Synthesis of Normally-Off Boolean Circuits: An Evolutionary Optimization Approach Utilizing Spintronic Devices

Arman Roohi, Ramtin Zand, Ronald F DeMara
Department of Electrical and Computer Engineering
University of Central Florida, Orlando, Florida, 32816
E-mail: {aroohi,ramtinmz}@knights.ucf.edu, ronald.demara@ucf.edu

Abstract—In this paper, we develop an evolutionary-driven circuit optimization methodology, which can be leveraged for the synthesis of spintronic-based normally-off computing (NoC) circuits. NoC architectures distribute nonvolatile memory elements throughout the CMOS logic plane, creating a new class of fine-grained functionally-constrained synthesis challenges. Spin-based NoC circuits synthesis objectives include increased computational throughput and reduced static power consumption. Our proposed methodology utilizes Genetic Algorithms (GAs) to optimize the implementation of a Boolean logic expression in terms of area, delay, or power consumption. It first leverages the spin-based device characteristics to achieve a primary semi-optimized implementation, then further performance optimization is applied to the implemented design based on the NoC requirements and optimization criteria. As a proof-of-concept, the optimization approach is leveraged to implement a functionally-complete set of Boolean logic gates using spin Hall effect (SHE)-magnetic tunnel junctions (MTJs), which are optimized for both power and delay objectives. NoC synthesis methodologies supporting NoC circuit design of emerging device and hybrid CMOS logic applications. Finally, Simulation results and analyses verified the functionality of our proposed optimization tool for NoC circuit implementations.

Keywords—Normally-off computing (NoC), Evolutionary algorithm, Majority logic, Spintronic devices, spin Hall Effect (SHE), Magnetic Tunnel Junction (MTJ).

I. Introduction

Although scaling of CMOS transistors has enabled the proliferation of computational technology since the 1960s, transistors will stop shrinking by the early 2020s, due to limitations of CMOS scaling [1], which has resulted in considerable research on emerging device technologies. Among promising devices, 2014 magnetism roadmap [2] identifies Spintronics devices such as magnetic tunnel junctions (MTJs) as capable beyond-CMOS candidates due to their non-volatility, near-zero standby power, and high integration density [3]. MTJ-based devices have been extensively researched in memory [4], [5] and functional building blocks [6]. Realizing large scale logic circuits remains a serious challenge [7], due to their high switching energy. Although spin-based devices such as MTJs suffer from higher switching energy consumption compared to CMOS, their intrinsic non-volatility characteristics make them a promising candidate to be utilized within normally-off computing (NoC) architectures [8]. NoC systems have attracted considerable attention in recent years due to their near-zero standby power dissipation, as well as their resiliency toward unstable power sources and power failures [9]. The primary building blocks of NoC architecture include non-volatile flip-flops, non-volatile memories, and non-volatile gates and logic circuits. Feasibility of a spintronic NoC system has been demonstrated in [10], verifying its capability to increase computational throughput, while reducing the energy consumption.

The unifying computational mechanism that underlies most spintronic devices is an accumulation mode operation that enables the realization of majority logic functions as basic computational building blocks. Spin-based majority gates (MGs) can be leveraged to realize Boolean gate implementations within a gate library, which can be utilized for NoC systems. Previous optimization and synthesis approaches for MG-based design mainly focus on a single technology implementation such as quantum-dot cellular automata (QCA) [11] and all-spin logic (ASL) [7]. However, hybrid CMOS/spintronic circuits require specific optimization methodology that considers the characteristics of both CMOS and spin-based devices.

In this paper, we propose a baseline optimization methodology to design spin-based NoC circuits. We implement an MG library containing a functionally-complete set of Boolean logic gates required for implementing representative NoC designs. First, the Boolean logic expressions are decomposed into their minterms. Then, minterms are mapped into a tree, which its root can be a 3-input or 5-input MG. Finally, genetic algorithms (GAs) are leveraged to optimize the implementation of a targeted Boolean expression using only 3-input and 5-input MGs based on the spin-based device characteristics and optimization criteria, i.e. area, delay, or power. In summary, the novel contributions of this paper are as follows: 1) an optimization tool is developed for generating optimized spin-based NoC systems using MGs, 2) the proposed methodology incorporates multilevel MG network without using inverters, 3) complementary modifications are applied to the optimization tool to realize and tune NoC circuit benefits, and 4) a spin-Hall effect (SHE) MG library is developed as a proof of concept.

