Majority-basedSynthesisforNanotechnologies

LucaAmarú,Pierre-EmmanuelGaillardon,GiovanniDeMicheli
IntegratedSystemsLaboratory(LSI),EPFL,Switzerland

Abstract—We study the logic synthesis of emergingnanotechnologieswhoseelementarydevicesabstractionisasurajityvoter.
Wearguethat synthesis tools, natively supporting the maj or
logic abstraction, are the technology enablers. This is because
theyallow designers to validate majority-based nanotechnolo-gieson large-scale benchmarks. We describe models and data-
structures for logic design with majority-based nanotechnologies
and we show results of applying new synthesis algorithms and
tools. We conclude that new logic synthesis methods are required
toachieveafairassessmentonemergingnanotechnologies.

I. INTRODUCTION

The arrival of post-CMOS nanotechnologies has brought
new devices whose logic models are different from tradi-
tionaltransistors. In this scenario, new logic abstractions
and synthesis techniques are key to exploit at best the enhanced
functionality of emerging nanodevices.

In this paper, we focus on the synthesis of new digital
technologies whose elementary device abstraction is a majority
voter. Specific examples for these nanotechnologies include,
but are not limited to, silicon nanowire [1], [2], graphene [3],
[4], resistiverandom-accessmemory [5], [6], spin-wave de-

gines [7], [8], quantum-dot cellular automata [9], nanomagnets
[10], DNA-logic [11], and many others [12]–[14]. We revisit
logic synthesis in light of its enabling role in the selection of
majority-based nanotechnologies. We first survey a selected
pool of nanodevices behaving as majority voters. Next, we fo-
cus on logic synthesis, where we present novel data structures
and optimization techniques based on majority logic. Experi-
mental results, over two representative nanotechnologies, show
that majority logic synthesis not only generates better circuits
than standard synthesis tools but also enables a larger gain
versus CMOS. This serves as an empirical demonstration of
potentially advantageous nanotechnologies that cannot emerge
without the leading support of logic synthesis.

The remainder of this paper is organized as follows. Section
II surveys some promising post-CMOS devices behaving as
majorityvoters. Section III describes synthesis models for
majority logic natively fitting the functionality of the devices
presentedin Section II. Section IV shows experimental results
for some of the nanotechnologies introduced in Section II
using the logic synthesis techniques presented in Section III.
Section V is a conclusion.

II. MAJORITY-BASED NANODEVICES

In this section, we review some promising nanodevices
in post-CMOS technologies. We focus on resistive RAM
and spin-wave nanotechnologies. Regardless of using charge-

based or non-charge-based physical phenomena to carry digital
information, all considered devices inherently implement a
majorityvoter. Themajority functionality is a keyasset for
these nanodevices in addition to their intrinsic performances.

A. Spin-Wave Devices

Spin Wave Devices (SWDs) are promising beyond-CMOS
candidates. Unlike charge-based technologies, SWDs use spin
as information carrier that propagates in waves [7], [8]. This
physical mechanism enables ultra-low power operation, two
orders of magnitude lower than current CMOS [7], [8].

Besides the extremely low-power consumption of SWD
logic, which is a key technological asset, the use of wave
computation in digital circuits can enhance its logic expressive
power. SWD logic computation is based on the interference of
spin waves. Depending on the phase of the propagating spin
waves/signals, their interference is constructive or destructive.
The final interference result is translated to the output via
Magnetoelectric (ME) cells. In this scenario, an inverter is
simply a waveguide with length equal to $1.5 \times \lambda_{SW}$. In this way, the information encoded in
the phase of the SW signal arrives inverted to the output ME

cell, Fig. 1(a). The actual logic primitive in SWD technology
is the majority voter, which is implemented by the symmetric
merging of three waveguides Fig. 1(b). Here, the length of

Fig. 1: Primitive gate areas and designs for SWD technology.
All distances are parameterized with the spin wave wavelength
$\lambda_{SW}$ [16].

each waveguide is $1.0 \times \lambda_{SW}$. In the majority
voter structure, the spin wave signal at the output is determined
by the majority phase of the input spin waves.

