High Performance MRAM-Based Stateful Logic

Hiwa Mahmoudi, Thomas Windbacher, Viktor Sverdlov, and Siegfried Selberherr
Institute for Microelectronics, TU Wien, Gußhausstraße 27–29/E360, A–1040 Wien, Austria
E-mail: {mahmoudi|windbacher|sverdlov|selberherr}@iue.tuwien.ac.at

***Abstract*—Static power due to the leakage currents has become a major concern in CMOS logic circuits as the technology is scaled down. Introducing non-volatility into logic circuits is a promising solution offering zero standby power and instant-on applications. Recently, spin-transfer torque magnetoresistive random-access memory (STT-MRAM) circuits have been presented to enable stateful logic by implementing reprogrammable- and implication-based magnetic tunnel junction logic operations. In this work we describe tradeoffs in the design of MRAM-based stateful logic architectures. It has been shown that although the implication logic outperforms the reprogrammable architecture, a combination of these two architectures reduces the number of required logic steps and the energy consumption, however, at the cost of reduced reliability. MRAM-based logic is also well suited for high performance parallel non-volatile computations as it is shown by an example.**

***Index Terms*—magnetic tunnel junction (MTJ), material implication (IMP), non-volatility, magnetoresistive random-access memory (MRAM), reprogrammable logic, spin transfer torque (STT), stateful logic**

I. Introduction

Fundamental physical limitations such as leakage, high power densities, process variability, and soaring costs will bring the scaling of the classical CMOS devices to an end [1]. Therefore, investigating possible alternative technologies to replace or at least to supplement CMOS is important to further enhance the performance of logic devices and circuits. Distributing non-volatile memory elements over a CMOS logic-circuit is expected to address some of the above-described limitations by providing ultra-low power and fast operation as it eliminates the static leakage (standby) power dissipation and reduces interconnection delay [2].

Spintronic devices, especially MgO-based magnetic tunnel junctions (MTJs), are strong candidates to replace CMOS-based memory due to their non-volatility and compatibility with CMOS technology [3]. The spin-transfer torque (STT) switching technique eliminates the physical mismatch between reading and writing of the MTJs and allows more scalable and smaller switching energies [4]. Magnetoresistive random-access memory (MRAM) with STT-MTJs as memory elements combines the speed of static RAMs (SRAMs), the density of dynamic RAMs (DRAMs), and the non-volatility of flash memory and has all the characteristics of a universal memory [5]. Furthermore, MTJ technology is attractive for building logic configurations which combine non-volatile memories and logic circuits (so-called logic-in-memory architecture) to overcome the leakage power issue [6], [7].

Recently, it has been demonstrated that direct communication between STT-MTJs enables intrinsic logic-in-memory (also known as "stateful" logic [9]), for which the MTJs are used as the main devices for logic computations [10], [11] and the need for intermediate sensing amplifiers is eliminated. This allows to reduce the power consumption, interconnection delay, as well as the device count. In [10] and [11] MTJ-based implication (Fig. 1a) and reprogrammable (Fig. 1b) logic gates are used to realize fundamental Boolean logic operations. In these gates the initial resistance states of the MTJs act as logic inputs. The final resistance state of the target/output MTJs represents the logic result of the gates. Depending on the input logic state, each gate provides a state dependent (conditional) STT switching behavior of the output MTJs. This conditional switching behavior corresponds to a fundamental logic operation provided by the gate as a basic operation. Due to an easy integration of MTJs on top of a CMOS circuit into a one transistor/one MTJ (1T/1MTJ) cell (Fig. 1c), the MTJ logic gates can be extended to MRAM-based logic arrays for large-scale non-volatile applications [12]. In [13] we showed that, for complex logic functions, the implication logic outperforms the reprogrammable architecture from both reliability and power consumption point of views. In this work, we demonstrate that although the implication logic is computationally complete, its combination with the reprogrammable gate reduces the number of required logic steps as complex logic functions

