Non-Volatile Processor Based on MRAM for Ultra-Low-Power IoT Devices

Sophiane Senni, Lionel Torres, Gilles Sassatelli, Abdoulaye Gamatié

Non-Volatile Processor Based on MRAM for Ultra-Low-Power IoT Devices

SOPHIANE SENNI, LIONEL TORRES, GILLES SASSATELLI,
and ABDOULAYE GAMATIE, LIRMM, UMR CNRS 5506, University of Montpellier

Over the past few years, a new era of smart connected devices has emerged in the market to enable the future world of the Internet of Things (IoT). A key requirement for IoT applications is the power consumption to allow very high autonomy in the case of battery-powered systems. Depending on the application, such devices will be most of the time in a low-power mode (sleep mode) and will wake up only when there is a task to accomplish (active mode). Emerging non-volatile memory technologies are seen as a very attractive solution to design ultra-low-power systems. Among these technologies, magnetic random access memory is a promising candidate, as it combines non-volatility, high density, reasonable latency, and low leakage. Integration of non-volatility as a new feature of memories has the great potential to allow full data retention after a complete shutdown with a fast wake-up time. This article explores the benefits of having a non-volatile processor to enable ultra-low-power IoT devices.

CCS Concepts: • **Hardware → Emerging technologies**; **Spintronics and magnetic technologies**; *Emerging architectures;*

Additional Key Words and Phrases: MRAM, non-volatility, internet of things, embedded systems

ACM Reference Format:
Sophiane Senni, Lionel Torres, Gilles Sassatelli, and Abdoulaye Gamatie. 2016. Non-volatile processor based on MRAM for ultra-low-power IoT devices. J. Emerg. Technol. Comput. Syst. 13, 2, Article 17 (November 2016), 23 pages.
DOI: http://dx.doi.org/10.1145/3001936

1. INTRODUCTION

The interest of promoting smart systems thanks to interconnected objects is growing fast to build up what is known as the Internet of Things (IoT). It is assumed that 50 billion objects will be connected in 2020 [Karnouskos et al. 2014]. IoT devices essentially do three actions: sense, process, and send. First, the object captures information from the external environment. It can be, for instance, movement, temperature, sound, location (via Global Positioning System (GPS)), or heart rate. Second, the device stores the measured data and does some minor computation. Finally, it sends data to a data center wirelessly. As a result, IoT objects have to comprise three main components: sensor, processing unit, and communication unit.

These kinds of devices are typically battery powered. Moreover, they have to be operational for a very long duration (several months or even years). Therefore, the

This work was supported by the European Union's Horizon 2020 research and innovation programme under Grant No. 687973, GREAT (heteroGeneous integRated magnetic tEchnology using multifunctional standardized sTack (MSS)), and the French National Research Agency under Grant No. ANR-15-CE24-0033-01 (MASTA project).

Authors' addresses: S. Senni, L. Torres, G. Sassatelli, and A. Gamatie, Microelectronics Department, Montpellier Laboratory of Informatics, Robotics and Microelectronics; emails: {senni, torres, sassatelli, gamatie}@lirmm.fr.

DOI: http://dx.doi.org/10.1145/3001936

energy consumption is definitely a critical constraint. Generally, an IoT object is most of the time inactive, staying in low-power mode (sleep mode) and waiting for the next work to achieve. This can be periodic or dependent on external events that will wake up the device. As a result, sleep mode power will consume the largest amount of energy and battery life. Today, several microcontrollers (MCUs) targeting low-power applications are available in the market. Commercial MCUs actually implement not only one but several power-down modes with different wake-up times, depending on from which low-power mode the MCU returns to the active mode. Thus, depending on the applications, system designers have to make a tradeoff considering the power consumption in both active and sleep modes, the wake-up time, and the ratio of time spent in active/sleep modes to ensure energy efficiency.

Within this context of energy efficiency, emerging non-volatile memory (NVM) technologies have raised a new interesting computing paradigm known as *normally-off computing* [Ando et al. 2014]. This is the ability for systems to be normally powered off (there is no operation to do) and momentarily powered on (there is a need to operate). Magnetic random access memory (MRAM) is seen by international technology roadmap for semiconductors (ITRS) [ITRS 2013] as one of the most promising NVM, as it shows excellent scalability and low access time compared to existing NVM such as Flash memory and other emerging NVM technologies such as phase-change random access memory (PCRAM) and resistive random access memory (or Oxyde-based RAM (OxRAM)). Currently, working memories of a processor, such as registers and static random access memory (SRAM), use volatile complementary metal oxide semiconductor (CMOS) transistors to retain information. If the system is powered down, then these kinds of memories lose data and a fairly long time (up to a few milliseconds [STM32L1 2016]) is necessary to restore information after a new power-up. Integrating the non-volatility feature will allow systems to keep data available, even after a power-down, thus significantly reducing the wake-up time. For battery-powered devices such as IoT objects, it will give the possibility to go into sleep mode more frequently, with zero leakage power since no power is required to retain the state of the system.

This article explores the opportunity of having a non-volatile processor by the integration of MRAM at register and main memory levels. Two capabilities introduced by the non-volatility are investigated. First, the *instant on/off*, which allows a recovery of the state of the processor after a power-down. Second, the *rollback* giving the possibility to restore a previous valid state of the processor, for instance, in the case of an execution error (soft errors) or a power failure. The main objective is to present the *rollback* principle and not to describe how those errors are detected. The cost in terms of performance and energy to save/restore the state of the processor is estimated. Considering the open-source Amber core [Santifort 2010], a 32-bit RISC[1] embedded processor, the main contributions of this article are summarized as follows:

(1) With two applications running on the processor, the following abilities were validated:
  - Save/restore the complete state of the processor to demonstrate the *instant-on/off* ability.
  - Restore a previous valid state of the processor (*rollback*) at runtime.

(2) Performance and energy of the save/restore procedures were estimated for several NVMs, including MRAM. In addition, to take advantage of the *instant-on/off*, the condition on the minimum time a non-volatile processor has to stay in sleep mode has been analyzed.

The rest of the article is organized as follows: Section 2 provides basic information on MRAM and then details the two MRAM technologies considered in this work. Section 3

[1] Reduced Instruction Set Computer.

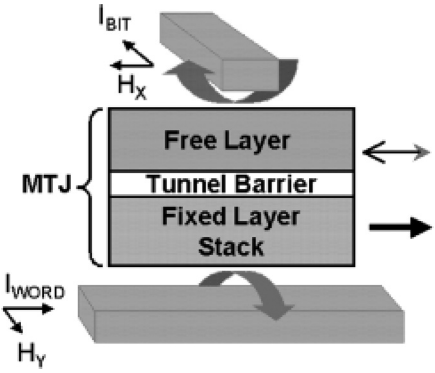


Fig. 1. Conventional MRAM [Andre et al. 2005].

exhibits an overview of the interests of MRAM for IoT devices according to the study of this article and the current state-of-the-art. Section 4 first describes both *instant on/off* and *rollback* capabilities and then shows simulation results of a complete backup/recovery of the state of the Amber core. Section 5 gives performance and energy estimations of the backup/recovery phases, and it discusses the performance and energy implications of integrating MRAM into a processor for both active and sleep modes. Section 6 reviews related work on non-volatile logic circuits. Section 7 concludes this article and shows the perspectives.

2. MRAM: BASICS AND TECHNOLOGIES

2.1. Basics

A MRAM bit is a magnetic tunnel junction (MTJ) consisting of two ferromagnetic layers separated by a thin insulating barrier. The information is stored as the magnetic orientation of one of the two layers, called the free layer (FL). The other layer, called the reference layer, provides the fixed reference magnetic orientation required for reading and writing. The tunnel magnetoresistance effect [Moodera et al. 1995] causes MTJ resistance to depend significantly on the relative orientation of the two magnetic layers: The antiparallel state provides much larger resistance than the parallel state. It enables the magnetic state of the FL to be sensed thanks to a current flowing through the MTJ. Hence, stored information can be read. Five methods have been proposed to switch the orientation of the FL: toggle [Engel et al. 2005], thermally assisted switching (TAS) [Prejbeanu et al. 2007], spin transfer torque (STT) [Khvalkovskiy et al. 2013], and voltage-induced switching, and the most recent method is called spin orbit torque (SOT) [Gambardella and Miron 2011]. Toggle-MRAM has a very high switching energy, and its scalability is limited. Hence, it is not considered for the remaining of this article. Although MeRAM (i.e., voltage-induced switching) and SOT-MRAM show very promising performance, they are always at experimental level and need further development. On the contrary, TAS-MRAM and STT-MRAM are quite mature since test chips already exist [Noguchi et al. 2013; Ikegami et al. 2014; Noguchi et al. 2015; Crocus 2016]. Therefore, only these two technologies are considered for the rest of the article.

