Inversion Optimization in Majority-Inverter Graphs

Eleonora Testa[1], Mathias Soeken[1], Odysseas Zografos[2], Luca Gaetano Amaru[3], Praveen Raghavan[2], Rudy Lauwereins[2], Pierre-Emmanuel Gaillardon[4], Giovanni De Micheli[1]
[1] EPFL, Switzerland [2] IMEC, Belgium [3] Synopsys, CA, USA [4] University of Utah, UT, USA

***Abstract*—Many emerging nanotechnologies realize majority gates as primitive building blocks and they benefit from a majority-based synthesis. Recently, *Majority-Inverter Graphs* (MIGs) have been introduced to abstract these new technologies. We present optimization techniques for MIGs that aim at rewriting the complemented edges of the graph without changing its shape. We demonstrate the performance of our optimization techniques by considering three cases of emerging technology design: semi-custom digital design using *Spin Wave Devices* (SWDs) and *Quantum-Dot Cellular Automata* (QCA); and logic in-memory operation within *Resistive Random Access Memories* (RRAMs). Our experimental results show that SWD and QCA technologies benefit from complemented edges minimization. Area, delay, and power of SWD-based circuits are improved by 13.8%, 21.1%, and 9.2% respectively, while the number of QCA cells in QCA-based circuits can be decreased by 4.9% on average. Reductions of 14.4% and 12.4% in the number of devices and sequential steps respectively can be achieved for RRAMs when the number of nodes with exactly one complemented input is increased during MIG optimization.**

I. Introduction

Nanotechnologies are being studied as replacement or enhancement for CMOS. Devices in these nanotechnologies have logic models different from standard transistors and many of them realize majority gates as primitive building blocks. Examples of these nanotechnologies are *Quantum-Dot Cellular Automata* (QCA, [1], [2]), *Spin Wave Devices* (SWD, [3], [4]) and *Resistive Random Access Memories* (RRAMs, [5], [6]). To properly assess these post-CMOS technologies, *Electronic Design Automation* (EDA) tools necessitate new logic synthesis techniques and abstractions [7]. Much work concerning majority synthesis has been carried out back in the 1960s [8], [9]. Recently, *Majority-Inverter Graphs* (MIG, [10]) are found to suitably abstract novel majority-based nanotechnologies [11], besides being a useful tool to reduce area and delay in standard CMOS circuits [10]. MIGs use the majority-of-three function $\langle xyz \rangle = xy \vee xz \vee yz$ and negation as only logic primitives; negations are simply represented as complemented edges in the graph. Previous work has considered inversion minimization [12], but they has not taken into account nanotechnologies applications. Minimization of inverters plays a predominant role in emerging technologies whose circuits are built using only majorities (MAJ) and inverters (INV) since area and delay costs depend on the number of INVs in the circuit. In this work, we exploit the intrinsic algebraic properties of MIGs which allow for superior rewriting possibilities as compared to other logic representations such as *And-Inverter Graphs* (AIGs). As an example, negations can be freely propagated through an MIG using self-duality, i.e., $\langle \bar{x}\bar{y}\bar{z} \rangle = \overline{\langle xyz \rangle}$.

In this paper, we present MIG rewriting techniques that target at optimizing inversion within the logic network. The purpose of this work is to implement MIG optimizations by working only on complemented edges. We demonstrate an approach to obtain a minimal number of complemented edges, achieving a reduction of 60.8% in average.

MIGs are an ideal logic representation in the synthesis of technologies that (i) are inherently based on majority and (ii) show different performance based on the influence of inverter metrics. In this paper, we demonstrate our proposed approach on three case studies addressing (i) SWDs and (ii) QCA, in which the number of inverters is minimized, and (iii) RRAMs, which show best performance when each switch has exactly one complemented input.

II. Background

A. Majority-based Technologies

Many emerging nanotechnologies carry natively the MAJ function. Here, we introduce the basics of SWDs, QCA and RRAMs.

1) Spin Wave Devices: Spin-based logic is among the most popular emerging device-circuit architectures [13] thanks to its non-volatility, intrinsic data parallelism, and high endurance. SWDs use spin as information carrier that propagates in waves. The operating principle of these circuits relies on a synthetic multiferroic stack used to generate and detect spin waves, called *Magneto-Electric* (ME) cell. The generated spin waves propagate in ferromagnetic wires, called spin wave buses. The computation principle is based on the interference of propagating spin waves, that can be constructive or destructive. The information is then encoded in the phase of the waves.