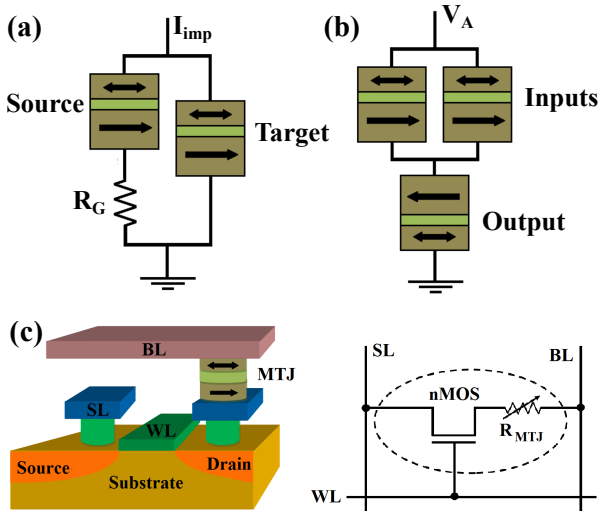

Fig. 1. MTJ-based implication (a) and reprogrammable (b) logic gates. (c) One-transistor/one-MTJ (1T/1MTJ) structure used in common STT-MRAM architecture [8].

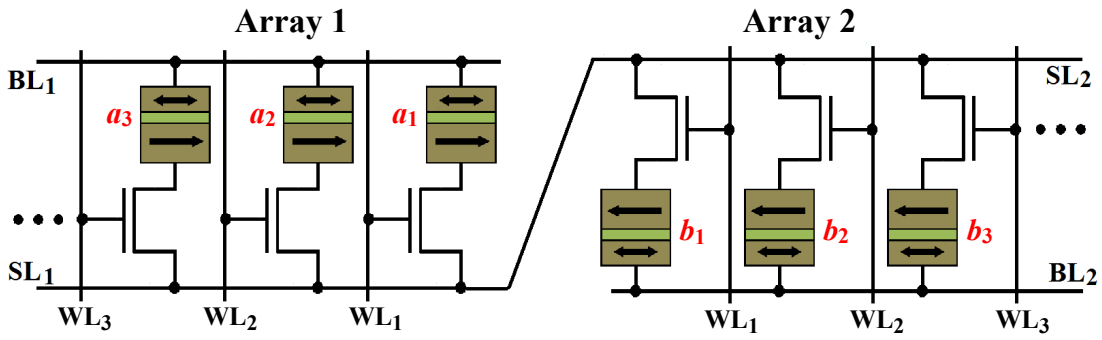

Fig. 2. MRAM logic architecture including two common STT-MRAM arrays [8] connected in series. The stored data (resistance state) in the MTJs are represented as a_i and b_i for Array 1 and Array 2, respectively.

are designed based on the basic operations from both architectures. Thus, the total time and the energy consumption are decreased. However, the total error probability increases due to the lower reliability of the reprogrammable architecture. It is shown that the time performance of the MRAM-based logic architectures is significantly improved by the parallelization of computations.

II. MRAM-Based Stateful Logic

In the common STT-MRAM architecture [8], the 1T/1MTJ cells are coupled in parallel between the current-carrying source lines (SLs) and bit lines (BLs) (Fig.2). Each cell contains one MTJ to store binary data and an access transistor to control the current flowing through the MTJ. In order to select a specific MTJ for read/write operations, a proper voltage signal is applied to the gate terminal of the corresponding access transistor through a word line (WL). In a memory (read/write) mode, a selecting voltage (V_s) is applied to an arbitrary WL and a proper current (or voltage) signal is applied to the current-carrying lines to readout/switch the resistance state of the selected MTJ.

In the implication logic mode [12], a selecting (V_s) and a pre-selecting ($V_{ps} < V_s$) are applied to two arbitrary WLs in one array. As V_{ps} is lower, its access transistor exhibits a higher channel resistance and thus provides the structural asymmetry required for the implication gate (R_G in Fig. 1a). In the reprogrammable logic mode [14], two MTJs are simultaneously selected as inputs in one array and one MTJ is selected as the output in the other array. Due to the serial connection of the arrays, the three simultaneously selected MTJs form the circuit topology required for the reprogrammable gate shown in Fig.1b. By applying a proper voltage difference (V_A) to the BLs of the arrays, the result of a basic reprogrammable logic operation is written in the output MTJ.