A conventional MRAM, shown in Figure 1, uses a simple way to program the MTJ where a sufficient magnetic field is generated thanks to a combination of two current flows applied simultaneously through a row and a column of an MTJ array. Two problems arose with this method. First, large current is needed to generate sufficient magnetic field to reverse the magnetization of the FL. Second, this approach suffers

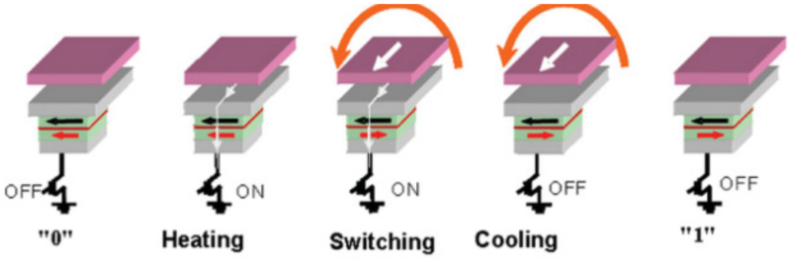


Fig. 2. Thermally assisted switching MRAM [Prejbeanu et al. 2013].

from selectivity problem: Some of the bits sharing the same row or column of the cell being programmed might be exposed to sufficient magnetic field and be switched unintentionally. This effect is one consequence of process variability. The magnetic field necessary to reverse the magnetization is not exactly the same for all the bits [Lewotsky 2013].

2.2. Thermally Assisted Switching

The aim of the TAS concept was to improve the downsize scalability of MRAM. The concept was developed by the Spintec laboratory, and TAS-MRAM is currently commercialized by Crocus Technology [Crocus 2016]. TAS-based MTJ uses an anti-ferromagnetic layer to block the magnetic orientation of the FL under a threshold temperature. To switch the bit cell, a select transistor provides a flow of current to heat the MTJ above the blocking temperature, thereby enabling storage of new information thanks to application of a magnetic field. Heating the FL allows TAS-MRAM to use a smaller magnetic field and hence less current than toggle MRAM to write the bit cell, since a single conductive line is sufficient to generate the required magnetic field. Blocking the FLs state using a coupling anti-ferromagnetic layer also significantly improves data stability, even scaling the technology node. As a result, TAS-MRAM makes it possible to reduce the switching energy while ensuring excellent data retention. This new method also solves the selectivity issue, since the MTJ has to be heated before writing.

Figure 2 shows a complete TAS write operation. Assuming the MTJ stores a 0 state (parallel state), the first step in the TAS method is to heat the FL by flowing a current through the MTJ to reach the blocking temperature (heating step in Figure 2). The second step is to generate an external magnetic field to switch the FL while heating the MTJ (switching step in Figure 2). Once the FL switches to the 1 state, the CMOS transistor responsible for the heating process is switched off whereas the MTJ remains under the external magnetic field (cooling step in Figure 2).

2.3. Spin Transfer Torque

STT-MRAM appeared with the need to reduce the switching energy consumption of MRAMs. Unlike the previous MRAMs, which use an external magnetic field to program a bit cell, STT-MRAM write operations are based on another physical phenomenon to switch the magnetic orientation of the FL called STT. The idea is that the FL can be switched by direct transfer of the spin angular momentum from spin-polarized electrons. In this way, a highly spin-polarized current flowing through the MTJ causes a torque applied by the injected electron spins on the magnetization of the FL. Applying sufficient current will cause sufficient torque to switch the bit cell, thereby enabling information to be written. Figure 3 depicts the STT effect.

Figure 3(a) shows the transition from an antiparallel to a parallel state. In this case, electrons go through the fixed layer first, and the fixed layer acts as a polarizer. Thus,

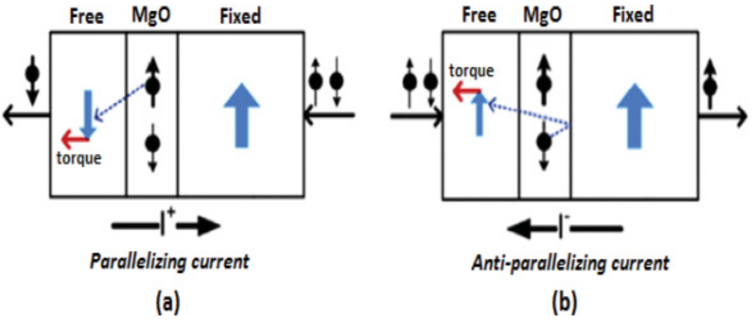


Fig. 3. Spin transfer torque effect: (a) illustration of the transition from an antiparallel to a parallel state, and (b) the transition from a parallel to an antiparallel state [Khvalkovskiy et al. 2013].

electrons are spin polarized in the magnetic orientation of the fixed layer. Once the insulating barrier (MgO) is crossed, the spin-polarized electrons exert torque on the magnetization of the FL until a magnetic orientation reversal occurs. A similar effect is depicted in Figure 3(b) for the transition from a parallel to an antiparallel state. In this case, electrons go through the FL first. While the majority of the electrons will be spin polarized in the magnetic orientation of the FL, a minority of electrons will still be spin polarized in the opposite direction of the FL. These minority electrons will be reflected at the barrier interface and will exert torque on the magnetization of the FL.

Two kinds of magnetization of the magnetic layers can be found in STT-MRAM: in-plane and perpendicular. In-plane magnetization is also used in toggle-MRAM and TAS-MRAM, in which the magnetic orientation is parallel to the plan of the MTJ, whereas in perpendicular magnetization, the magnetic orientation is perpendicular to the plan of the MTJ. Perpendicular STT-MRAM was introduced to further reduce the switching current of the MTJ and to improve scalability.

3. INTEREST OF MRAM FOR MCU AND IOT DEVICES: OVERVIEW

Based on the study made in this article and the state-of-the-art of MRAM research, this section aims at giving a clear overview of the key benefits of MRAM for MCUs and globally for future IoT devices. The discussion will focus on the main challenges considered in any system-on-chip (SoC) designs: high performance, low power, and small physical footprint.

3.1. High Performance

In embedded systems, high performance capability is desired for two main reasons:

(1) Meet the timing constraints to execute tasks of a given application.
(2) Finish the work faster to spend less time in active mode (i.e., remain most of the time in low power sleep mode to minimize energy consumption).

Compared to other NVM technologies, MRAM demonstrates competitive access time to replace SRAM at cache and memory levels. Even if the current state-of-the-art still shows better performance for SRAM, MRAM access time is sufficiently low to be used in low-power IoT devices that do not target very high frequencies. For instance, ultra-low-power MCUs proposed by STMicroelectronics and NXP do not exceed a 100MHz operating frequency. On the other hand, Jan et al. [2014] demonstrated a fully functional 8Mb perpendicular STT-MRAM chip with sub-5ns writing operations.

Furthermore, MRAM is able to integrate the logic part of the MCU by using hybrid CMOS/MTJ flip-flops (FFs). As noted in this article, the CMOS part of the FFs allows us to meet the high performance requirement of the applications while the magnetic

Table I. Performance Comparison between Flash and MRAM

Technology	NOR Flash	NAND Flash	TAS-MRAM	STT-MRAM
Read time	<80ns	20μs	<35ns	<10ns
Program time	10μs/byte	200μs/byte (0.4μs/byte in page mode)	<35ns	∼10ns
Erase time (Block size = 128kB)	1s/block	1ms/block	—	—
Read/Write voltage	∼5V / ∼10V	∼5V / >10V	—	<1V / ∼1V

part makes the device completely non-volatile with interesting capabilities such as *instant-on/off* and *rollback*.

3.2. Low Power

In addition to its good performance, MRAM can clearly help IoT devices to minimize their power consumption and extend the battery lifetime. This is possible thanks to:

(1) The ultra-low leakage of MRAM. The leakage power is only consumed by the peripheral circuitry used to read/write the MTJ elements.
(2) The non-volatility of MRAM, which gives the possibility to further reduce the power consumption of the device during sleep mode.

Basically, the total power consumption of an IoT device can be defined with Equation (1), where P_{active}, P_{sleep}, and P_{wakeup} are respectively the power consumption during active mode, sleep mode, and wake-up transition:

$$P_{total} = P_{active} + P_{sleep} + P_{wakeup}. \tag{1}$$

The weight of each element in the total power consumption strongly depends on the applications. For some applications, the system will spend most of its time in sleep mode. Therefore, a low sleep-power consumption is more critical. On the other hand, for applications such as data loggers, the device will often switch between active and sleep modes. In that case, the wake-up energy has to be reduced.

MRAM is definitely advantageous for the first kind of application. While current MCUs need to preserve memory contents and the state of the registers to resume the execution state, using MRAM allows us to completely power off the digital logic (including the memory and registers) without losing the state of the system. Hence, for applications spending a very long time in sleep mode, the energy savings could be significant.

Regarding the second kind of application, a more detailed analysis is required to evaluate the benefits of MRAM. The main consideration that has to be taken into account is the active/sleep mode duty cycle. As will be reported in this work, a backup penalty has to be considered when using MRAM. Therefore, analyzing the active/sleep mode period with a study similar to that in Section 5.2.4 is necessary. The evaluation in Section 5.2.4 showed encouraging results. However, to strengthen the high potential of MRAM for future devices, another analysis has to consider real data measurements from MCU products, IoT applications, and a real non-volatile processor design implementing the *instant-on/off*, which is part of the future work.

Additionally, data logging applications that need to record data measurements (e.g., from a sensor) in NVM could really benefit from using MRAM instead of Flash memory. Indeed, Flash memory is power hungry and time-consuming, as will be pointed out in Sections 5.1 and 5.2. For comparison purposes, Table I provides data on the performance of both Flash and MRAM technologies [Micheloni et al. 2010; Crippa et al. 2008; Dirik 2009; Meena et al. 2014; Nowak et al. 2016; Crocus 2016]. Moreover, this article shows that MRAM is suitable for backing-up/restoring the execution state of the system, whereas this is not practical with Flash due to its low performance and high programming energy. As a result, a full RAM and register retention is required for Flash systems, resulting in higher static power consumption during sleep mode. In addition, Flash memory suffers from a quite poor endurance (typically from 10^4 to 10^6 erase cycles [Meena et al. 2014]), resulting in a small lifetime, while MRAM demonstrates a very high endurance (about 10^{15} cycles [Huai et al. 2015]), making this technology more appropriate for applications that frequently have to record data in NVM.