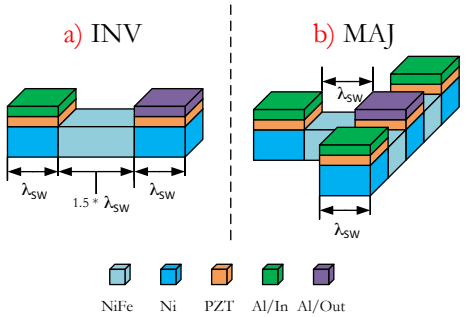

Fig. 1. Layout views of two standard cells used to P&R the SWD circuit [14].

SWD technology provides the capability of implementing simple and compact majority gates and it benefits from majority-based synthesis [14]. A majority (MAJ) and an inverter (INV) standard cell were presented in [15] and are shown in Fig. 1. An INV is simply a waveguide with length 1.5× the spin wavelength (λ_{SW}); the MAJ gates can be produced by symmetrically merging three waveguides.

In our work, we use large benchmarks whose layout cannot be readily achieved. Thus, we estimate area and delay based on the number of cells and the critical path length.

2) Quantum-Dot Cellular Automata: QCA is a nanotechnology first proposed in 1993 [16] that can be used to design circuits with high device density, high switching speed and low power consumption. The QCA technology is based on the interaction of QCA cells. Each cell consists of four quantum dots coupled by tunnel barriers and has two free electrons that are able to tunnel between the dots. The Coulomb repulsion is able to force these electrons in opposite corners of the cell producing two energetically equivalent polarizations, i.e., $P = 1$ and $P = -1$. These two polarizations are used to represent logic `1` and `0` respectively.

Since QCA based circuits can be built using only majority and inverter [1], much work has been made in the past to efficiently build QCA circuits based on majority gates [17], [18]. Figs. 2a and 2b show the layout of a QCA inverter and majority gate respectively.

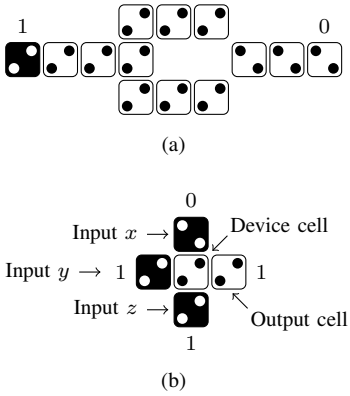

Fig. 2. QCA layout for inverter (a) and majority (b)

Designs and performance analysis of QCA-based circuits can be obtained using QCADesigner [19]. Fig. 3 shows the QCA layout of a full adder. Three majorities and two inverters are needed to generate the sum (S) and carry (C), while much area is used for routing. The number of cells is 166 and the total area is 483 nm^2. The delay is evaluated by counting the number of clock zones of different colors and it is equal to 5.

Since the QCA layouts are not available (as in the case of SWDs), QCADesigner cannot be used to derive such characteristics. Instead, we estimate the circuit area based on the number of QCA cells.

3) Resistive Random Access Memories: RRAMs are one of the most promising emerging nanotechnologies [20] thanks to its non-volatility, high endurance, and high density. These memories are two terminal devices capable to switch from a high resistance state to a low resistance state by applying a bias at its terminals, named P and Q. The internal resistance state Z_n is a function of the previous resistance state Z, the voltage at terminal P and the voltage at terminal Q and it can be expressed by $Z_n = \langle P\overline{Q}Z \rangle$. This basic operation, called *Resistive Majority* (RM_3), embeds both a majority-of-three and an inversion and can be used for synthesis of primary logic gates, enabling in-memory computing.

In principle, RRAMs can realize a MIG network by simply replacing each node by a RRAM cell and use inverters or other RRAM cells for inversion. Nevertheless the connection among cells requires R/W circuitry, making the implementation complicated and non-competitive. A computer architecture called *Programmable Logic-in-Memory* (PLiM) has been proposed [6] that allows logic operations on a regular RRAM array. Also, the idea of using MIGs to derive PLiM programs has recently been described [21], [22]. In PLiM, the MIG network is converted into a sequence of operations to be executed sequentially on some RRAM cells, thus requiring one R/W circuit. This approach is more appealing for exploring RRAMs as a computing technology, even in view of its possible parallelization.