II. Spin-based Normally-Off Computing Circuit

In recent years, NoC architectures have been investigated as a promising approach that has to potential to overcome the existing power-consumption issues limiting the performance improvement in VLSI systems [9]. In spin-based NoC architectures, spintronic devices such as MTJs are

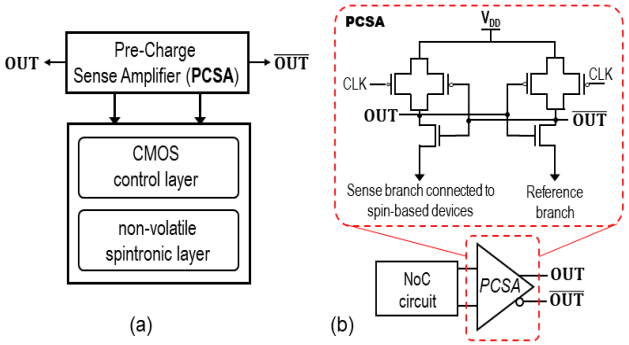


Fig. 1. (a) Generalized layer-structured spin-based NoC circuit, and (b) circuit structure of PCSA.

distributed throughout the CMOS logic layer. An MTJ is composed of two ferromagnetic layers, free layer and fixed layer, which are separated by a thin insulator. The polarization state of the free layer can readily change, whereas fixed layer is magnetically-pinned. Hence, there are two stable magnetization configurations, parallel (P), i.e. low resistance state, and anti-parallel (AP), i.e. high resistance state. P and AP states denote "0" and "1" in binary information, respectively. Spin-transfer torque (STT) switching is one of the most promising switching approaches for MTJs, due to its simplicity; however, it suffers from high write energy consumption [12]. Recently, 3-terminal spin-Hall effect (SHE)-MTJ has been introduced as an alternative offering promising benefits such as energy-efficient and high-speed write, as well as decoupled read and write paths [3]. A SHE-MTJ structure includes an MTJ that its free layer is directly connected to a heavy metal (HM). The magnetization direction of the free layer can be oriented according to the direction of an applied charge current that flows through HM. Non-volatility, normally-off, and instant-on characteristics of SHE-MTJ devices enable power-gating at a fine-granularity, which significantly reduces the leakage power dissipation. Moreover, the MTJ devices can be vertically stacked at the back-end process of CMOS fabrication resulting in a small area overhead.

As shown in Fig. 1(a), CMOS-based logic control layer is one of the most important components in spin-based NoC circuits, which enables the implementation of a Boolean logic within the non-volatile spintronic layer. Moreover, read operation for spin-based devices are performed by leveraging the widely-used pre-charge sense amplifier (PCSA) described in [13]. Figure 1(b) shows the PCSA consisting four PMOS transistors, two NMOS transistors which connects the PMOS transistors to the spin-based devices, and one NMOS transistor that connects the circuit to ground (GND). Sensing process in PCSA has two states called pre-charge and discharge. In pre-charge state, clock signal (CLK) is set to "0" which turns on the PMOS transistors, thus OUT and $\overline{\text{OUT}}$ are set to V_{DD}. In discharge state, CLK is set to "1" which turns off the PMOS transistors and turn on the NMOS transistor. Therefore, the circuit is connected to GND and disconnected from V_{DD}. Hence, the voltages of the OUT and $\overline{\text{OUT}}$ terminals, which are connected to the sense and reference MTJs, respectively, start to discharge. The reference MTJ resistance is designed in a manner such that its value in parallel configuration is between low resistance, R_P, and high resistance, R_{AP}, of the sense MTJ, $R_{(\text{reference MTJ})} = 1/2(R_{(\text{AP-sense MTJ})} + R_{(\text{P-sense MTJ})})$. Due to the difference between the MTJs' resistances, the discharge speed is different. The discharge state continues till the voltage drop in lower resistance branch results in a voltage difference larger than the threshold voltage between the source and gate of the PMOS transistor in higher resistance branch. Then, the higher resistance branch will be connected to V_{DD} and lower resistance branch will be discharged to GND.

As mentioned above, spintronic devices enable the realization of majority logic functions as basic computational building blocks. AND/OR gates can be readily implemented by majority gates, for instance by affixing one of the inputs of a 3-input MG to "1" or "0" states upon demand during the circuit operation, then a 2-input OR gate or a 2-input AND gate can be realized, respectively. The technology mapping process in most of the previously proposed synthesis techniques focuses on technologies which can be readily concatenated such as QCA [11] and ASL [7]. However, in spin-based NoC circuits, a CMOS-based sensing, and control unit is utilized to convert magnetic orientation and voltage signals to transfer data from one spin-based logic layer to another. The collaborative operation of two technologies with different area, delay and power characteristics, necessitate new synthesis paradigms, which includes both devices and conversion blocks characteristics. Hence, a novel methodology is proposed herein to realize an optimized spin-based NoC system based on the optimization criteria, i.e. area, delay, or power. To the best of our knowledge, this is the first work addressing the synthesis and optimization requirements of spin-based NoC architecture implementation.

III. Proposed Approach

In this section, our optimization methodology for spin-based NoC circuits is described, as shown in Fig. 2. Spin-based components are utilized for storing and computing, whereas CMOS-based elements are used for implementing logic in storage elements, as well as to conduct the read operation. Required sensing scheme is provided by PCSA, which generates both output (OUT) and invert of the output ($\overline{\text{OUT}}$). Hence, the intrinsic structure of the proposed spin-based NoC cell includes one MG, which provides a functionally-complete unit. Thus, in our proposed optimization methodology, the implementation cost of inverter gate is equal to zero. Our proposed evolutionary approach includes two levels of optimization to reduce the convergence time: *Technology-Dependent Optimization* and *Performance Optimization*.