3.3. Small Physical Footprint

Compared to existing memory technologies, MRAM offers a pretty high density. Its bit cell structure, consisting of one CMOS transistor and one MTJ, makes this technology denser than SRAM and as dense as DRAM. The current state-of-the-art shows that the MRAM drawback is the high read/write circuitry area. This is due to the large CMOS transistors required to generate sufficient write current. As a result, the current designs of hybrid CMOS/MTJ FFs are bigger than standard CMOS FFs by a ratio of 1.5 to 3 [Chabi et al. 2014]. However, recent experiments from the industry have demonstrated successful write operations of STT-MRAM with only a 7.5μA current [Nowak et al. 2016]. This low switching current allows us to further reduce the peripheral circuitry area thanks to smaller CMOS transistors. This definitely shows the potential of MRAM to further improve the energy and area efficiency of this NVM technology.

Another advantage of MRAM is its "above Integrated Circuit (IC)" integration process, which allows the MTJ to be stacked above the CMOS part of a digital circuit. A recent experiment on this topic evaluated the chip area reduction for a 2Mbit STT-MRAM test chip [Koike et al. 2016]. This capability allows new kind of architectures, known as logic-in-memory [Das et al. 2014; Pala et al. 2015], where memory is very close to the logic to increase both performance and area efficiency.

4. INSTANT ON/OFF AND ROLLBACK FEATURES

After giving an overview of the Amber-embedded processor core, this section will focus on the description of both *instant on/off* and *rollback* concepts. Then validation of a complete backup/recovery of the state of the Amber core via register-transfer level (RTL) simulation will be shown. Simulations have been done with two applications: *dhrystone 2.1* [Weicker 1984], which is included with the source code of the Amber core, and *blowfish*, a cipher algorithm proposed from the MiBench benchmark suite [Guthaus et al. 2001]. The registers of the processor are duplicated to emulate the non-volatile registers and save the state of the system. Then, control logic is added to enable the *rollback* capability.

4.1. Amber Core

The Amber core used in this work is an ARM-compatible 32-bit RISC processor fully compatible with the old ARMv2a instruction set architecture. There are two versions of the Amber core. The first is Amber 23, which has a three-stage pipeline, a unified instruction and data cache, and a 32-bit wishbone memory bus interface. The second is Amber 25 with a five-stage pipeline, separate instruction and data caches, and a 128-bit wishbone memory interface. In this work, Amber 23 is used, as it has the simplest architecture. Figure 4 depicts the architecture of the considered processor.

4.2. Instant On/off

The *instant on/off* function consists in saving a complete state of the processor before a power-down and then restoring the state after a new power-up. The state of a processor is contained in both registers and main memory. At the least, it is required to include the non-volatility into these two memory components to maintain the system state after

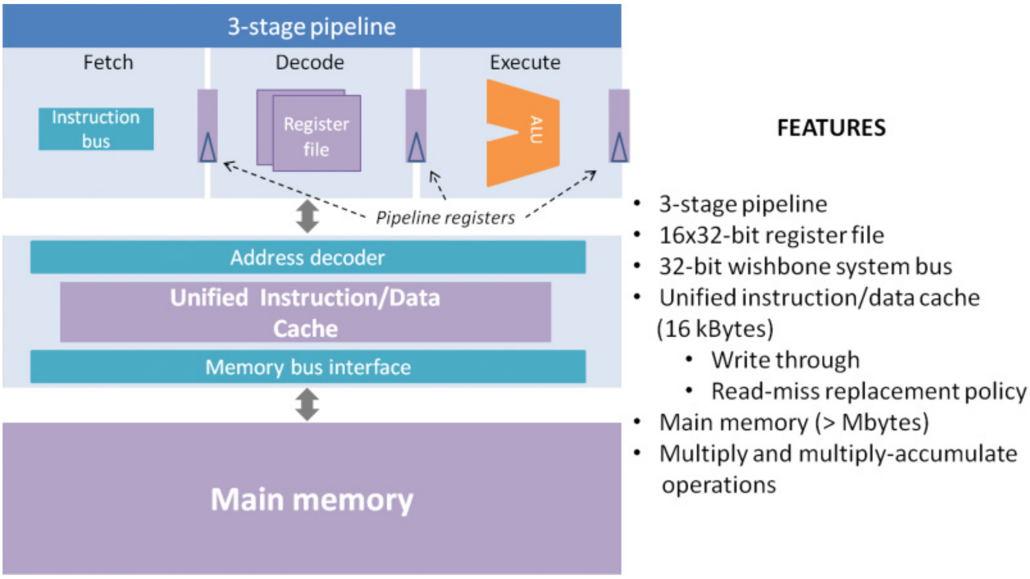

3-stage pipeline
Fetch
Decode
Execute
Instruction bus
Register file
ALU
Pipeline registers
Address decoder
Unified Instruction/Data Cache
Memory bus interface
Main memory
FEATURES
• 3-stage pipeline
• 16x32-bit register file
• 32-bit wishbone system bus
• Unified instruction/data cache (16 kBytes)
• Write through
• Read-miss replacement policy
• Main memory (> Mbytes)
• Multiply and multiply-accumulate operations

Fig. 4. Amber 23 core architecture.

a power-down. In our study, 1,644 FFs contain the state of the Amber processor core, including the register file, the pipeline registers, and some other internal registers.

The architecture considered in this study embeds a small cache memory. The latter does not need to be necessarily non-volatile to perform a backup/restore of the system state. A system without a cache will not affect the *instant-on/off* principle presented in this work. However, performance penalties have to be considered if a volatile cache is embedded. If the writing policy is a write-through,[2] then the backup performance is unchanged. Regarding the restore mode, a warm-up period has to be taken into account after a power-up to restore data from main memory to cache memory. If the writing policy is a write-back,[3] then all cache blocks marked as "dirty" have to be written back to the main memory before a power-down to preserve the state of the processor. Thus, in this case, both backup and restore performances are degraded.

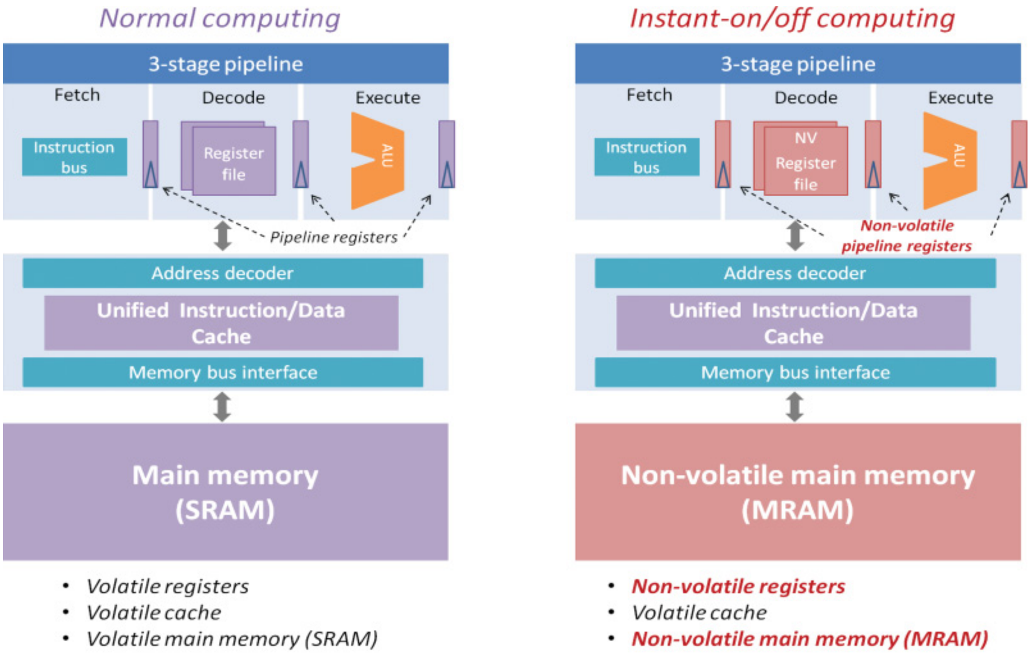
Figure 5 compares the original architecture of the Amber processor and the required architecture for a non-volatile processor with *instant-on/off* capability.

Figure 6(a) shows a non-volatile FF architecture based on MRAM, which can be typically used to design non-volatile registers. It consists of a standard CMOS FF for the volatile part and a MTJ for the non-volatile part. By means of a multiplexor, the input state of the CMOS FF is either the output state of the previous stage of the circuit (ff_d in the figure) or the state of the MTJ (MQ in the figure). Also, a write circuit allows us to store the state of the CMOS FF into the non-volatile MTJ.