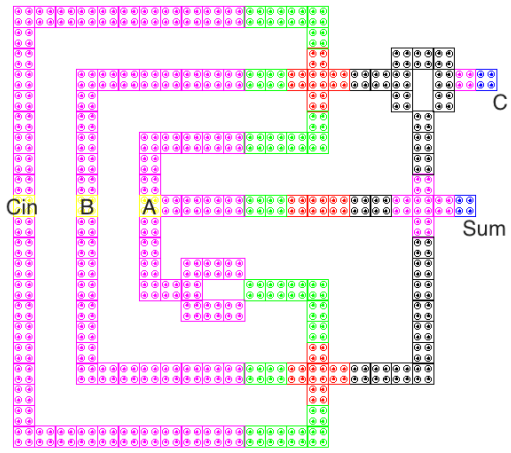

Fig. 3. QCA layout for the full adder

B. Majority-based Logic Manipulation

In this section, we describe MIGs [10], [23], [24], which are logic representation forms based on majority logic. A MIG is a data structure for Boolean function representation and optimization. It is defined as a homogeneous logic network consisting of 3-input majority nodes and regular/complemented edges. MIGs can efficiently represent Boolean functions thanks to the expressive power of the majority operator. Indeed, a majority operator can be configured to behave as a traditional conjunction (AND) or disjunction (OR) operator. In the case of 3-input majority operator, fixing one input to `0` realizes an AND while fixing one input to `1` realizes an OR. As a consequence of the AND/OR inclusion by MAJ, traditional *AND/OR/INV Graphs* (AOIGs) are a special case of MIGs and MIGs can be easily derived from AOIGs. An example of MIG representations derived from its optimal AOIG is depicted by Fig. 4a. Intuitively, MIGs are at least as compact as AOIGs. However, even smaller MIG representation arises when fully exploiting the majority functionality, i.e., with non-constant inputs [10]. In order to manipulate MIGs and reach advantageous MIG representations, a dedicated Boolean

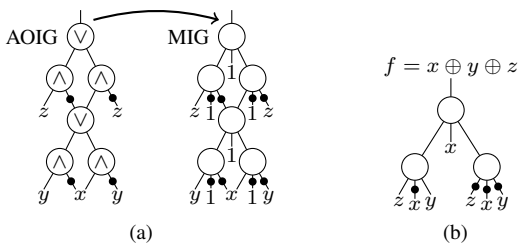

Fig. 4. Example (a) of MIG (right) for $f = x \oplus y \oplus z$ derived by transposing its optimal AOIG (left). Complements are represented by bubbles on the edges. Example (b) of optimized MIG for f.

algebra was introduced in [23]. The axiomatic system for the MIG Boolean algebra, referred to as Ω, is defined by five primitive transformation rules. By just using sequences of transformations in Ω it is possible to transform one MIG into its optimized structure. Fig. 4b shows the new MIG structure for the graph of Fig. 4a.We refer the reader to paper [10] for an in-depth discussion on MIG optimization recipes.

Here, we focus on MIG self-duality, which states that negations can be freely propagated through the graph. This MIG property is described by one of the axiom in Ω [10] referred as Inverter Propagation ($\Omega.I.$) and given by $\overline{\langle xyz\rangle} = \langle \bar{x}\bar{y}\bar{z}\rangle$.

III. Inversion Optimization

This section describes the techniques employed to optimize the number of complemented edges, i.e., inversions. First, we explain how to minimize the number of complemented edges in a MIG. Then, we present rules to maximize the number of nodes having one complemented input, since this represents a benefit for RRAMs-based circuits.

A. Minimization of Complemented Edges

Complemented edges minimization can be highly effective for improving performances of emerging technologies which use MAJ and INV as standard cells. We will focus on effects of inversion minimization for SWDs and QCA in the following section.

Minimization is obtained by recursively applying the inverter propagation axiom $\Omega.I$ to move complemented edges on the inputs to the output. To reduce the number of complemented edges, we apply the transformations rules mentioned below on all the nodes of the MIG. These transformations do not change the depth nor the size of the MIG. Different ways of applying $\Omega.I$ can be adopted depending on whether the node has constant inputs or not. Furthermore, the axiom can be applied considering one node at a time (one-level) or evaluating savings in the number of complemented edges for two levels of the MIG (two-level). A taxonomy of the rules used to decrease complemented edges is proposed in the following; then the procedure used to minimize complemented edges is described.