B. Resistive RAM

The Resistive RAM (RRAM) technology is receiving
widespread research attention as candidate for high-density
and low-cost storage. RRAM store information as an internal
resistive state, which can be either a Low Resistance State
(LRS) or a High Resistance State (HRS) [5], [6]. Their internal
resistance state of the device, $Z_n$, can be modified by applying
a positive or a negative voltage. This is a function of the
previous resistance state $Z$, the voltage at the terminal $P$ and
the voltage at the terminal $Q$. The functionality of RRAM can
be summarized by a state machine [5], [6], as shown in Fig. 2.
Further details can be found in [6]. Transition occurs only for
the conditions \( P = 0, Q = 1 \), i.e., \( V_{PQ} < 0 \) so \( Z \to 0 \) and \( P = 1, Q = 0 \), i.e., \( V_{PQ} > 0 \) so \( Z \to 1 \). Thus, it is possible to express \( Z_n \) as the following:
\[
Z_n = (P \overline{Q})Z + (P + Q)\overline{Z}
\]
\[
= PZ + \overline{Q}Z + PQ\overline{Z}
\]
\[
= PZ + \overline{Q}Z + PQ
\]
\[
= M_3(P, Q, Z)
\]
where \( M_3 \) is the majority Boolean function with 3 inputs.

C. Logic Abstraction and Model for Post-CMOS Devices

The aforementioned nanodevices inherently realize majority gates as primitive building block for computation. Note that majority voting is the basic computational brick also for nanotechnologies other than the ones surveyed in this paper. For example, DNA-logic [11], graphene reconfigurable gates [17], quantum-dot cellular automata [9], etc., are promising majority-based nanotechnologies.

Fig. 2: Resistive majority operation with RRAM devices [15].

Fig. 3: Common logic abstraction for SWD, RRAM, Graphene reconfigurable gates, QCA and DNA logic.

The far reaching consequence of embedding majority operators as atomic functions in nanotechnologies is the ability to implement arithmetic logic with few physical resources.

III. MAJORITY LOGIC SYNTHESIS

In this section, we review data structures and optimization algorithms based on majority logic. We describe Majority-Inverter-Graph (MIG) [18]–[20] as a logic representation form and we define optimization routines running on it.

A. Majority-Inverter Graph

A MIG is a data structure for Boolean function representation and optimization. An MIG is defined as a logic network that consists of 3-input majority nodes and regular/complemented edges [18]–[20]. This model can be extended to wider majority operators but we consider here only 3-input majority nodes for the sake of simplicity.

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. Two examples of MIG representations derived from their optimal AOIGs are depicted by Fig. 4. AND/OR operators are replaced node-wise by MAJ-3 operators with a constant input.

Fig. 4: Examples of MIG representations (right) for (a) \( f = x \oplus y \oplus z \) and (b) \( g = x(y + uv) \) derived by transposing their optimal AOIG representations (left). Complement attributes are represented by bubbles on the edges.

Intuitively, MIGs are at least as compact as AOIGs. However, even smaller MIG representation arise when fully exploiting the majority functionality, i.e., with non-constant inputs [20]. Later, we will show two smaller and lower depth MIGs for the two examples in Fig. 4.

B. Majority-Inverter Graph Manipulation and Optimization

We are interested in compact MIG representations because they translate in smaller and faster physical implementations. In order to manipulate MIGs and reach advantageous MIG representations, a dedicated Boolean algebra was introduced in [18]. The axiomatic system for the MIG Boolean algebra, referred to as \( \Omega \), is defined by the five following primitive transformation rules.

\[
\begin{align*}
\text{Commutativity} & \quad \Omega.C \\
M(x, y, z) & = M(y, x, z) = M(z, y, x) \\
\text{Majority} & \quad \Omega.M \\
\text{if}(x = y) & : M(x, y, z) = x = y \\
\text{if}(x = y') & : M(x, y, z) = z \\
\text{Associativity} & \quad \Omega.A \\
M(x, u, M(y, u, z)) & = M(z, u, M(y, u, x)) \\
\text{Distributivity} & \quad \Omega.D \\
M(x, y, M(y, u, z)) & = M(M(x, y, u), M(x, y, v), z) \\
\text{Inverter Propagation} & \quad \Omega.I \\
M'(x, y, z) & = M(x', y', z')
\end{align*}
\]
Some of these axioms are inspired by median algebra and others by the properties of the median operator in a distributive lattice. A strong property of MIGs and their algebraic framework is about reachability. It has been proven that, by using a sequence of transformations drawn from $\Omega$, it is possible to traverse the entire MIG representation space [20]. In other words, given any two equivalent MIG representations, it is possible to transform one into the other by just using axioms in $\Omega$. This results is of paramount interest to logic synthesis because it guarantees that the best MIG, for a given target metric, can always be reached. Unfortunately, deriving a sequence of $\Omega$ transformations is an intractable problem. As for traditional logic optimization, heuristic techniques provide here fast solutions with reasonable quality [22]. An efficient depth optimization heuristic for MIGs consists of local $\Omega$ rules iterated over the critical path [18]. Here, the rationale driving local transformations is to push up variables with largest depth.