Figure 6(b) depicts the timing diagram of the non-volatile FF. When the write signal ff_mw is activated for sufficient time (i.e., write latency), volatile data (ff_d) are stored

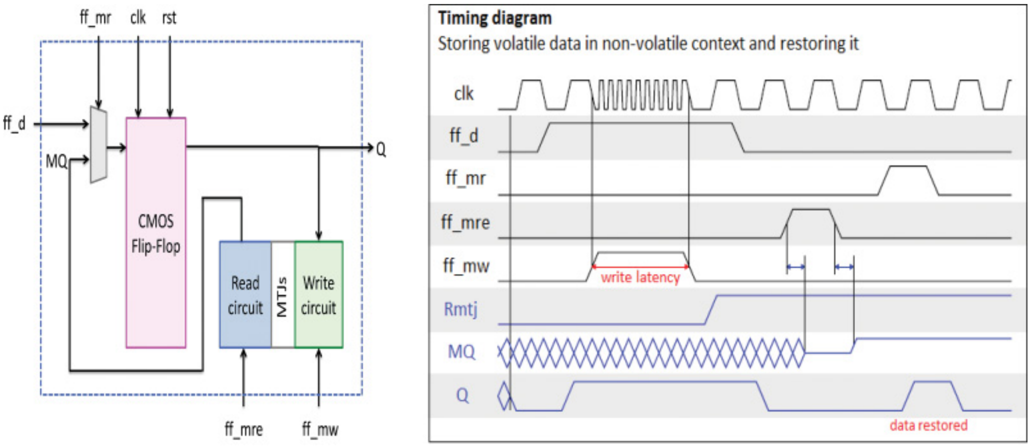
[2] A scheme in which writes always update both cache and main memory, ensuring that data are always consistent between the two.

[3] A scheme that handles writes by updating values only to the block in the cache and then writing the modified block to the lower level of the hierarchy when the block is replaced.



(a) (b)

Fig. 5. Amber architecture with instant-on/off computing: (a) original Amber architecture and (b) Amber architecture with non-volatile MRAM.



(a) (b)

Fig. 6. MRAM-based non-volatile flip-flop: (a) architecture and (b) timing diagram.

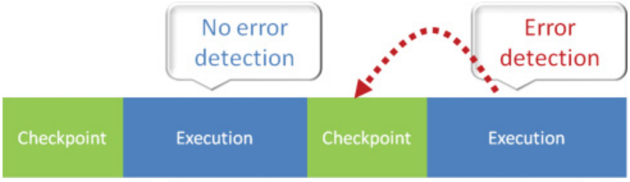


Fig. 7. Rollback principle.

in a non-volatile context (Rmtj). Restoring this non-volatile context is performed by activating the signal ff_mre and then the signal ff_mr.

Assuming the processor uses this kind of non-volatile FF and the main memory is non-volatile, then the *instant on/off* procedure is as described in Algorithm 1.

```
ALGORITHM 1: Instant-on/off Procedure
(1) For each FF, save the current state by writing the value from the CMOS FF into the MTJ;
(2) Power down the processor. As the main memory is non-volatile, data are preserved;
(3) Power up the processor. As the main memory is non-volatile, data are available;
(4) For each FF, restore the backup data by reading the value from the MTJ into the CMOS FF.
```

4.3. Rollback

The *rollback* is the ability to return to a previously valid state of the processor in the case, for instance, of an execution error or a power failure. We assume that an error detection mechanism is available in the processor architecture to identify errors during execution as proposed, for instance, in Yu et al. [2008] or Wali et al. [2016]. The principle of the *rollback* is shown in Figure 7.

Checkpoints can be created to save the state of the system either periodically or at strategic instants during the execution of the application. Then, if a system failure occurs, there is the possibility to come back to the last *checkpoint*. A *checkpoint* consists of a backup of both registers and main memory. Indeed, after each *checkpoint*, the main memory contents will most probably be modified. Therefore, to avoid inconsistency in memory, it is necessary to add an additional memory (called *checkpoint memory* in the rest of the article) to keep a backup of the memory contents. To make it easier, the main memory is duplicated. One memory will be used for the normal execution, whereas the other one will be used to store the *checkpoint*. In a real application, the *checkpoint memory* size is smaller than the main memory. This size depends on both the application and the *checkpointing* period. An alternative solution to perform a *checkpoint* at memory level is the use of a double context non-volatile SRAM cell as proposed in Jovanovic et al. [2015]. Hence, it is possible to optimize the silicon area overhead.

As for *instant-on/off*, the cache memory does not need to be necessarily non-volatile to implement the *rollback* at the cost of performance penalties for backup (in the case of write-back policy) and restore (cache warmup). If a *rollback* is performed at runtime (i.e., without powering down/restarting the processor), then the cache has to be flushed when restoring a *checkpoint* to avoid inconsistency between cache and memory.

Figure 8 compares the original architecture of the Amber processor and the required architecture for a non-volatile processor with both *instant-on/off* and *rollback* capabilities.

Assuming the processor uses MRAM-based FFs as described in Figure 6(a) and the main memory is duplicated, then the *rollback* procedure is as described in Algorithm 2.

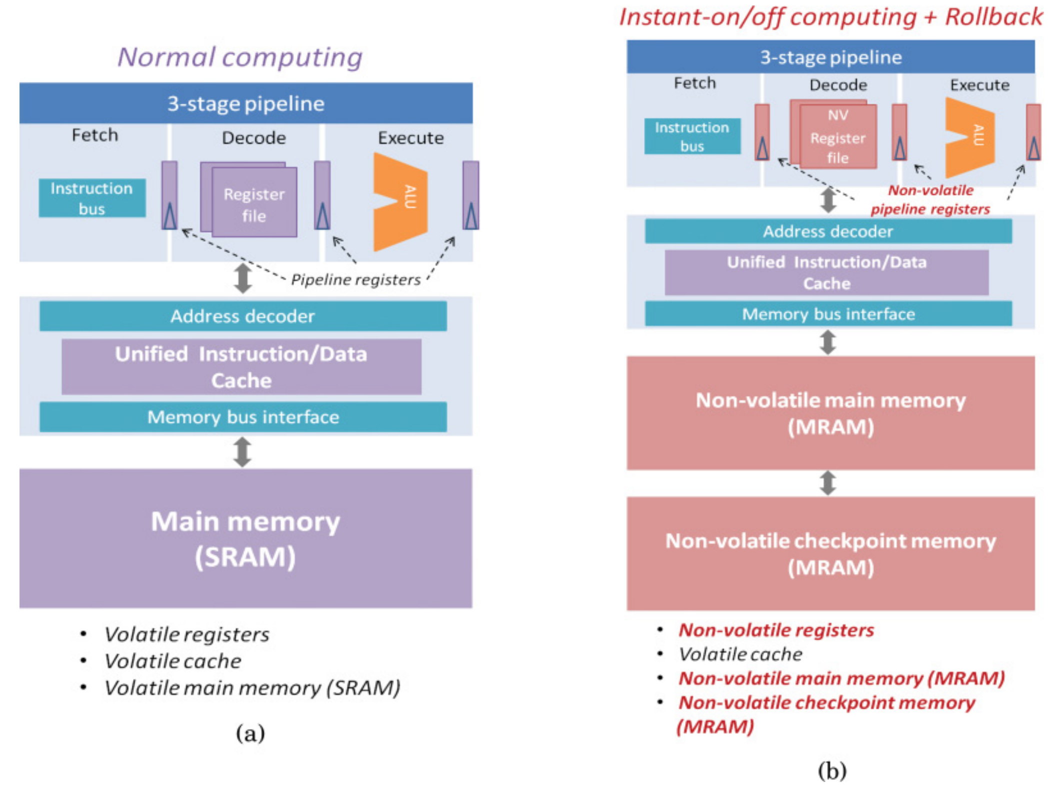


Fig. 8. Amber architecture with instant-on/off computing and rollback capability: (a) original Amber architecture and (b) Amber architecture with non-volatile MRAM and checkpoint memory for rollback.

ALGORITHM 2: Rollback Procedure

```
(1) Create checkpoints during the execution of the application;
(2) A system failure is detected;
(3) Stall the processor;
(4) Restore the last checkpoint which consists in;
    - Restoring the state of the FFs by reading the value from the MTJ into the CMOS flip-flop;
    - Restoring the main memory contents by copying data from the checkpoint memory to the main memory;
(5) Take the execution of the application up again.
```

4.4. RTL Simulation

This section aims at validating the *rollback* function via RTL simulation. It is recalled that the objective is to validate the possibility to completely save/restore the state of the processor. For the logic implementation, all the registers storing the state of the Amber core are duplicated as described in Figure 9. The original registers, named volatile registers in the figure, are used for the normal execution while the duplicated ones, named non-volatile registers in the figure, store the state of the core.

The main memory is also duplicated to allow a backup of the memory contents. As the objective is only to validate the *rollback* functionality, a non-synthesizable main memory model is used for fast simulation purpose. At the beginning of the application, both main memory and *checkpoint memory* contain the same data. Then, during the

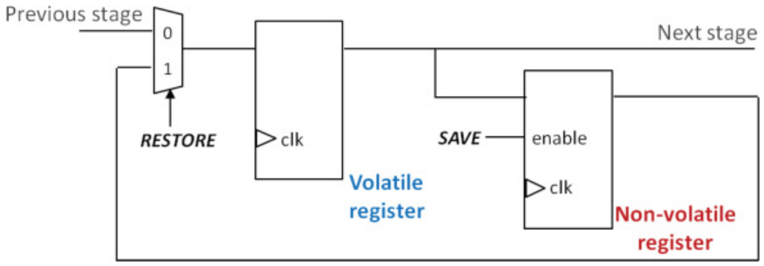


Fig. 9. Logic implementation of the registers.