1) One-level Rules for MAJ: Here, we describe rules that apply to *MAJ* nodes, which are nodes with no constant inputs. The rules used to decrease complemented edges aim at moving complemented edges of the inputs to the output. They can be formalized as:

$$\text{Rules}\begin{cases} \textbf{MAJ\_3:} \begin{cases} \langle \bar{x}\bar{y}\bar{z}\rangle = \overline{\langle xyz\rangle} \\ \overline{\langle \bar{x}\bar{y}\bar{z}\rangle} = \langle xyz\rangle \end{cases} \\ \textbf{MAJ\_2:}\ \overline{\langle x\bar{y}\bar{z}\rangle} = \langle \bar{x}yz\rangle \end{cases} \tag{1}$$

The MAJ_3 rules consider nodes in which the three inputs are complemented. Considering each node, these transformations lead to a decrease in the number of complemented edges equal to $3 + (\#\mathrm{CO} - \#\mathrm{NCO})$ where $\#\mathrm{CO}$ is the number of complemented outputs of the node and $\#\mathrm{NCO}$ are the uncomplemented outputs. Savings for the MAJ_2 rule are equal to: $1 + (\#\mathrm{CO} - \#\mathrm{NCO})$. For these rules, only one node is considered at a time. We evaluate savings according to the formulas mentioned above; each node is changed using one of the transformation rules if savings larger than 0 can be achieved. An example is given in Fig. 5. Fig. 5a shows the MIG of the full adder composed by the three nodes 1, 2, and 3. It is possible to apply the MAJ_3 rule on node 2 with savings of 3 and the MAJ_2 on node 1 with savings of 2. Fig. 5b illustrates the MIG of the full adder after the one-level transformations applied on the nodes of the first level.

(a) (b)

Fig. 5. MIG for a full adder, before (a) and after (b) one-level rules to decrease the number of complemented edges. Bubbles represent complementation of the edges

2) One-level Rules for AND/OR: All the rules mentioned above are applied to nodes in which none of the inputs is set to a constant value, but they equally work for *AND* and *OR* nodes, which are majority nodes in which one of the input is set to `0` and `1`, respectively. Our MIG data structure only has constant `1`, hence in AND nodes the constant input is $\bar{1} = 0$. The additional rules used to decrease complemented edges for AND/OR are:

$$\text{AND/OR Rules}\begin{cases} \textbf{AND\_3:} \begin{cases} \langle \bar{x}\bar{y}\bar{1}\rangle = \overline{\langle xy1\rangle} \\ \overline{\langle \bar{x}\bar{y}\bar{1}\rangle} = \langle xy1\rangle \end{cases} \\ \textbf{AND\_2:}\ \overline{\langle x\bar{y}\bar{1}\rangle} = \langle \bar{x}y1\rangle \\ \textbf{OR\_2:}\ \overline{\langle \bar{x}\bar{y}1\rangle} = \langle xy\bar{1}\rangle \end{cases} \tag{2}$$

Note that complements on constant inputs are not accounted in the total amount of complemented edges as in physical implementations both constants `0` and `1` are present. The savings estimation is adjusted for the AND/OR cases. The AND_3 rule has savings equal to $2+(\#\mathrm{CO}-\#\mathrm{NCO})$, while for the AND_2 rule they are equal to: $\#\mathrm{CO} - \#\mathrm{NCO}$. For OR rule savings are: $2 + (\#\mathrm{CO} - \#\mathrm{NCO})$.

3) Two-level Rules: In two level transformations we consider one node (main node) and all the nodes on its outputs (parents). By doing this, we consider that transforming only the main node may not result in savings greater than 0, but this transformation changes the complemented edges pattern of the parents and consequently it may result in a total two-level savings larger than 0 when applying transformations on the parents. We evaluate the total two-level savings as the sum of the savings obtained by changing the main node and the savings achieved on the parents if the main node is changed. If the total two-level savings are positive, first the main node is changed according to the rules 1, then all the one-level rules are applied to the parents. In Fig. 6a, none of the nodes, which are called here node 1, 2 and 3, can be changed by the one-level transformation since there is no benefit in the total number of complemented edges. On the other hand, changing node 1 according to the inverter propagation axiom generates the possibility of applying one of the one level rules on node 3. If node 1 is changed, its savings are equal to 0, but, thanks to this first transformation, the MAJ_2 rule can be applied on node 3. Fig. 6b shows the full adder after changing node 1; while Fig. 6c shows the final circuit. In this case, the number of complemented edges is not reduced from step a to step b but it is reduced from step b to c, with total savings equal to 2.