In combination with writing logic '0' and logic '1' operations, both implication and reprogrammable logic architectures are computationally complete. Thus, any Boolean logic function can be computed in a series of sequential steps using these architectures. In fact, as only one operation at a time can be performed in this logic framework, complex logic functions are implemented by using a set of subsequent operations. As an example, in the implication logic architecture the implementation of an XOR function ($a_3 \leftarrow a_1$ XOR a_2) includes:

$$
\begin{aligned}
&\text{TRUE}: a_3, a_4 = 1\\
&\text{NIMP}: \overline{a_3 \rightarrow a_1} \equiv \{a_3' = a_3.\overline{a_1} = \overline{a_1}\}\\
&\text{NIMP}: \overline{a_4 \rightarrow a_2} \equiv \{a_4' = a_4.\overline{a_2} = \overline{a_2}\}\\
&\text{NIMP}: \overline{a_2 \rightarrow a_1} \equiv \{a_2' = a_2.\overline{a_1}\}\\
&\text{NIMP}: \overline{a_4 \rightarrow a_3} \equiv \{a_4' = a_4.\overline{a_3} = \overline{a_2}.a_1\}\\
&\text{TRUE}: a_1 = 1\\
&\text{NIMP}: \overline{a_1 \rightarrow a_2} \equiv \{a_1' = a_1.\overline{a_2} = 1.(\overline{a_2.\overline{a_1}}) = \overline{a_2} + a_1\}\\
&\text{NIMP}: \overline{a_1 \rightarrow a_4} \equiv \{a_1' = a_1.\overline{a_4} = (\overline{a_2} + a_1).(a_2 + \overline{a_1})\}\\
&\text{TRUE}: a_3 = 1\\
&\text{NIMP}: \overline{a_3 \rightarrow a_1} \equiv \{a_3' = \overline{a_1} = a_2.\overline{a_1} + \overline{a_2}.a_1\}\\
&\qquad\qquad \equiv \{a_3 \leftarrow a_1 \text{ XOR } a_2\} \qquad (1)
\end{aligned}
$$

NIMP (negated IMP) is the basic logic operation of the implication logic. According to (1), when multiple non-volatile logic fan-out is required, a set of TRUE and NIMP operations (performing NOT and COPY functions) allows to copy the information in the array without the need for intermediate sensing/writing operations.

Reliability-based design of an XOR function in the reprogrammable architecture requires the following steps [15]:

$$
\begin{aligned}
&\text{Preset}: b_1 = 1\\
&\text{AND}: b_1 \leftarrow a_1.a_2\\
&\text{Preset}: a_3 = 0, b_2 = 1\\
&\text{NAND}: a_3 \leftarrow \overline{b_1.b_2} \equiv \overline{b_1}\\
&\text{Preset}: b_1 = 0\\
&\text{NAND}: b_1 \leftarrow \overline{a_1.a_3},\\
&\text{Preset}: b_2 = 0\\
&\text{NAND}: b_2 \leftarrow \overline{a_2.a_3},\\
&\text{Preset}: a_3 = 0\\
&\text{NAND}: a_3 \leftarrow \overline{b_1.b_2} \equiv a_1 \text{ XOR } a_2, \qquad (2)
\end{aligned}
$$

where AND and NAND are the most reliable basic logic operations of the reprogrammable architecture [15].

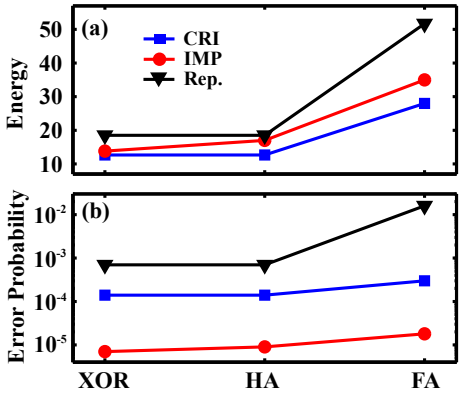

Fig. 3. (a) Energy consumption for complex logic functions. The y-axis is normalized by the amount of energy required for high-to-low MTJ resistance switching. (b) $\overline{E}_{\rm f}$ for different MRAM-based implementations of functions XOR, half adder (HA), and full adder (FA).