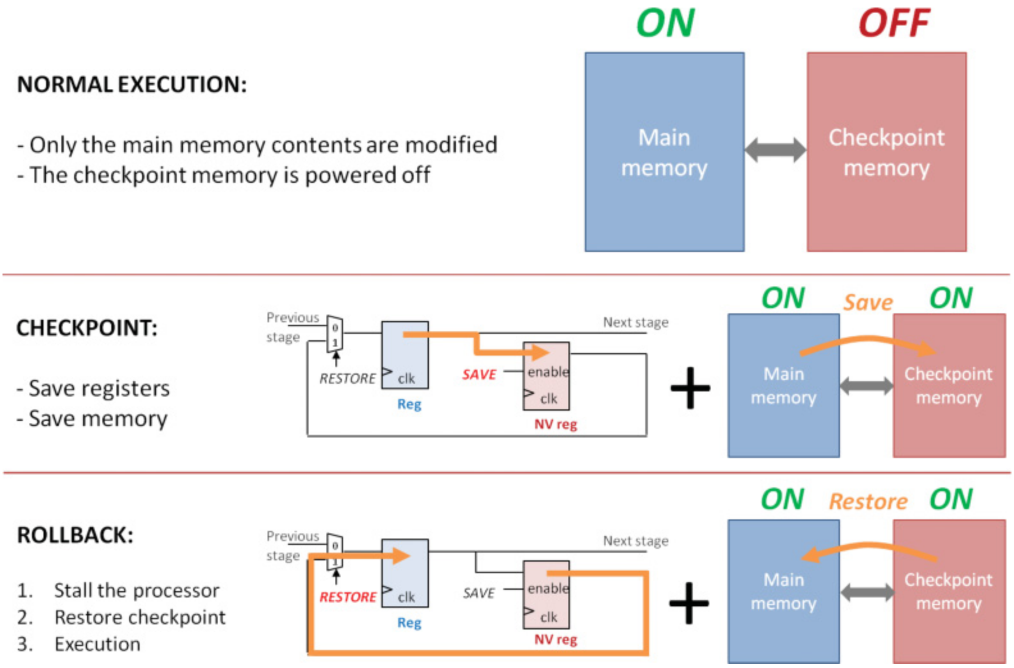


Fig. 10. Checkpointing and rollback.

execution of the application, only the main memory contents are modified. At the next *checkpoint*, only the main memory locations that were modified during the execution are copied into the *checkpoint memory* (Figure 10). Thus, copying all the contents of the main memory at each *checkpoint* is avoided. For that, all writes into the main memory between two *checkpoints* are tracked by storing all the corresponding memory addresses into a small buffer (Figure 11). In the case of a system failure before the next *checkpoint*, all the memory addresses present in this buffer correspond to the main memory locations to be restored for a *rollback*. Data are restored from the *checkpoint memory* to the main memory. If the address buffer is full before the next *checkpoint*, then a creation of a *checkpoint* is forced.

Figure 12 shows the simulation results of the *blowfish* application. In Figure 12(a), Electronic Code Book encryption mode is executed where a *checkpointing/rollback* is demonstrated. In a similar way in Figure 12(b), validation is made executing other encryption modes (CBC, CFB, and OFB modes). The *rollback* mechanism has also been validated with the *dhrystone 2.1* application (the simulation result is not shown for the sake of brevity).

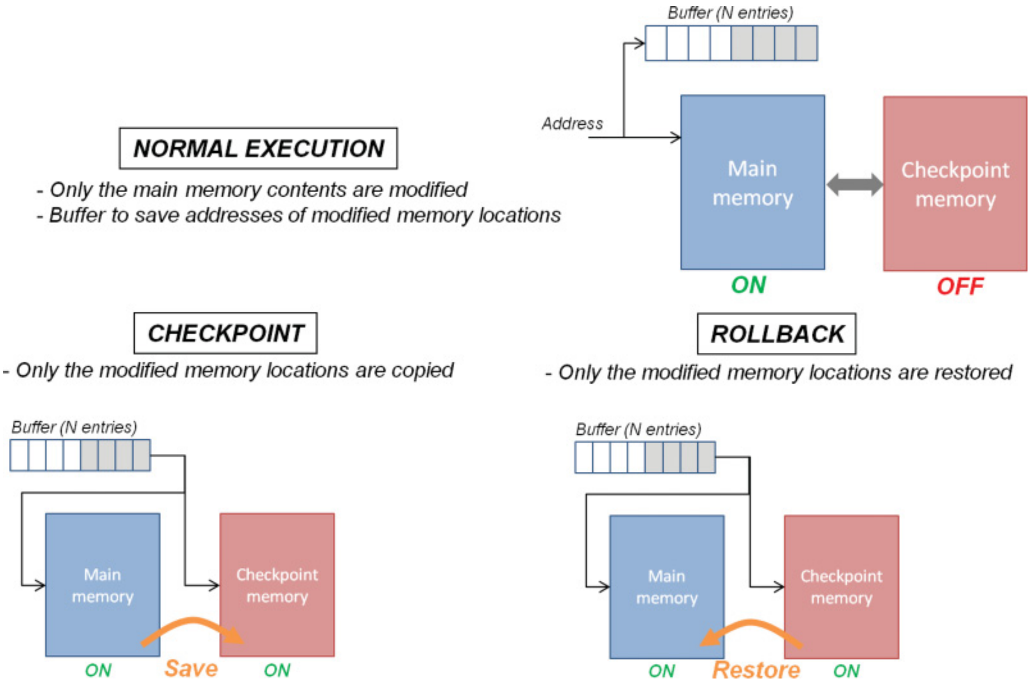
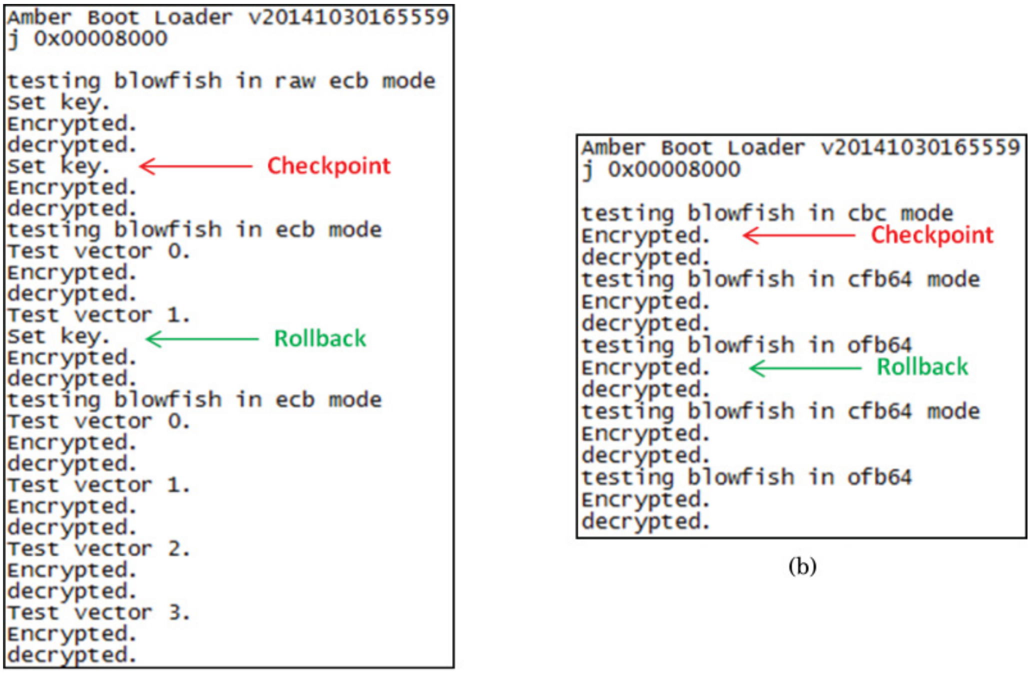


Fig. 11. Checkpointing and rollback: address buffer for memory changes tracking.



```
Amber Boot Loader v20141030165559
j 0x00008000

testing blowfish in raw ecb mode
Set key.
Encrypted.
decrypted.
Set key.    ←  Checkpoint
Encrypted.
decrypted.
testing blowfish in ecb mode
Test vector 0.
Encrypted.
decrypted.
Test vector 1.
Set key.    ←  Rollback
Encrypted.
decrypted.
testing blowfish in ecb mode
Test vector 0.
Encrypted.
decrypted.
Test vector 1.
Encrypted.
decrypted.
Test vector 2.
Encrypted.
decrypted.
Test vector 3.
Encrypted.
decrypted.
```

(a)

```
Amber Boot Loader v20141030165559
j 0x00008000

testing blowfish in cbc mode
Encrypted.    ←  Checkpoint
decrypted.
testing blowfish in cfb64 mode
Encrypted.
decrypted.
testing blowfish in ofb64
Encrypted.    ←  Rollback
decrypted.
testing blowfish in cfb64 mode
Encrypted.
decrypted.
testing blowfish in ofb64
Encrypted.
decrypted.
```

(b)

Fig. 12. Validation of the rollback capability (terminal outputs of the blowfish application).