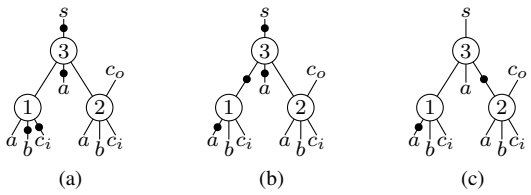

Fig. 6. Two-level transformation applied on the MIG of a full adder.

4) Minimization Procedure: We minimize the complemented edges using the reduction rules proposed above in the way given in Alg. 1. We apply each single rule from node 0 to the last node of the circuit until no further rules can be applied and we have reached the minimal number of complemented edges achievable with our procedure. At the end of the minimization process, the MIG has the same logic depth and the same size of the unoptimized one.

```
1 while the number of compl. edges is decreasing do
2 |   AND_3(n) for each node n;
3 |   AND_2(n) for each node n;
4 |   MAJ_3(n) for each node n;
5 |   MAJ_2(n) for each node n;
6 |   Two-level(n) for each node n;
7 end
```

Algorithm 1: Complemented edges minimization in MIGs.

B. Constrained Inversion Optimization

Previous sections describe how it is possible to minimize the number of complemented edges of the MIG by using iteratively reduction rules. It is worth noticing that we can take advantage of the proposed transformations to obtain a desired complemented edges configuration, which depends on the specific application. A desired number of complemented edges per node is obtained by applying the same rules used for minimization by adding some constraints. For example we can take advantage of some of the inverter rules to manipulate the distribution of complemented edges in such a way each node has one input inverted. The presence of a single complemented edge on each MIG node represents a benefit for RRAM-based circuits, and can be effective for optimizing both the number of instructions and required RRAMs.

In this case, the MAJ_3 and MAJ_2 rules proposed in Section III-A are used in order to get the highest number of nodes with one complemented input. These rules are not simply applied iteratively on each node, but the effects that these transformations have on the node outputs are considered. As an example, Fig. 7a shows a MIG with 11 complemented edges optimized for RRAMs; ideal nodes are those with one complemented edge on one input, which are nodes 4, 6, 7, 8, and 9 in Fig. 7a. In this case the number of ideal nodes is 5. MAJ_2 rule can be applied to nodes 1 and 3, and consequently the MAJ_3 rule can be applied on node 5. Fig. 7b shows the MIG with a minimal number of complemented edges. This is not the best configuration in terms of number of node with one complemented input, since in this second case the ideal nodes are only 2. Changing nodes 1 and 3 result in advantage equal to 0, since it will create two ideal nodes but it will eliminate previously ideal nodes 4 and 6. Moreover, changing node 5 will further eliminate ideal nodes 7, 8, and 9.

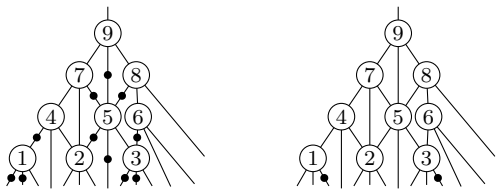

Fig. 7. Example showing how minimization is not good to obtain one complemented input on each node

IV. Experimental Results

In this section, first, we describe the improvement obtained in the number of complemented edges with the minimization rules proposed in Section III-A. Then, we test the optimization techniques discussed above for the three different case study technologies.

A. Logic Optimization

We developed a C program to minimize the number of complemented edges in the MIG according to Alg. 1. We evaluate our approach on the already optimized arithmetic circuits of the EPFL benchmarks[1] using the rewrite recipe described in [10]. We compare the number of complemented edges; results are shown in Table I.

The largest improvement in the number of complemented edges is obtained after the first loop (column 1 in Table I). Our results show that by applying once one-level and two-level transformations is possible to decrease the number of complemented edges by 59.2% as average. On average, 4 loops are necessary to reach the minimal number of complemented edges. The final improvement is of 60.8% in average. We apply formal verification to check the correct behavior of the circuits.

As mentioned before, other optimizations of the complemented edges are possible. We developed a procedure to obtain the largest number of nodes with one complemented input. Results showing the number of complemented edges are shown in the last column of Table I. As expected, the final number of complemented edges is smaller than the original one, but larger than the minimal number of complemented edges.

B. First Case Study: SWDs

In this section, we show how minimizing complemented edges can improve performances of SWD-based circuits. To evaluate performances of SWD circuits, we use specifications of SWD technology as modeled in [13], [15]. Area of SWD circuits is evaluated as the cumulative area of majority and inverter standard cells; the number of MAJ cells is equal to the size of the MIG, while the number of INV cells has been evaluated as the number of MIG nodes with at least one complemented fanout. For the total delay, we consider the path with the maximum number of majority nodes and complemented edges.