A. Technology-Dependent Optimization

In the first level of the optimization, Genetic Algorithms (GAs) are utilized to optimize the implementa-

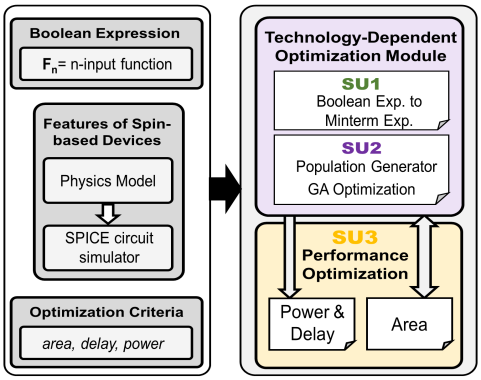


Fig. 2. Schematic of proposed evolutionary approach to realize MG-based NoC circuit.

tion of a Boolean logic expression in terms of area, delay, or power. Although GAs were selected for the proof-of-concept framework due to their rapid prototyping ability to achieve multi-objective optimization, extensions are identified subsequently in Section 3.2 and Section 5. It leverages the spin-based device characteristics as inputs to achieve a semi-optimized implementation. First, a transforming unit, which is Synthesis Unit (SU1), decomposes a Boolean expression into its minterms. Then, the generated minterm expression is applied to a mapping and optimization unit, which is Synthesis Unit 2 (SU2), along with optimization criteria and characteristics of spin-based building blocks. For instance, in a design with 3-input and 5-input spin-based MGs as building blocks, first MGs are separately implemented, and their related delay, area, and power consumption are measured. Then, the obtained results are leveraged to define their implementation cost within the optimization methodology. Finally, the GAs are utilized to optimize a Boolean logic implementation based on the optimization criteria and the obtained implementation cost of the spin-based building blocks in SU3.

GAs are biologically-inspired algorithms which are one of the most popular multi-objective tools due to their ability to empirically explore complex search spaces regardless of their gradient or higher derivatives to realize suitable designs in a design prototyping environment. As shown in Fig. 3, the mapping and optimization unit involves three main steps as explained in the following sequence.

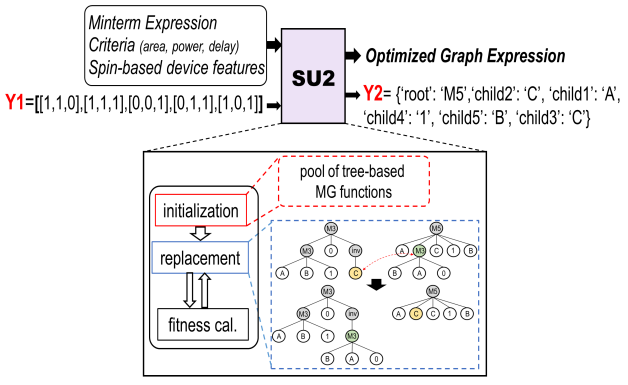


Fig. 3. Operations of F1 and F2 blocks for A·B+C in technology-dependent optimization process.

1) *Initialization*: an initial set of tree-based structures are created, in which each parent can have three or five random children. Each of the trees is a chromosome and the complete set is called initial population. The GA convergence time could be adjusted by the population size and range chromosome variety. Extending the population size leads to increasing the variety of chromosomes, which is limited to some upper bounds. However, this extension leads to an increase in the total processing time of GA.

2) *Fitness Evaluation*: To evolve the population toward better solutions the fitness of each chromosome is evaluated. Therefore, a fitness function is defined to assign a fitness value to the chromosomes. This evaluation is required in several occasions during the operation of the algorithm including parent selection, and constitution of the next generation population. Herein, the fitness function is expressed by $f(t_i) = N(m, t_i)/(\text{length of } t_i)+1/N(r, t_i)+1/(\text{number of gate})$, where m is the applied input minterms, $N()$ is a function that calculates the number of minterms in m, which is implemented by t_i tree, and r is the remainder of the minterms that should not be implemented. As it can be seen, the fitness function has an inverse relation with the length of the tree, which results in producing balanced trees. It enables performing a larger number of parallel operations at each level leading to power and delay optimized implementations.

3) *Replacement*: GA generates new offspring(s) from selected parents with a defined probability to achieve improved solutions to the problem. Herein, the sub-tree has been selected as the crossover operator, which selects two nodes and exchanges their sub-trees rooted from the selected nodes. Moreover, a mutation operation is performed by creating randomly generated chromosomes and exchanging them by a number of randomly selected chromosomes with a specific probability. The mutation operation is applied to avoid the algorithm being trapped in a local optimum. Tournament selection has been utilized in order to select the parents for crossover and mutation operators. The algorithm stops when no improvement in fitness function happens after more than 100 generations. The output of this mapping and optimization unit is an optimized graph expression, as shown in Fig. 3. Algorithm 1 and Fig. 4(a) illustrate the evolutionary approach leveraged in the proposed technology-dependent optimization methodology.