III. Performance Improvement and Results

For the performance analysis, we follow the 1T/1MTJ model presented in our previous work [13] and the STT-MTJ Spice model [16] for the energy calculations. The average error probability of a complex Boolean logic function (f), which is implemented as a sequence of the basic implication or reprogrammable logic operations, is defined as [15]

$$\overline{E}_{\rm f} = 1 - R(f) = 1 - \prod_{i=1}^{n_{\rm f}} [1 - \overline{E}_{\rm b}(i)], \qquad (3)$$

where $R(f)$ shows the reliability of f, $n_{\rm f}$ is the total number of required basic logic operations for implementing f, and $\overline{E}_{\rm b}(i)$ denotes to the average error probability of the $i-$th basic logic operation. For example, implication or reprogrammable-based XOR function includes seven NIMP ($n_{\rm f} = 7$) or one AND and four NAND ($n_{\rm f} = 5$) basic functions.

Fig. 3 shows the energy consumptions and the error probabilities ($\overline{E}_{\rm f}$) for MRAM-based XOR, half adder, and full adder logic functions. It demonstrates that the implication-based implementation of complex functions exhibits a better performance with respect to power consumption (Fig. 3a). It also provides about two orders of magnitude higher reliability than the reprogrammable architecture (Fig. 3b). Therefore, the implication logic outperforms the reprogrammable logic and is expected to be the implementation of choice for MRAM-based stateful logic circuits. However, in the following we discuss more design tradeoffs by combining these logic architectures.

A. Combining Reprogrammable and Implication Logic

Due to computational completeness, implication and reprogrammable MRAM-based stateful logic architectures can be used independently to design any Boolean logic function. However, their combination can minimize the number of required logic steps as it provides more degrees of freedom by utilizing more fundamental logic operations of several gates. Therefore, it reduces the execution time and the energy consumption of complex logic functions. For example, in the combined reprogrammable-implication (CRI) architecture the XOR function can be designed as:

$$\begin{aligned}
&\text{TRUE}: a_3, a_4 = 1\\
&\text{NIMP}: \overline{a_3 \rightarrow a_1} \equiv \{a_3' = a_3.\overline{a_1} = \overline{a_1}\}\\
&\text{NIMP}: \overline{a_4 \rightarrow a_2} \equiv \{a_4' = a_4.\overline{a_2} = \overline{a_2}\}\\
&\text{Preset}: b_1 = 0\\
&\text{NAND}: b_1 \leftarrow \overline{a_1.a_4}\\
&\text{Preset}: b_2 = 0\\
&\text{NAND}: b_2 \leftarrow \overline{a_2.a_3}\\
&\text{Preset}: a_5 = 1\\
&\text{AND}: a_5 \leftarrow \overline{b_1.b_2} \equiv a_1 \text{ XOR } a_2, \qquad (4)
\end{aligned}$$

According to Fig. 3a, the CRI design provides a lower energy consumption compared to both implication and reprogrammable designs. However, its error probabilities is higher that the implication design since it employs basic reprogrammable operations.

B. Parallel MRAM-Based Computation

Parallelization of several MRAM arrays can be used to perform simultaneous operations on the same WLs to decrease the number of required serial steps. For example, in the implication architecture by applying the selecting and pre-selecting voltage signals to two WLs in Fig.4, the corresponding MTJs are selected and pre-selected in all arrays. Therefore, by applying corresponding current signals ($I_{\rm imp}$) simultaneously to all current-carrying lines, implication operations are simultaneously performed in all arrays. Similarly, reprogrammable operations can be parallelized when the two group of coupled arrays are connected in series.

For logic functions in which intermediate results have to be used as input of next logic steps (e.g. a two-bit full adder shown in Fig.5) only some parts of the computations can be performed in parallel. As the carry output from the first full adder ($c_{\rm out}$) is required as an input for the second full adder ($c_{\rm in}'$) it is not possible to parallelize all computations.