To test our minimization techniques, we compute area, delay, and energy of the SWD-based circuits, for both the non-optimized and optimized MIG.

The improvements for the optimized circuit are shown in Table II. Due to the smaller amount of complemented edges, the SWD circuit area of the optimized MIG is lower for all benchmarks considered, even if the number of majority nodes

[1] http://lsi.epfl.ch/benchmarks

TABLE I. COMPLEMENTED EDGES OPTIMIZATION

Benchmark	$\#N$	#CE	$\#\text{CE}_{\min}$						Improvement						$\#\text{CE}_{1\text{inp}}$
			1	2	3	4	5	6	1	2	3	4	5	6	
adder	2978	2905	1431	1278	1245	1232	1230	**1228**	50.7%	56.0%	57.1%	57.6%	57.7%	**57.7%**	2620
log2	37582	38585	14744	14380	14289	**14287**			61.8%	62.7%	63.0%	**63.0%**			28259
max	7202	6543	2977	2967	**2965**				54.5%	54.7%	**54.7%**				4348
multiplier	41885	39589	19089	18953	**18941**				51.8%	52.1%	**52.2%**				33606
sin	7890	7625	3183	3126	**3124**				58.3%	59.0%	**59.0%**				6305
sqrt	52344	53327	20895	20232	20131	20105	**20101**		60.8%	62.1%	62.2%	62.3%	**62.3%**		44213
square	19200	23390	5519	5461	5456	**5455**			76.4%	76.7%	76.7%	**76.7%**			13650
Average									59.2%	60.5%	60.7%	60.8%	60.8%	60.8%	

$\#N$: number of MIG nodes, #CE: number of complemented edges, $\#\text{CE}_{\min}$: number of complemented edges after minimization, Improvement: improvement with regards to the initial values, $\#\text{CE}_{1\text{inp}}$: number of complemented edges after constrained optimization. For each circuit, the minimization is stopped when the minimal number of CE is reached.

TABLE II. SWD EXPERIMENTAL RESULTS

Benchmark	$\#N$	MIG		Optimized MIG		Improvements			
		#INV	Depth	#INV	Depth	Area	Delay	Energy	ADEP
adder	2978	1449	21	830	19	9.1%	9.5%	5.9%	22.6%
log2	37582	22481	342	8445	260	15.6%	24.0%	10.4%	42.5%
max	7202	3147	52	1107	37	12.7%	28.8%	8.2%	43.0%
multiplier	41885	25594	205	12948	163	12.5%	20.5%	8.3%	36.2%
sin	7890	3823	164	1932	128	10.5%	22.0%	6.9%	34.9%
sqrt	52344	28734	1249	12840	980	12.9%	21.5%	8.5%	37.5%
square	19200	17158	70	5012	55	23.5%	21.4%	16.2%	49.6%
Average						13.8%	21.1%	9.2%	38.1%

$\#N$: number of MIG nodes, #INV: number of inverters of the MIG.

is unchanged. The improvement in terms of area is 13.8% as average. There is a gain for delay (21.1%) and for energy (9.2%) as well. The *Area-Delay-Energy-Product* (ADEP) is totally improved by 38.1% as average.

C. Second Case Study: QCA

In this section, we show that improvement in the number of QCA cells can be obtained by minimizing the number of complemented edges. To evaluate the number of QCA cells, we consider both majority, inverter, and wire routing. We consider that 5 cells are necessary for each majority cells, while 9 cells are used to build each wire. Thirteen QCA cells are necessary to create the inverter shown in Fig. 2b. In other words, four more cells are necessary on each wire to invert the signal.

To test our optimization, we evaluate the number of QCA cells both for unoptimized circuits and for circuits with a minimal number of complemented edges. The improvement in the number of QCA cells is shown in Table III and it is 4.9% on average. As expected, since we consider wire routing, the improvement is not as large as the one obtained for the logic optimization.

We attempted also a comparison with the results presented in [2] on the 10 most complex benchmarks. Since the synthesis procedures were different (SIS vs MIGhty [23]), the starting point of our inverter optimization is different, thus invalidating a direct comparison. Nevertheless, using MIG, we could achieve an improvement in the number of levels of 38.5% in average.