B. Power and Delay Optimization

The computational mechanism underlying most spintronic devices is charge accumulation mode operation. Therefore, increasing the input current decreases the operation's delay at the expense of increasing power consumption. As it was mentioned in Section 2, AND/OR gates can be readily implemented by majority gates, for instance, to realize a 2-input OR (AND) gate, one of the input transistors of a 3-input MG should be ON (OFF). Therefore, disjunction operator (OR) has larger power consumption than conjunction operator (AND), due to a higher number

Algorithm 1 Mapping and Optimization Algorithm

procedure TRANSFORMING(1)
  Input: Boolean Expression (Gate functionality)
  Generate truth table of an applied n-input function
  Convert Boolean expression to minterms
  Output: Minterms expression set
end procedure
procedure MAPPING AND OPTIMIZATION ALGORITHM
  Input$_1$: Generated minterms
  Input$_3$: Optimization criteria (area, delay, power)
  Step1: *Initialization*
  Define implementation cost for 3-input MG (M3) and 5-input MG (M5)
  Adjust optimization factor based on cost and applied criterion
  Generate pool of tree-based MG functions
  Label root with M3 (M5) & add node with respect to root
  Step2: *Fitness Evaluation*
  $f(t_i) = \frac{N(m,t_i)}{(length(t_i))} + \frac{1.0}{N(r,t_i)+1} + \frac{1.0}{\#gate+1}$
  Compute $f(t_i)$ for the constructed tree t_i
  if progress in f is less than threshold **or** constraint on the upper limit number of generations is reached **then**
    Break
  end if
  Step3: *Crossover*
  P_1, P_2= Randomly select two branches from different trees
  if root (P_1) & root (P_2) are M3 or M5 **then**
    Replace selected sub-trees with each other
  else
    Repeat Step3 until reach a leaf then Break
  end if
  Repeat Step2
  Output: Optimized graph expression
end procedure

of ON transistors that leads to the higher input current. Since the implementation cost of an inverter is equal to zero in our optimization methodology, disjunction operators and conjunction operators can be replaced according to the well-known De Morgan's law without any redundancy cost. Hence, a third functional unit (SU3) is added to the optimization tool, which replaces the OR (AND) operations by AND(OR)-inverter operations within the logic implementation to reduce power (delay). Algorithm 2 describes SU3 functionality, which first takes the optimized tree obtained by SU2. Then, it executes a pre-order traversal scheme to visit a node, check its value, and update it, recursively. All of the trees or sub-trees with a root labeled M3 or M5 are examined to find any leaf with value "1". Then, it replaces "1" with "0" and inverts all of the remaining leaves with the same parent. Finally, it uses the $\overline{\text{OUT}}$ signal instead of OUT to invert the whole tree or sub-tree. An example of a power-optimized implementation of (A+B+C+D) expression and its corresponding normalized simulation results are shown in Fig. 4(b) and Fig. 4(d), respectively.

C. Area Optimization

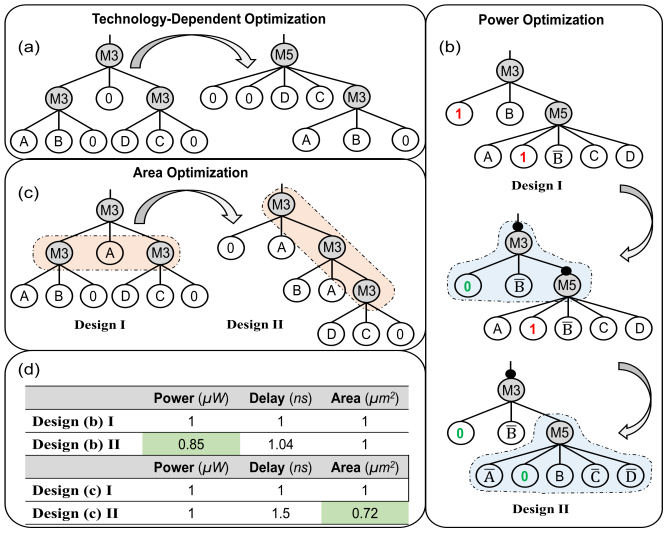
In the proposed NoC architecture each MG node requires one PCSA. Therefore, the number of PCSAs required for each layer depends on the number of MG nodes existing in that non-volatile spintronic layer. On the other hand, PCSAs can be shared between different non-volatile spintronic layers. Thus, the number of PCSAs required for implementing a NoC circuit is equal to the maximum number of MG nodes utilized in any non-volatile spintronic layer. However, According to the fitness function described in Section 3.1, trees with balanced structure have larger fitness value. Although the balanced tree structure, e.g. shown in Design I of Fig. 4(c), provides an optimized implementation in term of delay or power consumption, it requires a larger number of PCSAs due to having more MG nodes in second layer leading to higher area overhead. Hence, for area optimization we have modified the fitness function to $f(t_i) = N(m,t_i)/(\text{length of } t_i) + 1/N(r,t_i) + 1/(\text{number of gate}) + 1/(n_{MG}+1)$, where n_{MG} is the maximum number of PCSAs in the implemented design. The procedure leveraged a breadth-first search technique to find the maximum number of MGs in one level. Therefore, the optimization methodology creates an unbalanced tree with less number of MG nodes in each layer as shown in Design II of the Fig. 4(c). Thus, only a single PCSA is required to implement the A·(B+CD) Boolean expression, which results in decreased area consumption while increasing delay, as shown in Fig. 4(d). This is caused by the increased sequential operations required to deliver the output of each logic layer to the next one.