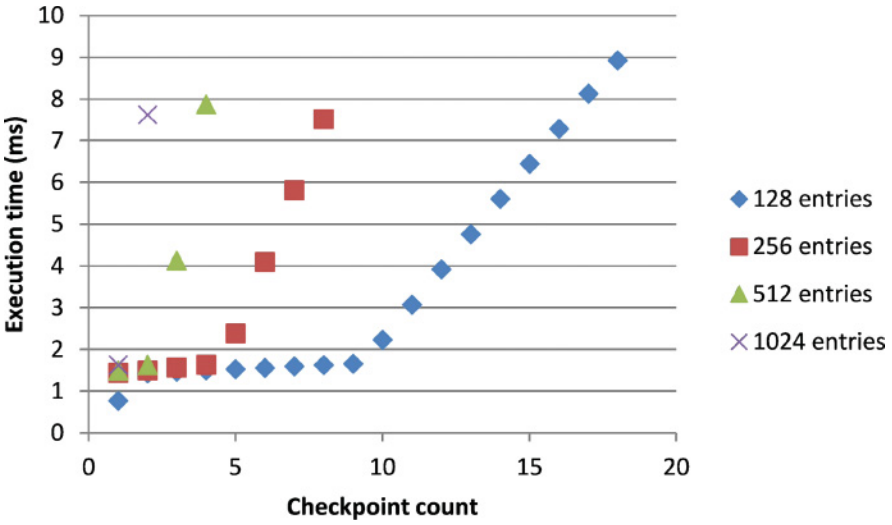


Fig. 13. Checkpoint count for different address buffer size (blowfish application).

Table II. Non-Volatile Flip-Flops Performance

	Latency (ns)		Energy (pJ)	
Technology	Restore	Back-up	Restore	Back-up
STT-MRAM [Chabi et al. 2014]	0.2	4	0.012	0.5
TAS-MRAM [Jovanovic et al. 2015]	0.13	16	0.012	5.2
OxRAM [Jovanović et al. 2014]	6	70	1.4	28
PCRAM [Choi et al. 2013]	100[4]	100	2	125

As mentioned above, the *checkpoint memory* is much smaller than the main memory if a real system is considered. The size will depend on the application. Figure 13 give us an idea of the memory overhead for the *blowfish* application. An encryption operation followed by a decryption operation are executed. The figure shows how many *checkpoints* are carried out for different address buffer sizes. The number of *checkpoints* corresponds to the number of time the buffer becomes full. As observed in the results, for a 1,024-entry buffer size, only two *checkpoints* are forced. If we consider this buffer size, then the *checkpoint memory* size has to be only 4kBytes.

5. MRAM-BASED NON-VOLATILE PROCESSOR: PERFORMANCE AND ENERGY

Many FFs based on MRAM were proposed in the literature to enable the design of non-volatile circuits such as in Na et al. [2013] and Zhao et al. [2014]. In order to estimate overall performance of the MRAM-based non-volatile processor, we consider information from the current state-of-the-art of MRAM-based FFs. For comparison purposes, the cases of OxRAM-based FFs and PCRAM-based FFs are also evaluated. Table II shows the time and the energy to backup/restore the state of a FF for the different NVMs.

[4] Note that the related work on PCRAM-based non-volatile flip-flop has clearly reported the backup time (i.e., 100ns) but not the restore time. Therefore, this work assumes that the backup and restore times are the same.

Considering all the parameters, MRAM-based FFs show the best performance. STT-MRAM has the smallest latency and energy for the backup. On the other hand, TAS-MRAM shows better performance to restore the non-volatile data.

These FFs are designed to have a dual-storage facility (hybrid). The CMOS stage of the FF uses cross-coupled inverters (latch) to store one data bit in its electrical (volatile) form. On the other hand, the magnetic stage uses an MTJ (in the case of MRAM) to store one non-volatile data bit.

In the rest of the section, the performance and energy implications of integrating MRAM into a processor architecture are discussed for each power mode, that is, active and sleep modes, and for the transitions between these modes.

5.1. Performance

5.1.1. Active Mode. Depending on the application, it could be very useful for the system to run at a higher speed in the active mode so it can return quickly to low-power mode. However, running at high frequency also increases the active power. The designer has to analyze the best case for the application, taking into account other factors such as the frequency at which the system needs to switch between the active and the sleep mode.

Use of fast-access registers into the processor is necessary if high-speed operation is required. Hybrid CMOS/MTJ FFs are suitable to build fast-access non-volatile registers thanks to their dual-storage facility. The CMOS stage storing data in its volatile form is used during normal execution of the system. The MTJ state is only used when there is a need to back up or restore the system state. Therefore, building the registers using these hybrid CMOS/MTJ FFs will not affect the performance of the processor in active mode.

5.1.2. Back-up. If required, then external Flash memory is used in commercial MCUs to restore the program and data when going out from low-power mode to active mode. It can also be used to log data before entering sleep mode if these data have to be used later in the application. This logging phase can take several milliseconds due to the long erase/program procedure of Flash memory. As this memory is usually external, additional latency will increase the backup process due to data transfer through the serial communication interface.

Thanks to its low access latency compared to Flash, MRAM is suitable to be integrated in both registers and main memory, allowing a small backup time. Since the main memory is assumed to be non-volatile, a backup of the state of the processor corresponds to a backup of the registers. Therefore, depending on the NVM technology used, the backup time of the processor is the backup time of the FF (see Table II).

5.1.3. Wake-up. The wake-up time is a key parameter for ultra-low-power devices. It informs us if the system can return from low-power mode to active mode quickly enough to accomplish the task at hand. Existing low-power MCUs include several low-power modes with different wake-up times so the customer can choose the appropriate configuration for a given application. The components that have a significant influence on the wake-up time if they are turned off are the embedded memories, that is, the registers and the main memory. Not retaining data into embedded memories requires the system to load data from external Flash memory at each power-up, which is time- and energy-consuming.

In the case of MRAM-based registers, the wake-up time of the processor corresponds to the time to restore the registers states after a power-up. Assuming the main memory is non-volatile, data are already available in this memory after a power-up. Hence, the wake-up time is the latency to restore the FF state (see Table II). In an actual system,

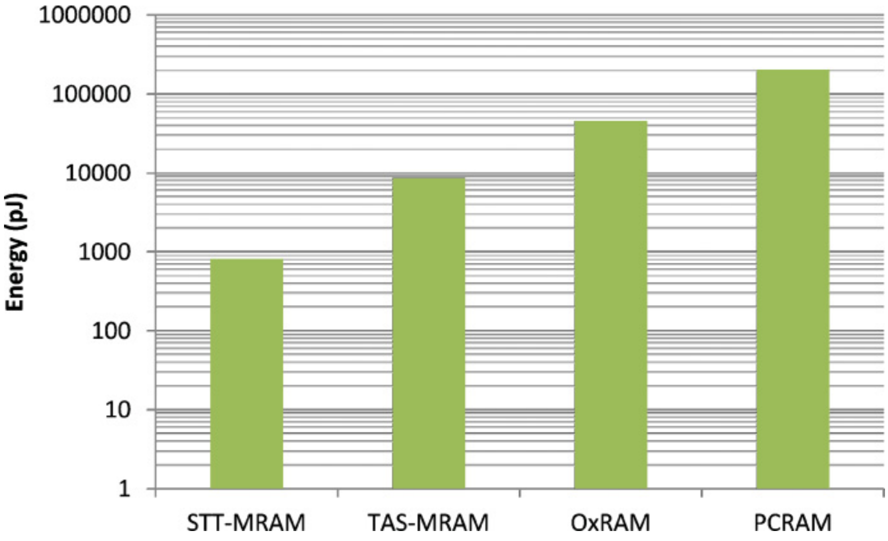


Fig. 14. Back-up energy (logarithmic scale).

it is also important to consider a delay for the power supply and clock stabilization times.

5.2. Energy

5.2.1. Active Mode. Depending on the amount of time the device remains in active mode and the frequency at which the MCU is running, the energy consumption can be more or less important. The active power consumption can be decreased if the MCU is running at low frequency. But the time to process data will increase. On the other hand, running at higher frequency will allow us to quickly return to low-power mode at the cost of higher active power consumption.

As for performance, using hybrid CMOS/MTJ FFs for registers into the processor will not affect the active energy consumption since this is the CMOS part that is used during normal operation.

5.2.2. Back-up. In addition to the long time it takes to log data, the backup phase using Flash memory is power hungry. The required current to erase and program Flash can vary from 4 to 12mA [Borgeson et al. 2012]. Besides, Flash memory still requires operation voltages of more than 10V [Meena et al. 2014].

Figure 14 estimates the backup energy of the Amber core when implementing nonvolatile MRAM-, OxRAM-, and PCRAM-based registers. As already mentioned in Section 4.2, 1,644 FFs have to be saved to retain the state of the Amber core. Therefore, the backup energy is estimated as the energy to write into the MTJ (in the case of MRAM) times the number of FFs. As observed in Figure 14, use of STT-MRAM leads to around 800pJ backup energy, whereas the backup energy reaches about 9nJ when TAS-MRAM is used. Use of OxRAM and PCRAM show respectively backup energies of 46nJ and 206nJ.