D. Third Case Study: RRAMs

In this section, we show that increasing the number of nodes with one complemented input can improve performances of RRAM-based circuits. The size of the MIG and the number of complemented edges have an impact on the quality of the PLiM architecture [6] optimization with respect to the number of required RRAMs and number of steps. The PLiM architecture is able to manage only serial operations and only one node can be computed at a time; we compute the total number of instructions as the sum of instructions necessary for each single node. Here, we maximize the number of nodes with one complemented input using rules proposed in Section III-A. We use the same approach presented in [22] to obtain PLiM optimal results, but we do not change the number of nodes of the MIG. We compare performances for RRAM-based circuit for the unoptimized case, the PLiM optimal and the minimized case proposed in Section IV-A. Results are shown in Table IV. Our experimental results shows that, by forcing one complemented input on the largest amount of nodes, the number of required RRAMs is reduced almost twice as compared to when minimizing the number of complemented edges. However, the number of computational steps sees better improvement when minimizing the number of complemented edges compared to the tailored optimization strategy. This is not surprising as the number of RRAMs and the number of steps are contradictory objectives and multi-objective techniques would be required in order to optimize both of them.

V. CONCLUSION

We described rules to optimize the complemented edges distribution according to a specific application. We minimized the number of complemented edges and we showed how minimization affects SWDs and QCA circuits performances. We also illustrated rules to obtain one complemented input on the largest amount of nodes. This optimization technique aims at obtaining better results in terms of performances and number of steps for RRAM based circuits.

At the logic level, our results showed that a decrease in the number of complemented edges up to 60.8% can be achieved. Experiments showed that optimized SWD circuits are improved in terms of area, delay, and power by complemented edges minimization. The ADEP can be improved of 38.1%. QCA-based circuits benefit from inversion minimization and a reduction of 4.9% can be achieved in the number of QCA cells. Regarding RRAMs, within a PLiM architecture the number of

TABLE III. QCA EXPERIMENTAL RESULTS

Benchmark	$\#N$	$\#L$	$\#E$	MIG		Minimal		Impr.
				#INV	#QCA cells	#INV	#QCA cells	
adder	2978	12	7452	1449	87754	830	85278	2.8%
log2	37582	181	78917	22481	988087	8445	931943	5.7%
max	7202	27	16186	3147	194272	1107	186112	4.2%
multiplier	41885	111	92904	25594	1147937	12948	1097353	4.4%
sin	7890	91	17275	3823	210217	1932	202653	3.6%
sqrt	52344	690	111816	28734	1383000	12840	1319424	4.6%
square	19200	36	39836	17158	523156	5012	474572	9.3%
Average								4.9%

$\#N$: number of MIG nodes, $\#L$: number of levels in the MIG not considering inverters, $\#E$: number of edges, $\#INV$: number of inverters,#QCA cells: number of QCA cells, Impr: improvement of the minimal values with respect to the initial MIG.

TABLE IV. RRAM EXPERIMENTAL RESULTS

Benchmark	$\#N$	MIG			PLiM Optimal					Minimal				
		#CE	$\#R$	$\#I$	#CE	$\#R$	Impr.	$\#I$	Impr.	#CE	$\#R$	Impr.	$\#I$	Impr.
adder	2978	2905	263	6583	2620	261	0.8%	6586	-0.1%	1228	261	0.8%	6238	5.2%
log2	37582	38585	1851	85000	28259	1748	5.6%	73891	13.1%	14287	1774	4.2%	65446	23.0%
max	7202	6543	567	13462	4348	562	0.9%	11385	15.4%	2965	565	0.4%	11110	17.5%
multiplier	41885	39589	613	100202	33606	551	10.1%	83078	17.1%	18941	537	12.4%	72646	27.5%
sin	7890	7625	671	16899	6305	571	14.9%	15807	6.5%	3124	577	14.0%	13826	18.2%
sqrt	52344	53327	501	111321	44213	507	-1.2%	101748	8.6%	20101	445	11.2%	87967	21.0%
square	19200	23390	941	52494	13650	287	69.5%	38776	26.1%	5455	845	10.2%	35737	31.9%
Average							14.4%		12.4%			7.6%		20.6%

$\#N$: number of MIG nodes, #CE: number of complemented edges of the MIG, $\#R$: number of required RRAMs, $\#I$: number of instructions.

instructions can be decreased by up to 14.4% and the number of required RRAMs devices can be reduced by up to 12.4%.