Algorithm 2 Power Optimization

Input: Technology-Dependent Optimized tree (t)
procedure PRE_ORDER_TRAVERSAL (T)
  while t is not NULL **do**
    if t is tree **then**
      x ← root(t)
      if x == (M3 or not M3) **then**
        for j in 3 **do**
          pre-order traversal (sub-tree j)
        end for
      else if x== (M5 or not M5) **then**
        for j in 5 **do**
          pre_order_traversal (sub-tree j)
        end for
      end if
    else
      x ← value(t)
      if x==1 **then**
        `Update` (invert) all children with parent(x)
        `Select` $\overline{\text{OUT}}$ instead of OUT (vice versa) to output
      end if
      return
    end if
  end while
end procedure
Output: Power optimized tree

IV. Proof-of-Concept: SHE-Based MG

In this section, we have leveraged our proposed optimization methodology to implement a functionally-complete set of Boolean logic gates using SHE-MTJ, as a proof-of-concept for optimized spin-based NoC circuit design. Herein, the SHE-MTJ model developed in [5] is utilized to design a 3-input and 5-input MGs, as shown in Fig. 5(a) and Fig. 5(b), respectively. In accordance with Fig.5, PMOS transistors are leveraged to produce the input charge currents, while a PCSA is used to sense the SHE-MTJs' states. The dimensions of the 3-input (5-input) SHE-MG is designed in a manner such that at least two (three) out of the three (five) input transistors should be



	Power (μW)	Delay (ns)	Area (μm^2)
Design (b) I	1	1	1
Design (b) II	0.85	1.04	1
	Power (μW)	**Delay (ns)**	**Area (μm^2)**
Design (c) I	1	1	1
Design (c) II	1	1.5	0.72

Fig. 4. (a) technology-dependent optimization for F= A.B.C.D, (b) power optimization for F= A+B+C+D, (c) area optimization for F= A·(B+CD), and (d) comparison results for Designs in (b) and (c).

ON to produce a current amplitude greater than the critical current of SHE-MTJ. The functionality of the SHE-MGs is verified by SPICE circuit simulation. Table 1 and Table 2 list the write operation performance of the 3-input and 5-input MGs, respectively. Switching the 5-input SHE-MG requires larger number of ON transistors leading to higher input charge current, which increases the power consumption, while decreasing switching delay [14]. Table 3 lists read operation results for 3-input and 5-input SHE-MG. MGs can be readily configured to perform AND or OR gates by fixing one (two) of three (five) inputs to "0" or "1", respectively. Figure 5(c) depicts 2-input and 3-input AND operation using 3-input and 5-input SHE-MGs, respectively.

To implement spin-based NoC cells, the 3-input and 5-input SHE-MGs are defined as functional blocks and their characteristics are applied to the optimization tool. The proposed evolutionary approach is leveraged to implement a functionally-complete set of Boolean logic gates. For each of the Boolean functions, power and area optimization resulted in an identical implementation, while the delay optimization generated a different implementation. As

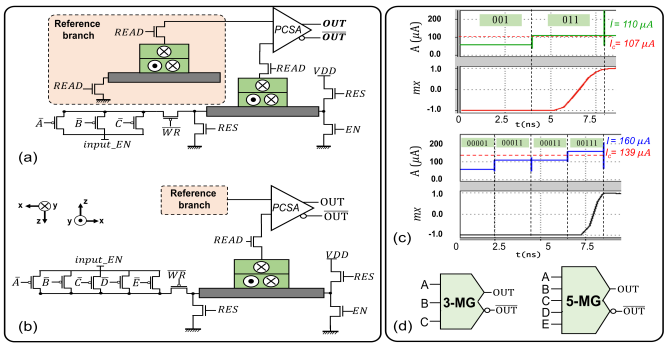


Fig. 5. (a) SHE-MTJ based 3-input MG, (b) SHE-MTJ based 5-input MG, (c) simulation results for 2-input OR logic, and 3-input AND logic using (a) and (b), respectively, and (d) schematic of 3-input and 5-input SHE-MG.