We assume that a_1 (a_2) and a_1' (a_2') are stored in the first (second) MTJs in the MRAM arrays 1 and $1'$ which have coupled WLs. The AND and the XOR functions between

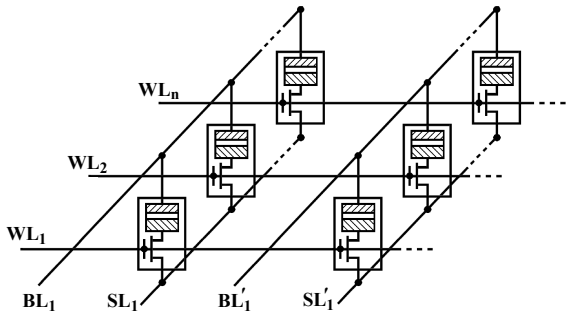

Fig. 4. Coupled MRAM arrays suited for parallel MRAM-based stateful computations.

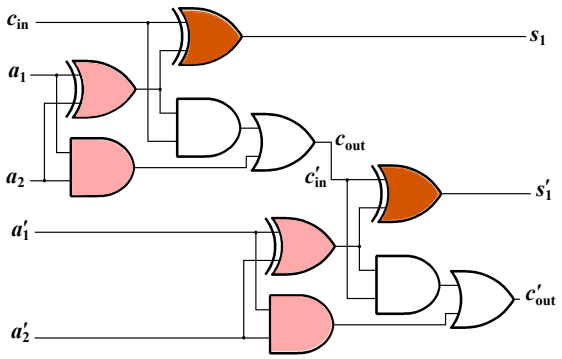

Fig. 5. Logic diagram of a two-bit full adder.

(a_1, a_2) and (b_1, b_2) can be performed in parallel as explained before. Afterwards, $c_{\rm out}$ $(c'_{\rm in})$ is calculated without parallelization. This part of the calculations is performed in Array 1. Then by using a read/write operation the result is written into the MTJ in Array $1'$ which is in the same WL as the MTJ that holds $c_{\rm in}$ in Array 1. After that, the XOR functions are performed in parallel to calculate s_1 and s_2 and finally the calculations are continued to compute $c'_{\rm out}$ in Array $1'$. As a result, by parallelization of the MRAM-based logic arrays, the total calculation time required for the implication-based and CRI-based implementations of a two-bit full adder are decreased by about 40% and 50% (Fig.6).

IV. Conclusion

We have described the possible tradeoffs to optimze the execution time, energy consumption, and the reliability of the MRAM-based stateful logic architecture. It has been shown that a combined reprogrammable-implication logic architecture minimizes the total number of the required logic steps and thus the energy consumptions. However, it decreases the reliability of the MRAM-based computations. It is demonstrated that the parallelization of MRAM-based computations can significantly reduce the execution time. Since MRAM-based computing systems merge logic and memory, the necessary communication between separate units is largely decreased. It also features a simple circuit structure and delocalizes computational execution, which opens the door for innovation in computational paradigms.

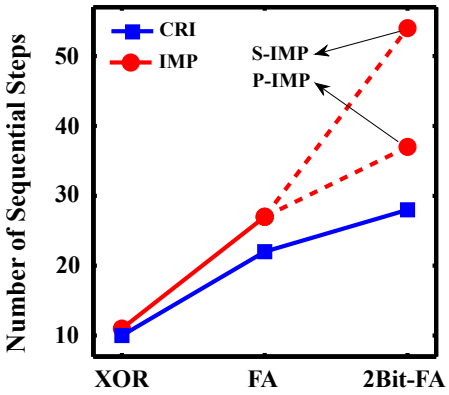

Fig. 6. Required sequential steps for serial (S-IMP) and parallel (P-IMP) MRAM-based implication and combined reprogrammable-implication (CRI) architectures.

Acknowledgment

The work is supported by the European Research Council through the grant #247056 MOSILSPIN.

References

[1] V. V. Zhirnov, R. K. Cavin, J. A. Hutchby, and G. I. Bourianoff, "Limits to Binary Logic Switch Scaling - a Gedanken Model," *Proc. IEEE*, vol. 91, pp. 1934–1939, 2003.

[2] S. Matsunaga, J. Hayakawa, S. Ikeda, K. Miura, T. Endoh, H. Ohno, and T. Hanyu, "MTJ-Based Nonvolatile Logic-in-Memory Circuit, Future Prospects and Issues," *Proc. Des. Autom. Test Eur. Conf. (DATE)*, pp. 433–435, 2009.

