABLE III. COMPARISON OF RESULTS WITH EXISTING APPROACHES FOR SYNTHESIS OF RRAM-BASED CIRCUITS USING BDD AND AIG

Benchmark	Inputs	BDD [11]		MIG-IMP		MIG-MAJ		Benchmark	Inputs	AIG [12]	MIG-IMP		MIG-MAJ	
		R	S	R	S	R	S			S	R	S	R	S
5xp1_90	7	84	73	199	99	149	36	9sym_d	9	1418	923	175	398	60
alu4_98	14	642	334	2160	176	1370	72	con1f1	7	18	70	75	28	26
apex1	45	1626	705	3676	165	2343	56	con2f2	7	19	60	76	24	24
apex2	39	122	237	531	143	358	56	exam1_d	3	12	43	44	19	16
apex4	9	2073	447	4728	143	2820	64	exam3_d	4	12	50	55	20	23
apex5	117	806	888	1482	141	1053	47	max46_d	9	427	408	131	193	48
apex6	135	770	1169	1652	121	1018	44	newill_d	8	50	129	109	57	40
apex7	49	290	437	408	132	277	48	newtag_d	8	21	90	96	36	33
b9	41	125	298	252	87	168	32	rd53f1	5	27	60	64	24	25
clip	9	120	89	312	110	217	40	rd53f2	5	57	77	77	35	28
cm150a	21	56	**127**	147	77	95	32	rd53f3	5	32	86	66	38	24
cm162a	14	46	102	90	86	60	30	rd73f1	7	238	291	121	140	44
cm163a	16	42	116	102	76	68	27	rd73f2	7	46	129	88	57	32
cordic	23	32	149	189	121	134	48	rd73f3	7	104	193	107	84	39
misex1	8	83	69	111	66	76	24	rd84f1	8	351	430	153	187	52
misex3	14	444	185	2207	165	1444	67	rd84f2	8	47	172	88	76	31
parity	16	23	113	216	132	152	53	rd84f3	8	23	90	50	36	15
seq	41	1566	692	3189	153	1970	64	rd84f4	8	345	473	141	214	47
t481	16	26	107	148	142	90	52	sao2f1	10	102	110	108	72	35
table5	17	580	168	2630	154	1723	64	sao2f2	10	112	234	119	98	42
too_large	38	282	232	510	164	322	64	sao2f3	10	380	325	143	143	55
x1	51	230	398	569	99	435	36	sao2f4	10	252	326	143	163	59
x2	10	60	80	66	76	46	26	sym10_d	10	1172	1475	187	643	72
x3	135	770	1169	1729	99	1008	44	t481_d	16	1564	1285	187	567	72
x4	94	401	642	599	77	391	28	xor5_d	5	32	86	66	38	24
$\sum$	979	**11299**	9026	27902	3004	17787	**1154**	$\sum$	194	6861	7615	2669	**3390**	**966**

[2] Y. Ho, Huang, G.M., and P. Li, "Dynamical properties and design analysis for nonvolatile memristor memories," *IEEE Trans. Circuits Syst.*, vol. 58, no. 4, pp. 724–736, 2011.

[3] K.-C. Liu, W.-H. Tzeng, K.-M. Chang, Y.-C. Chan, C.-C. Kuo, and C.-W. Cheng, "The resistive switching characteristics of a Ti/Gd2O3/Pt RRAM device," *Microelectronics Reliability*, vol. 50, no. 5, pp. 670–673, 2010.

[4] Y. Pershin and M. Di Ventra, "Practical approach to programmable analog circuits with memristors," *IEEE Trans. Circuits Syst.*, vol. 57, no. 8, pp. 1857–1864, 2010.

[5] K.-T. T. Cheng and D. B. Strukov, "3D CMOS-memristor hybrid circuits: Devices, integration, architecture, and applications," in *Proc. of the 2012 ACM International Symposium on Physical Design*, 2012, pp. 33–40.

[6] P.-E. Gaillardon, X. Tang, J. Sandrini, M. Thammasack, S. R. Omam, D. Sacchetto, Y. Leblebici, and G. De Micheli, "A ultra-low-power FPGA based on monolithically integrated RRAMs," in *DATE*, 2015, pp. 1203–1208.

[7] D. B. Strukov, "Smart connections," *Nature*, vol. 476, pp. 403–405, 2011.

[8] L. Chua, "Memristor-The missing circuit element," *IEEE Trans. Circuit Theory*, vol. 18, no. 5, pp. 507–519, 1971.

[9] ——, "Resistance switching memories are memristors," *Applied Physics A*, vol. 102, no. 4, pp. 765–783, 2011.

[10] E. Lehtonen, J. Poikonen, and M. Laiho, "Implication logic synthesis methods for memristors," in *ISCAS*, 2012, pp. 2441–2444.

[11] S. Chakraborti, P. Chowdhary, K. Datta, and I. Sengupta, "BDD based synthesis of Boolean functions using memristors," in *IDT*, 2014, pp. 136–141.

[12] J. Bürger, C. Teuscher, and M. Perkowski, "Digital logic synthesis for memristors," in *Reed-Muller 2013*, 2013.

[13] L. Amarù, P.-E. Gaillardon, and G. De Micheli, "Majority-inverter graph: A novel data-structure and algorithms for efficient logic optimization," in *DAC '14*, 2014, pp. 194:1–194:6.

[14] L. Amarù, A. Petkovska, P.-E. Gaillardon, D. N. Bruna, P. Ienne, and G. De Micheli, "Majority-inverter graph for FPGA synthesis," in *SASIMI*, 2015.

[15] P.-E. Gaillardon, L. Amarú, A. Siemon, E. Linn, R. Waser, A. Chattopadhyay, and G. De Micheli, "The programmable logic-in-memory (PLiM) computer," in *DATE*, 2016.

[16] J. Borghetti, G. S. Snider, P. J. Kuekes, J. J. Yang, D. R. Stewart, and R. S. Williams, "Memristive switches enable stateful logic operations via material implication," *Nature*, vol. 464, pp. 873–876, 2010.

[17] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *ISCAS*, 1989, pp. 1929–1934.

[18] S. Yang, "Logic synthesis and optimization benchmarks user guide: Version 3.0." MCNC, 1991.