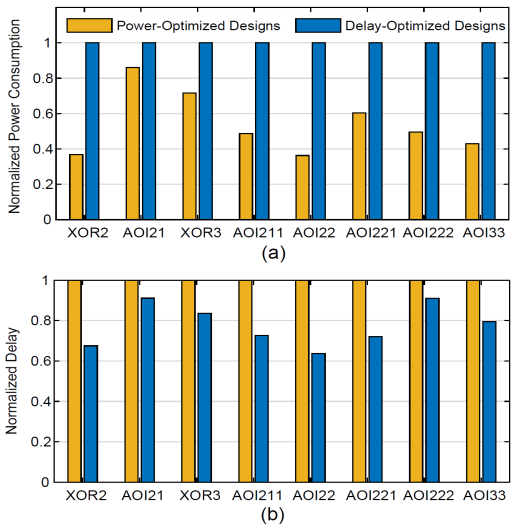


Fig. 6. Normalized results for (a) power consumption, and (b) delay leveraging two optimization approaches.

listed in Table 4, there are higher number of "0" inputs in power-optimized structures, while delay-optimized implementations include more "1" inputs. This is achieved by the performance optimization method introduced in Section 3, which leads to smaller input current for power-optimized and larger input current for delay-optimized implementations. Moreover, Fig. 6 exhibits the normalized power dissipation and delay for selected 2-input to 6-input Boolean logic circuits, which are implemented using the proposed power-optimization and delay-optimization paradigms. The results verify the efficiency of our optimization methodology for NoC circuit implementations.

TABLE I

WRITE PERFORMANCE OF 3-INPUT SHE-MG.

Inputs A	B	C	No.†	Write Operation Power (μW)	Delay (ns)
0	0	0	1	0.124	-
0	0	1	3	67.7	-
0	1	1	3	130.3	2.98
1	1	1	1	187.9	1.91
Average				**97.7**	**2.84**

(†)Number of 3-input Boolean expression with same features.

TABLE II

WRITE PERFORMANCE OF 5-INPUT SHE-MG.

Inputs A	B	C	D	E	No.†	Write Operation Power (μW)	Delay (ns)
0	0	0	0	0	1	0.32	-
0	0	0	0	1	5	67.2	-
0	0	0	1	1	10	128.4	-
0	0	1	1	1	10	185.4	2.22
0	1	1	1	1	5	236.64	1.69
1	1	1	1	1	1	283.68	1.39
Average						**154.4**	**2.11**

(†)Number of 5-input Boolean expression with same features.

TABLE III

READ PERFORMANCE OF SHE-MGS.

Design	Read 0 Power	Read 0 Delay	Read 1 Power	Read 1 Delay
SHE-based 3-MG	1.64 μW	20 ps	1.28 μW	20 ps
SHE-based 5-MG	1.35 μW	21 ps	1.53 μW	22 ps

TABLE IV

Optimized implementation of functionally-complete set of Boolean logic gates using SHE-MGs.