[3] B. N. Engel, J. Akerman, B. Butcher, R. W. Dave, M. DeHerrera, M. Durlam, G. Grynkewich, J. Janesky, S. V. Pietambaram, N. D. Rizzo, J. M. Slaughter, K. Smith, J. J. Sun, and S. Tehrani, "A 4-Mb Toggle MRAM Based on a Novel Bit and Switching Method," *IEEE Trans. Magn.*, vol. 41, no. 1, pp. 132–136, 2005.

[4] C. Chappert, A. Fert, and F. N. V. Dau, "The Emergence of Spin Electronics in Data Storage," *Nat. Mater.*, vol. 6, pp. 813–823, 2007.

[5] C. Augustine, N. Mojumder, X. Fong, H. Choday, S. P. Park, and K. Roy, "STT-MRAMs for Future Universal Memories: Perspective and Prospective," *Proceedings of the 28th International Conference on Microelectronics (MIEL)*, pp. 349–355, 2012.

[6] W. Zhao, E. Belhaire, C. Chappert, F. Jacquet, and P. Mazoyer, "New Non-Volatile Logic Based on Spin-MTJ," *Phys. Status Solidi (a)*, vol. 205, pp. 1373–1377, 2008.

[7] M. Natsui, D. Suzuki, N. Sakimura, R. Nebashi, Y. Tsuji, A. Morioka, T. Sugibayashi, S. Miura, H. Honjo, K. Kinoshita *et al.*, "Nonvolatile Logic-in-Memory Array Processor in 90nm MTJ/MOS Achieving 75% Leakage Reduction using Cycle-Based Power Gating," *Proceedings of the International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pp. 194–195, 2013.

[8] M. Hosomi, H. Yamagishi, T. Yamamoto, K. Bessho, Y. Higo, K. Yamane, H. Yamada, M. Shoji, H. Hachinoa, C. Fukumoto, H. Nagao, and H. Kano, "A Novel Nonvolatile Memory with Spin Torque Transfer Magnetization Switching: Spin-RAM," *IEDM Tech. Dig.*, pp. 459–462, 2005.

[9] 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, no. 7290, pp. 873–876, 2010.

[10] H. Mahmoudi, T. Windbacher, V. Sverdlov, and S. Selberherr, "Implication Logic Gates using Spin-Transfer-Torque-Operated Magnetic Tunnel Junctions for Intrinsic Logic-in-Memory," *Solid-State Electron.*, vol. 84, pp. 191–197, 2013.

[11] A. Lyle, S. Patil, J. Harms, B. Glass, X. Yao, D. Lilja, and J. P. Wang, "Magnetic Tunnel Junction Logic Architecture for Realization of Simultaneous Computation and Communication," *IEEE Trans. Magn.*, vol. 47, pp. 2970–2973, 2011.

[12] H. Mahmoudi, T. Windbacher, V. Sverdlov, and S. Selberherr, "MRAM-based Logic Array for Large-Scale Non-Volatile Logic-in-Memory Applications," *Proceedings of the 2013 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, pp. 26–27, 2013.

[13] H. Mahmoudi, T. Windbacher, V. Sverdlov, and S. Selberherr, "Performance Analysis and Comparison of Two 1T/1MTJ-based Logic Gates," *Proceedings of the 18th International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*, pp. 163–166, 2013.

[14] H. Mahmoudi, T. Windbacher, V. Sverdlov, and S. Selberherr, "STT-MRAM-Based Reprogrammable Logic Gates for Large-Scale Non-Volatile Logic Integration," *Proceedings of the International Conference on Nanoscale Magnetism (ICNM)*, p. 208, 2013.

[15] H. Mahmoudi, T. Windbacher, V. Sverdlov, and S. Selberherr, "Reliability Analysis and Comparison of Implication and Reprogrammable Logic Gates in Magnetic Tunnel Junction Logic Circuits," *IEEE Trans. Magn.*, vol. 49, pp. 5620–5628, 2013.

[16] J. D. Harms, F. Ebrahimi, X. F. Yao, and J. P. Wang, "SPICE Macromodel of Spin-Torque-Transfer-Operated Magnetic Tunnel Junctions," *IEEE Trans. Electron Devices*, vol. 57, no. 6, pp. 1425–1430, 2010.