As previously anticipated, by using the MIG algebraic framework it is possible to obtain better MIGs for the examples in Fig. 4. Fig. 5 shows the new MIG structures, which are optimized in both depth (number of levels) and size (number of nodes). These MIGs can be reached using a sequence of $\Omega$ axioms starting from their unoptimized structures. We refer the reader to paper [20] for an in-depth discussion on MIG optimization recipes.

IV. MAJORITY-BASED DESIGN FOR NANOTECHNOLOGIES

In this section, we experiment the efficacy of MIGs in the synthesis of emerging nanodevices. We target two different emerging technologies taken from the devices surveyed in Section II. Note that many other nanodevices may benefit from the presented majority synthesis methodologies [12]–[14].

A. SWD-MIG

In SWDs, the logic primitive is a majority voter. In this scenario, we use the MIG data structure to represent and synthesize SWD circuits. The intrinsic correspondence between MIG elements and SWDs makes MIG optimization naturally extendable to obtain minimum cost SWD implementations. For this purpose, ad hoc cost functions are assigned to MIG elements during optimization as per Table I. These cost functions are derived from the SWD technology implementation of majority and inverter gates [16].

For the sake of clarity, we comment on our proposed MIG-based SWD synthesis flow by means of an example. The logic function in this example is $g = x \cdot (y + u \cdot v)$. This function is initially represented by the MIG in Fig. 4(b), which has a SWD delay cost of 4 and an SWD area cost of 14. By using $\Omega$ transformations, it is possible to reach the optimized MIG depicted by Fig. 5(b). Such an optimized MIG counts the same number of nodes and complemented edges of the original one but one fewer level of depth. In this way, the associated area cost remains 14 but the delay is reduced to 3. After the optimization, each MIG element is mapped onto its corresponding SWD gate. Fig. 6 depicts the SWD mapping for the original (a) and optimized (b) MIGs.

![Fig. 6: SWD circuits implementing function $g = x \cdot (y + u \cdot v)$.]

(a) Standard circuit (b) MIG optimized circuit. ME stands for Magneto-Electric cell and SW for Spin Wave waveguide.

The circuit in Fig. 6(b) features roughly the same area occupation as the one in Fig. 6(a) but shorter input-output path. As per the cost functions in Table I, the achieved speed-up is roughly 25%. Including the physical models in [16], the refined speed-up becomes 18.2%.

We validate hereafter the efficiency of our proposed MIG-based SWD synthesis flow for larger circuits [16]. We also provide a comparison reference to 10-nm CMOS technology.

In MIG-based SWD synthesis, we employed the MIGhty MIG optimizer [18]. As traditional-synthesis counterpart, we employed ABC tool [21] with optimization commands resyn2 and produced in output an AND-Inverter Graph (AIG). The AIGs mapping procedure onto SWDs is in common with MIGs: AND nodes are simply mapped to MAJ gates with one input biased to logic 0. For advanced CMOS, we used a commercial synthesis tool fed with a standard-cell library produced by a 10-nm CMOS process flow. The circuit benchmarks are taken from the MCNC suite.

| TABLE I: Cost Functions for MIGs Mapped onto SWDs |
| MIG Element | SWD Gate | Area Cost | Delay Cost |
| Majority node | Majority Gate | 4 | 1 |
| Complemented edge | Inverter Gate | 1 | 1 |

Table II summarizes the performance results of SWD and CMOS Technologies.