Standard Functions	Efficiency Criteria	
	Power*	**Delay**
1. A.B (or $\overline{\text{A.B}}$)	$\mathbf{M}(A,B,0)$	$\overline{\mathbf{M}(\overline{A},\overline{B},1,0,1)}$
2. A+B (or $\overline{\text{A}+\text{B}}$)	$\overline{\mathbf{M}(\overline{A},\overline{B},0)}$	$\mathbf{M}(A,B,0,1,1)$
3. $\overline{\text{A}}$.B+A.$\overline{\text{B}}$ (or $\overline{\text{A}}$.$\overline{\text{B}}$+A.B)	$\mathbf{M}(\overline{\mathbf{M}(\overline{A},B,0)},\overline{\mathbf{M}(\overline{B},A,0)},0)$	$\mathbf{M}(\overline{\mathbf{M}(A,B,A,B,1)},\overline{B},\overline{\mathbf{M}(\overline{A},\overline{B},\overline{B},1,\overline{A})},B,1)$
4. A.B.C (or $\overline{\text{A}}$.$\overline{\text{B}}$.$\overline{\text{C}}$)	$\mathbf{M}(A,B,C,0,0)$	$\overline{\mathbf{M}(\overline{A},\overline{B},\overline{C},1,1)}$
5. A+B+C (or $\overline{\text{A}+\text{B}+\text{C}}$)	$\overline{\mathbf{M}(\overline{A},\overline{B},\overline{C},0,0)}$	$\mathbf{M}(A,B,C,1,1)$
6. $\overline{(\text{A.B})+\text{C}}$	$\mathbf{M}(\overline{A},\overline{B},\overline{C},\overline{C},0)$	$\overline{\mathbf{M}(A,B,C,C,1)}$
7. $\overline{(\text{A}+\text{B}).\text{C}}$	$\overline{\mathbf{M}(A,B,C,C,0)}$	$\mathbf{M}(\overline{A},\overline{B},\overline{C},\overline{C},1)$
8. A.$\overline{\text{C}}$+B.C	$\mathbf{M}(\mathbf{M}(\overline{A},\overline{C},0)\mathbf{M}(0,B,C),\overline{C})$	$\mathbf{M}(A,B,\overline{\mathbf{M}(\overline{B},\overline{C},A,1,1)},\mathbf{M}(1,\overline{B},1,A,C),\overline{C})$
9. A.B+B.C+A.C	$\mathbf{M}(A,B,C)$	$\mathbf{M}(A,B,C,1,0)$
10. A⊕B⊕C	$\mathbf{M}(\mathbf{M}(A,B,\overline{C}),C,\overline{\mathbf{M}(B,A,C)})$	$\mathbf{M}(A,\overline{\mathbf{M}(A,B,C,0,1)},B,C,\overline{\mathbf{M}(A,B,C,0,1)})$
11. A.B.C.D (or $\overline{\text{A.B.C.D}}$)	$\mathbf{M}(D,0,\mathbf{M}(A,B,\overline{D},C,0))$	$\overline{\mathbf{M}(\overline{A},\overline{D},1,1,\overline{\mathbf{M}(A,\overline{B},\overline{C},\overline{D},1)})}$
12. A+B+C+D (or $\overline{\text{A}+\text{B}+\text{C}+\text{D}}$)	$\overline{\mathbf{M}(\overline{B},0,\mathbf{M}(\overline{A},B,\overline{C},\overline{D},0))}$	$\mathbf{M}(C,1,\mathbf{M}(\overline{C},B,A,1,D),\overline{B},B)$
13. $\overline{(\text{A.B}+\text{C}+\text{D})}$	$\mathbf{M}(\overline{C},0,\mathbf{M}(\overline{A},\overline{B},\overline{D},\overline{D},0))$	$\overline{\mathbf{M}(C,D,1,1,\mathbf{M}(A,B,D,C,1))}$
14. $\overline{(\text{A}+\text{B}).\text{C.D}}$	$\overline{\mathbf{M}(0,C,\mathbf{M}(A,B,D,D,0))}$	$\overline{\mathbf{M}(\mathbf{M}(\overline{A},\overline{B},\overline{C},\overline{D},1),\overline{C},\overline{D},1,1)}$
15. $\overline{(\text{A.B}+\text{C.D})}$	$\mathbf{M}(\overline{\mathbf{M}(A,B,0)},\overline{\mathbf{M}(C,D,0)},0)$	$\mathbf{M}(C,\overline{\mathbf{M}(1,1,0,\overline{A},\overline{B})},\overline{\mathbf{M}(1,C,D,\overline{A},\overline{B})},D,1)$
16. $\overline{(\text{A}+\text{B}).(\text{C}+\text{D})}$	$\mathbf{M}(\mathbf{M}(A,\overline{B},0),\overline{B},\mathbf{M}(0,\overline{C},\overline{D}))$	$\mathbf{M}(\overline{A},\overline{B},B,\mathbf{M}(\overline{B},\overline{B},\overline{C},\overline{D},1),\mathbf{M}(A,B,\overline{C},\overline{D},1))$
17. $\overline{(\text{A.B}+\text{C.D}+\text{E})}$	$\mathbf{M}(\overline{\mathbf{M}(A,B,\overline{E})},0,\mathbf{M}(\overline{C},\overline{D},\overline{E},\overline{E},0))$	$\mathbf{M}(E,\overline{E},1,\mathbf{M}(B,A,E,\overline{E},\overline{E}),\mathbf{M}(C,1,D,E,E))$
18. $\overline{(\text{A}+\text{B}).(\text{C}+\text{D}).\text{E}}$	$\overline{\mathbf{M}(\mathbf{M}(A,B,E),0,\mathbf{M}(C,D,E,E,0))}$	$\mathbf{M}(E,\overline{E},1,\mathbf{M}(\overline{C},\overline{D},\overline{E},1,\overline{E}),\mathbf{M}(B,A,E,\overline{E},\overline{E}))$
19. $\overline{(\text{A.B}+\text{C.D}+\text{E.F})}$	$\mathbf{M}(\overline{\mathbf{M}(A,B,0)},\overline{\mathbf{M}(C,D,0)},\overline{\mathbf{M}(E,F,0)},0,0)$	$\overline{\mathbf{M}(\overline{\mathbf{M}(\overline{A},\overline{B},1)},\overline{\mathbf{M}(\overline{C},\overline{D},1)},\overline{\mathbf{M}(\overline{E},\overline{F},1)},1,1)}$
20. $\overline{(\text{A}+\text{B}).(\text{C}+\text{D}).(\text{E}+\text{F})}$	$\mathbf{M}(0,\overline{\mathbf{M}(\overline{A},\overline{B},0)},\overline{\mathbf{M}(\overline{C},\overline{D},0)},0,\overline{\mathbf{M}(\overline{E},\overline{F},0)})$	$\mathbf{M}(1,\overline{\mathbf{M}(A,B,1)},\overline{\mathbf{M}(1,C,D)},1,\overline{\mathbf{M}(E,1,F)})$
21. $\overline{(\text{A}+\text{B}+\text{C}).(\text{D}+\text{E}+\text{F})}$	$\overline{\mathbf{M}(\mathbf{M}(\overline{A},\overline{B},\overline{C},0,0),0,\mathbf{M}(\overline{D},\overline{E},\overline{F},0,0))}$	$\mathbf{M}(0,1,\overline{\mathbf{M}(A,B,C,1,1)},1,\overline{\mathbf{M}(D,E,F,1,1)})$

(∗) Power and area optimization resulted in an identical implementation.