5.2.3. Wake-up. When estimating the average energy consumption of a system switching between active and sleep modes, the wake-up energy has to be considered. In current MCUs, this transition energy can be significantly high if returning to active mode requires data recovery from non-volatile memory (Flash). If the contents of both

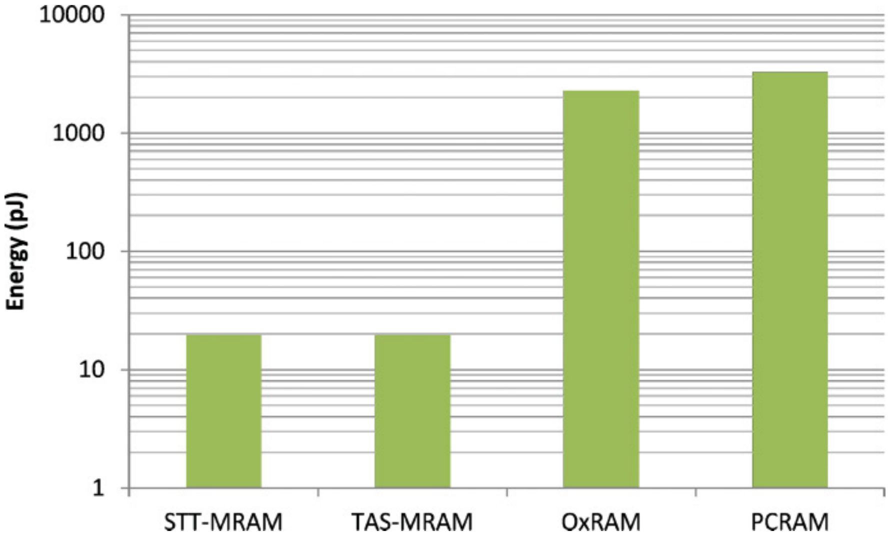


Fig. 15. Wake-up energy (logarithmic scale).

registers and main memory are not retained during low-power mode, then the boot process to run the program will also consume valuable energy. Wang et al. [2012] showed that the energy consumption to restore 1,607 FFs from off-chip Flash (on-chip Flash) is 1.3μJ (0.6μJ). Maintaining the registers and the main memory contents will decrease the wake-up time and so the wake-up energy. However, the leakage power in low-power mode will increase because of the current needed to enable data retention.

Figure 15 shows the wake-up energy for the Amber core if non-volatile FFs based on MRAM, OxRAM, or PCRAM are used. This energy corresponds to the read energy of the MTJ (in the case of MRAM) times the number of FFs. The results show a wake-up energy of 20pJ, 2.3nJ, and 3.3nJ respectively for MRAM (both STT and TAS), OxRAM, and PCRAM.

5.2.4. Sleep Mode. The leakage current is clearly an important factor for devices spending most of their time in low-power mode. The deeper the system sleeps (most components being turned off), the lower it consumes energy, but the longer it takes to return to active mode. Presence of leakage current in existing low-power MCUs is mainly due to the volatility of embedded memories. As already mentioned, it is necessary to keep these components turned on to allow fast wake-up time.

Integrating MRAM into the registers and the main memory for processors gives us the valuable advantage of removing power consumption when the system remains in sleep mode. As the leakage current increases dramatically with the decreasing size of the CMOS transistor, the non-volatility of MRAM is a very attractive feature that has the potential to be integrated not only at the storage level but also at main memory, cache memory, and the register level.

Figures 16(a) and (b) depict the profiles of the energy consumption respectively for a classical MCU without *instant-on/off* capability and a non-volatile MCU with *instant-on/off* capability. In these figures, switching between active and sleep modes is assumed to be periodic. P_{active} (T_{active}) corresponds to the power consumption (time) in active mode. $P_{leakage}$ (T_{sleep}) is the power consumption (time) in sleep mode. T_{wakeup} is the wake-up time. For the system related to Figure 16(a), we assume that data into

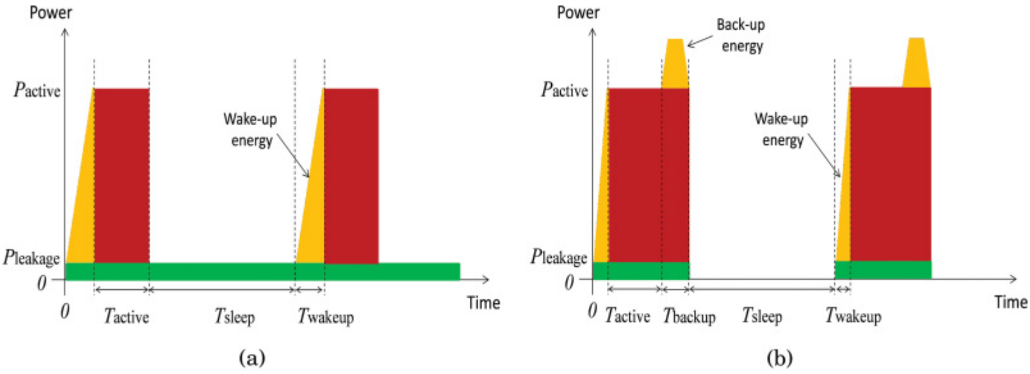


Fig. 16. Energy profile: (a) without instant-on/off (b) with instant-on/off.

the registers and the main memory are retained during sleep mode. Hence, we also assume there is no backup energy when switching from active mode to sleep mode.

As mentioned above, the active energy consumption is not changed when replacing CMOS-based registers by hybrid CMOS/MTJ registers. Therefore, the active power consumption is the same for both figures. The most visible energy gain when comparing the two figures is during sleep mode. Enabling the *instant-on/off* feature thanks to MRAM removes power consumption during this mode. However, an energy overhead is observed due to the required backup phase before entering sleep mode. Therefore, if MRAM is used (Figure 16(b)), the system will be more energy efficient if the backup phase consumes lower energy than the leakage energy normally consumed during the sleep mode in Figure 16(a). Thus, the time spent in sleep mode (T_{sleep}) has to be significant enough. The condition to reduce the total energy consumption when using MRAM is given by Equation (2) as follows:

$$(P_{active} + P_{leakage}) \times T_{backup} + E_{backup} < P_{leakage} \times T_{sleep}. \tag{2}$$

As a result, the condition that has to be verified on the time T_{sleep} is given by Equation (3) as follows:

$$T_{sleep} > \frac{(P_{active} + P_{leakage}) \times T_{backup} + E_{backup}}{P_{leakage}}. \tag{3}$$

This condition on T_{sleep} has been determined for the Amber core used in this work. The processor has been synthesized using a 65nm CMOS low-power High Threshold Voltage (HVT) process. On the basis of the synthesis results, the dynamic power and the leakage power are respectively equal to 173mW (at 40MHz) and 12mW. Using the backup time and energy estimated in this article for the four considered NVMs, the time T_{sleep} to reduce the total power consumption when using *instant-on/off* has to be higher than 130ns, 968ns, 4.9μs, and 18.7μs if the FFs are respectively based on STT-MRAM, TAS-MRAM, OxRAM, and PCRAM. These small time values, in the case of the Amber core and specifically when using MRAM, allows the processor to frequently go into sleep mode and thus shows the great potential of using MRAM to significantly reduce the total energy consumption of a system.

Assuming such a non-volatile processor using non-volatile FFs for the registers, it is important to note that writing at the same time into all the non-volatile parts of the FFs during the backup phase can lead to a high peak current, which is not appreciated for the system. If we consider the write current of the STT-MRAM used in this work, which is about 100μA, then the peak current to write into the 1,644 FFs of the Amber processor core exceeds 160mA. Therefore, in practice, the backup would be performed gradually.

6. RELATED WORK

Emerging NVMs have attracted a large part of the research community on the study of non-volatile logic circuits. Use of these memories for data storage and logic devices were and continue to be investigated due to the high interest it arouses for nanoelectronic systems such as field-programmable gate arrays (FPGA) and processors. Thanks to their CMOS-compatibility, emerging NVMs allow us to design hybrid CMOS/NVM logic elements capable of retaining their current state even after a power-down of the system. This section aims at giving an overview of the studies that have been done on non-volatile logic circuits.

6.1. Non-Volatile Logic Elements

Many studies explored the feasibility of designing non-volatile logic elements such as FFs and made a comparison with the counterpart CMOS-based circuits. Na et al. [2013] studied STT-MRAM-based FFs evaluating two different structures (merged latch and sensing circuit (MLS) and separated latch and sensing circuit (SLS)) and several sensing and write circuits for the non-volatile part. Zhao et al. [2014] investigated non-volatile logic gates and possible design optimizations using compact models of STT-MRAM and OxRAM NVM technologies. In addition, a non-volatile full-adder based on these two NVMs has been validated by simulation by Zhang et al. [2013] and compared with its counterparts based on CMOS only. Jabeur et al. [2014] evaluated a non-volatile FF based on the recent SOT-MRAM technology and compared it with a STT-MRAM-based FF. Within the context of energy-harvesting and IoT applications, Wang et al. [2010] and Jovanović et al. [2014] proposed a non-volatile FF respectively based on ferroelectric RAM (FeRAM) and OxRAM and validated by simulation the possibility to save/restore the logic state after a power-off of the device. Jovanovic et al. [2015], whose results have been used in this work for TAS-MRAM-based FFs, have made an exhaustive performance/energy analysis of a set of hybrid CMOS/MTJ cells that can be used for both data storage and logic devices in SoCs. Chabi et al. [2014], Jovanović et al. [2014], and Choi et al. [2013], whose results have been also used in this article, proposed respectively a hybrid CMOS/STT-MRAM FF, hybrid CMOS/OxRAM FF, and hybrid CMOS/PCRAM FF to allow system power-off in sleep mode.