<table>
<thead>
<tr>
<th>Technology</th>
<th>ADP Product (a.u.)</th>
<th>Gain vs CMOS</th>
<th>Gain vs AIG</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMOS</td>
<td>0.077.06</td>
<td>18.45 x</td>
<td>1.22 x</td>
</tr>
<tr>
<td>SWD - AIG</td>
<td>526.25</td>
<td>18.45 x</td>
<td>-</td>
</tr>
<tr>
<td>SWD - MIG</td>
<td>432.59</td>
<td>22.44 x</td>
<td>1.22 x</td>
</tr>
</tbody>
</table>

Table II summarizes the performance of the benchmarks in the Area-Delay-Power (ADP) product to better point out the significant improvement MIG synthesis brings to light. SWD circuits synthesized via MIGs have 1.30× smaller ADP than SWD circuits synthesized via traditional AIGs. This is thanks to the SWD delay improvement enabled by MIGs. As compared to the 10-nm CMOS technology, SWD circuits synthesized by MIGs have 17.02× smaller ADP, offering an
 ultra-low power, compact SWD implementation with reduced penalty in delay.

B. RRAM-MIG

In Section II, we have seen that RRAM elementary devices can realize a majority voting operation in addition to storing the result. Therefore, the MIG data structure is the native logic abstraction for RRAM in-memory computation.

To demonstrate the efficacy of the RRAM-MIG coupling, we map a lightweight cryptography block cipher [23] on a RRAM array using MIG-based design techniques [15]. The target cryptography block cipher is PRESENT, originally introduced in [23]. The main operations in the PRESENT encryption algorithm are addRoundKey, sBoxLayer, pLayer, and KeyUpdate [23].

For the sake of brevity, we give here details only on the sBoxLayer operation. We refer the interested reader to [23] for details on the other operations. The sBoxLayer operates on 4-bit long strings. Each string is processed individually by a 4-input, 4-output combinational Boolean function, called operator S. In order to map S into the RRAM memory array, we use MIG representation and optimization. The optimization goal is to reduce the number of majority operations.

The S operator is nothing but a Boolean function with primary inputs $p_{i0}, p_{i1}, p_{i2}, p_{i3}$ and primary outputs $p_{o0}, p_{o1}, p_{o2}, p_{o3}$. After optimization, the MIG for such function counts 11 nodes and 4 levels. Each majority node is mapped into a set of primitive RRAM memory/computing instructions. The S operator requires a total of 38 cycles for its operation in the RRAM array.

Using an analogous MIG-mapping approach, all the PRESENT encryption operations can be performed directly on the RRAM array.

The overall performance of the MIG-based PRESENT implementation on the RRAM array has been estimated considering a RRAM technology aligned with the ITRS 2013 predictions. More precisely, we assume a write time of 1 ns and a write energy of 0.1 fJ/bit. Table III summarizes the number of $M_3$ instructions and Read/Write (R/W) cycles required by the different operations of the PRESENT cipher.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Instructions (#M3)</th>
<th>Cycles (#R/W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key copy</td>
<td>80</td>
<td>720</td>
</tr>
<tr>
<td>Cipher copy</td>
<td>64</td>
<td>576</td>
</tr>
<tr>
<td>AddRoundKey</td>
<td>448</td>
<td>4032</td>
</tr>
<tr>
<td>sBoxLayer</td>
<td>608</td>
<td>5472</td>
</tr>
<tr>
<td>pLayer</td>
<td>64</td>
<td>576</td>
</tr>
<tr>
<td>KeyUpdate</td>
<td>760</td>
<td>6840</td>
</tr>
<tr>
<td>PRESENT Block</td>
<td>58 872</td>
<td>455 184</td>
</tr>
<tr>
<td>Energy Throughput</td>
<td>(6J)</td>
<td>(6Kbps)</td>
</tr>
<tr>
<td>PRESENT Block</td>
<td>5.88</td>
<td>120 7</td>
</tr>
</tbody>
</table>

The total number of primitive majority instructions for the encryption of a 64-bit cipher text is 58872 [15]. The total energy required for one block encryption operation is 5.88 pJ. The total throughput reachable by the system is 120.7 kbps, making it competitive to silicon implementations [23].

V. CONCLUSIONS

In this paper, we considered the relation between logic synthesis and emerging nanotechnologies. We reviewed a class of promising nanodevices whose logic abstraction is a majority voter. We demonstrated that new logic synthesis techniques, natively supporting this abstraction, are the technology enablers as they allow designers to validate nanotechnologies on large-scale benchmarks. New research in logic synthesis is becoming essential to permit a fair evaluation on nanotechnologies with different logic abstractions than standard CMOS.

ACKNOWLEDGMENT

This research was partially supported by ERC-2009-AdG-246810 and SNSF-200021-146600.

REFERENCES

[21] ABC synthesis tool - available online.