V. Conclusion

In this paper, we have developed an evolutionary approach to optimize the implementation of spin-based NoC circuits. The 3-input and 5-input MGs are introduced as functional building blocks and their characteristics are applied to the proposed optimization tool. First, GAs are utilized in our methodology to perform a technology-dependent optimization to generate an optimized implementation based on the obtained characteristics of spin-based building blocks. Then, complementary performance and area optimization are introduced to improve the implementation based on the requirements of the NoC system. As a proof of concept, we have developed and examined 3-input and 5-input MGs using SHE-MTJs and leveraged their characteristics to implement a functionally-complete set of Boolean logic gates. Simulation results, as well as power, delay, and area analyses verified the functionality of our proposed optimization tool for NoC circuits. The future work includes developing more advanced genetic transformations to improve the proposed framework's capability of handling larger scale optimization and synthesis.

References

[1] M. M. Waldrop, "The chips are down for moore's law." *Nature*, vol. 530, no. 7589, pp. 144–147, 2016.

[2] R. L. Stamps, S. Breitkreutz, J. Åkerman, A. V. Chumak, Y. Otani, G. E. Bauer, J.-U. Thiele, M. Bowen, S. A. Majetich, M. Kläui *et al.*, "The 2014 magnetism roadmap," *J PHYS D APPL PHYS*, vol. 47, no. 33, p. 333001, 2014.

[3] X. Fong, Y. Kim, K. Yogendra, D. Fan, A. Sengupta, A. Raghunathan, and K. Roy, "Spin-transfer torque devices for logic and memory: Prospects and perspectives," *IEEE T COMPUT AID D*, vol. 35, no. 1, pp. 1–22, 2016.

[4] R. Zand, A. Roohi, S. Salehi, and R. F. DeMara, "Scalable adaptive spintronic reconfigurable logic using area-matched mtj design," *IEEE T CIRCUITS-II*, vol. 63, no. 7, pp. 678–682, July 2016.

[5] R. Zand, A. Roohi, D. Fan, and R. F. DeMara, "Energy-efficient nonvolatile reconfigurable logic using spin hall effect-based lookup tables," *IEEE T NANOTECHNOL*, vol. 16, no. 1, pp. 32–43, Jan 2017.

[6] A. Roohi, R. Zand, D. Fan, and R. F. DeMara, "Voltage-based concatenatable full adder using spin hall effect switching," *IEEE T COMPUT AID D*, vol. 36, no. 12, pp. 2134–2138, 2017.

[7] Z. Pajouhi, S. Venkataramani, K. Yogendra, A. Raghunathan, and K. Roy, "Exploring spin-transfer-torque devices for logic applications," *IEEE T COMPUT AID D*, vol. 34, no. 9, pp. 1441–1454, 2015.

[8] K. Ando, S. Fujita, J. Ito, S. Yuasa, Y. Suzuki, Y. Nakatani, T. Miyazaki, and H. Yoda, "Spin-transfer torque magnetoresistive random-access memory technologies for normally off computing," *J APPL PHYS*, vol. 115, no. 17, p. 172607, 2014.

[9] T.-K. Chien, L.-Y. Chiou, C.-C. Lee, Y.-C. Chuang, S.-H. Ke, S.-S. Sheu, H.-Y. Li, P.-H. Wang, T.-K. Ku, M.-J. Tsai *et al.*, "An energy-efficient nonvolatile microprocessor considering software-hardware interaction for energy harvesting applications," in *VLSI Design, Automation and Test (VLSI-DAT), 2016 International Symposium on*. IEEE, 2016, pp. 1–4.

[10] T. Kawahara, "Scalable spin-transfer torque ram technology for normally-off computing," *IEEE DES TEST COMPUT*, no. 1, pp. 52–63, 2010.

[11] K. Walus, G. Schulhof, G. Jullien, R. Zhang, and W. Wang, "Circuit design based on majority gates for applications with quantum-dot cellular automata," in *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Eighth Asilomar Conference on*, vol. 2. IEEE, 2004, pp. 1354–1357.

[12] Y. Zhang, X. Wang, Y. Li, A. K. Jones, and Y. Chen, "Asymmetry of mtj switching and its implication to stt-ram designs," in *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 2012, pp. 1313–1318.

[13] W. Zhao, C. Chappert, V. Javerliac, and J.-P. Noziere, "High speed, high stability and low power sensing amplifier for mtj/cmos hybrid logic circuits," *IEEE T MAGN*, vol. 45, no. 10, pp. 3784–3787, 2009.

[14] R. Zand, A. Roohi, and R. F. DeMara, "Energy-efficient and process-variation-resilient write circuit schemes for spin hall effect mram device," *IEEE T VLSI*, vol. 25, no. 9, pp. 2394–2401, 2017.