We conclude that new logic synthesis techniques and optimizations are essential to validate emerging nanotechnologies which use logic models different from standard transistors.

ACKNOWLEDGMENT

This research was supported by H2020-ERC-2014-ADG 669354 CyberCare and by the Swiss National Science Foundation, project number 200021 146600.

REFERENCES

[1] I. Amlani, "Digital logic gate using quantum-dot cellular automata," *Science*, vol. 284, no. 5412, pp. 289–291, 1999.

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

[3] T. Schneider, A. A. Serga, B. Leven, B. Hillebrands, R. L. Stamps, and M. P. Kostylev, "Realization of spin-wave logic gates," *Applied Physics Letters*, vol. 92, no. 2, pp. 1–4, 2008.

[4] A. Khitun and K. L. Wang, "Nano scale computational architectures with spin wave bus," *Superlattices and Microstructures*, vol. 38, no. 3, pp. 184–200, 2005.

[5] E. Linn, R. Rosezin, C. Kügeler, and R. Waser, "Complementary resistive switches for passive nanocrossbar memories," *Nature materials*, vol. 9, no. 5, pp. 403–406, 2010.

[6] 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 *Design Automation and Test in Europe*, 2016.

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

[8] S. B. Akers, "Synthesis of combinational logic using three-input majority gates," *Annual Symposium on Switching Circuit Theory and Logical Design*, pp. 149–158, 1962.

[9] R. Lindaman, "A theorem for deriving majority-logic networks within an augmented Boolean algebra," *IRE Trans. On Electronic Computers*, vol. 47, pp. 338–342, 1960.

[10] L. Amarú, P.-E. Gaillardon, and G. De Micheli, "Majority-inverter graph: A new paradigm for logic optimization," *IEEE T-CAD*, 2015.

[11] ——, "Majority-based synthesis for nanotechnologies," *ASPDAC*, pp. 499–502, 2016.

[12] S. Muroga, *Threshold logic and its applications*. NY, New York: John Wiley & Sons Inc., 1971.

[13] O. Zografos, B. Sorée, A. Vaysset, S. Cosemans, L. Amarù, P.-E. Gaillardon, G. De Micheli, R. Lauwereins, S. Sayan, P. Raghavan, I. P. Radu, and A. Thean, "Design and benchmarking of hybrid CMOS-spin wave device circuits compared to 10nm CMOS," in *NANO*, 2015.

[14] O. Zografos, L. Amarú, P.-E. Gaillardon, P. Raghavan, and G. De Micheli, "Majority logic synthesis for spin wave technology," in *Euromicro Conference Digital Systems and Design*, 2014.

[15] O. Zografos, P. Raghavan, Y. Sherazi, A. Vaysset, F. Ciubatoru, B. Soree, R. Lauwereins, I. Radu, and A. Thean, "Area and routing efficiency of SWD circuits compared to advanced CMOS," *International Conference on IC Design & Technology*, pp. 1–4, 2015.

[16] C. S. Lent, D. P. Tougaw, W. Porod, and G. H. Bernstein, "Quantum cellular automata," in *IEEE Trans. On Nanotechnology*, vol. 4, 1993, pp. 49–57.

[17] R. Zhang, K. Walus, W. Wang, and G. Jullien, "A method of majority logic reduction for quantum cellular automata," in *IEEE Trans. On Nanotechnology*, vol. 3, 2004, pp. 443–450.

[18] R. Zhang, P. Gupta, and N. K. Jha, "Synthesis of majority and minority networks and its applications to qca, tpl ans set based nanotechnologies," in *Internation Conference on VLSI Design*, 2005, pp. 229–234.

[19] QCA Designer. Available: http://www.qcadesigner.ca.

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

[21] S. Shirinzadeh, M. Soeken, P.-E. Gaillardon, and R. Drechsler, "Fast logic synthesis for RRAM-based in-memory computing using majority-inverter graphs," *Design Automation and Test in Europe*, 2016.

[22] M. Soeken, S. Shirinzadeh, P.-E. Gaillardon, L. Amarú, R. Drechsler, and G. De Micheli, "An MIG-based compiler for programmable logic-in-memory architectures," *Design Automation Conference*, 2016.

[23] L. Amarú, P.-E. Gaillardon, and G. De Micheli, "Majority-inverter graph: A novel data-structure and algorithms for efficient logic optimization," in *Design Automation Conference*, 2014.

[24] ——, "Boolean logic optimization in majority-inverter graphs," *Design Automation Conference*, 2015.