6.2. Non-Volatile Reconfigurable Logic

Studies have been conducted to also explore the benefits of integrating emerging NVM into reconfigurable logic systems such as FPGAs. Major issues of such circuits are the low-power efficiency due to the high leakage current and logic density due to the use of SRAM for the configuration storage. Moreover, the volatility of SRAM forces the system to be reprogrammed at power-up from external Flash memory leading to a long start latency. Guillemenet et al. [2010] evaluated an FPGA architecture based on TAS-MRAM technology. In 2010, a full non-volatile FPGA was developed using 130nm CMOS technology and Crocus 120nm TAS-MRAM [Holland 2010]. Zhao et al. [2009], Paul et al. [2011], Ahari et al. [2014], and Turkyilmaz et al. [2014] respectively explored the use of STT-MRAM, PCRAM, and RRAM into FPGAs.

The main benefits of including emerging NVMs into reconfigurable circuits are the ability to turn off the system and save total power consumption thanks to the non-volatility. Moreover, a fast start-up time is possible in comparison with classical SRAM-based FPGAs. Previous studies also demonstrated new features using emerging NVMs for the configuration storage such as run-time reconfiguration and multi-context configuration capabilities.

6.3. Non-Volatile Processors

Processor architectures have also been targeted by the research community to evaluate the benefits of designing non-volatile processors including emerging NVM into the registers. Wang et al. [2012] presented the first fabricated non-volatile processor (130nm CMOS process) based on ferroelectric FFs with a 3μs wake-up time. The next year, Khanna et al. [2013, 2014] introduced a full non-volatile logic-based 32-bit microcontroller SoC (130nm CMOS process) also using FeRAM technology. Instead of using non-volatile FFs, small FeRAM-based memory arrays are distributed throughout the SoC to back up the FFs data. A complete area/performance/energy evaluation has been made that showed a 384ns wake-up time capability. These works demonstrated the feasibility of designing non-volatile processors with fast save/restore times and zero leakage standby mode. However, FeRAM does not have the same potential as MRAM, which shows faster access latency, lower access energy, and higher density [Meena et al. 2014]. A non-volatile microprocessor unit based on STT-MRAM has been evaluated by Koike et al. [2013] using a 90nm CMOS process. Simulation results showed a $3\mu s$ save/restore time for the pipeline register. Nonetheless, the design simulated was simplified by implementing only 12 instructions and the capacities of the instruction/ data memories were reduced to 32 words × 32 bits. A fully non-volatile 16-bit MCU using 90nm standard CMOS and three-terminal SpinRAM technology has been demonstrated in Sakimura et al. [2014]. Regarding the context of energy harvesting devices, Ma et al. [2015] presented a simulation platform to explore the design space for a non-volatile processor with different architectures and different input power sources. Xie et al. [2015] proposed a checkpointing scheme to avoid inconsistency in memory when restoring a previous valid state of a processor after a power failure.

6.4. Discussion

All the previous works clearly highlight the high interest of designing non-volatile systems to reduce the total energy consumption but also to integrate new interesting features thanks to the non-volatility. Although these studies validated the ability to save/restore the system state at device and circuit levels, they only consider power failures for *rollback*. As a result, previous papers only demonstrated a restore operation following a power-down of the processor. Therefore, the restore operation is the same for both *instant-on/off* and *rollback*. In this study, execution failures due to soft errors are also considered. This article demonstrates a *rollback* procedure at runtime, that is, without powering down the processor, and analyzes the memory overhead for the *rollback* procedure. In addition, this work analyzes the backup/restore performance/energy for several emerging memory technologies based on results from the state-of-the-art.

7. CONCLUSION

So far, emerging non-volatile memories are present in many studies to explore new computing paradigms in various systems such as reconfigurable logic devices, processors, and data storage. Aptitude to bring non-volatility at a deep level in a SoC, for example, at the register level, has the potential to drastically reduce the total power consumption of devices thanks to the *instant-on-off* capability. This article investigated the use of MRAM to design a non-volatile processor within the context of the internet of things. Validation of a complete backup/recovery of the state of a full 32-bit RISC-like processor has been performed via RTL simulation with two applications. The possibility to restore a previous valid state of the processor (*rollback*) at runtime has also been validated considering the context of an execution error (soft errors). Based on the results of a previous work that exhaustively characterized several hybrid CMOS/MTJ cells, performance and energy estimations of the backup/restore phases have been done.

Furthermore, the energy profile of a system switching between an active and a sleep mode has been analyzed to determine in which case the *instant-on / off* capability allows better energy efficiency. For future work, a real design of a full non-volatile processor using a design kit for MRAM is envisaged to accurately evaluate the performance, energy, and area overhead.

REFERENCES

Ali Ahari, Hossein Asadi, Behnam Khaleghi, and Mehdi B. Tahoori. 2014. A power-efficient reconfigurable architecture using PCM configuration technology. In *Conference on Design, Automation & Test in Europe*. European Design and Automation Association, 336.

K. Ando, S. Fujita, J. Ito, S. Yuasa, Y. Suzuki, Y. Nakatani, T. Miyazaki, and H. Yoda. 2014. Spin-transfer torque magnetoresistive random-access memory technologies for normally off computing. *J. Appl. Phys.* 115, 17 (2014), 172607.

Thomas W. Andre, Joseph J. Nahas, Chitra K. Subramanian, Bradley J. Garni, Halbert S. Lin, Asim Omair, and William L. Martino Jr. 2005. A 4-Mb 0.18-μm 1T1MTJ toggle MRAM with balanced three input sensing scheme and locally mirrored unidirectional write drivers. *IEEE J. Solid-State Circ.* 40, 1 (2005), 301–309.

Jacob Borgeson, Stephan Shauer, and Horst Diewald. 2012. Benchmarking MCU power consumption for ultra-low-power applications. White Paper (2012).

Djaafar Chabi, Weisheng Zhao, Erya Deng, Yue Zhang, Nesrine Ben Romdhane, Jacques-Olivier Klein, and Claude Chappert. 2014. Ultra low power magnetic flip-flop based on checkpointing/power gating and self-enable mechanisms. *IEEE Trans. Circ. Syst. I: Regul. Pap.* 61, 6 (2014), 1755–1765.

Jun-Myung Choi, Chul-Moon Jung, and Kyeong-Sik Min. 2013. PCRAM flip-flop circuits with sequential sleep-in control scheme and selective write latch. *J. Semicond. Technol. Sci.* 13, 1 (2013), 58–64.

L. Crippa, R. Micheloni, I. Motta, and M. Sangalli. 2008. Nonvolatile memories: NOR vs. NAND architectures. In *Memories in Wireless Systems*. Springer, 29–53.

Crocus. 2016. Homepage. Retrieved from http://www.crocus-technology.com/.

Jayita Das, Syed M. Alam, and Sanjukta Bhanja. 2014. STT-based non-volatile logic-in-memory framework. In *Field-Coupled Nanocomputing*. Springer, 173–193.

Cagdas Dirik. 2009. Performance analysis of NAND flash memory solid-state disks.

B. N. Engel, Johan Åkerman, B. Butcher, R. W. Dave, M. DeHerrera, M. Durlam, G. Grynkewich, J. Janesky, S. V. Pietambaram, N. D. Rizzo, and others. 2005. A 4-Mb toggle MRAM based on a novel bit and switching method. *IEEE Trans. Magnet.* 41, 1 (2005), 132–136.

Pietro Gambardella and Ioan Mihai Miron. 2011. Current-induced spin–orbit torques. *Philos. Trans. Roy. Soc. Lond. A: Math. Phys. Eng. Sci.* 369, 1948 (2011), 3175–3197.

Yoann Guillemenet, Lionel Torres, and Gilles Sassatelli. 2010. Non-volatile run-time field-programmable gate arrays structures using thermally assisted switching magnetic random access memories. *IET Comput. Dig. Tech.* 4, 3 (2010), 211–226.

Matthew R. Guthaus, Jeffrey S. Ringenberg, Dan Ernst, Todd M. Austin, Trevor Mudge, and Richard B. Brown. 2001. MiBench: A free, commercially representative embedded benchmark suite. In *2001 IEEE International Workshop on Workload Characterization (WWC-4)*. IEEE, 3–14.

C. Holland. 2010. First MRAM-based FPGA taped-out. Retrieved from http://www.eetimes.com/General/DisplayPrintViewContent? contentItemId 4200035.

Y. Huai, J. Zhang, Y. Zhou, X. Wang, E. Abedifard, Z. Wang, X. Hao, D. Jung, K. Satoh, H. Gan, and others. 2015. PMTJ driven STT MRAM with 300mm process. In *2015 IEEE Magnetics Conference (INTERMAG)*. IEEE, 1–1.

Kenshin Ikegami, Hiroki Noguchi, Chikayoshi Kamata, M. Amano, Kiyohiko Abe, K. Kushida, Eiji Kitagawa, Toshihiko Ochiai, Naoharu Shimomura, A. Kawasumi, and others. 2014. A 4ns, 0.9 V write voltage embedded perpendicular STT-MRAM fabricated by MTJ-Last process. In *Proceedings of Technical Program-2014 International Symposium on VLSI Technology, Systems and Application (VLSI-TSA)*. IEEE, 1–2.

ITRS. 2013. International technology roadmap for semiconductors. Retrieved from http://www.itrs.net/.

Kotb Jabeur, Gregory Di Pendina, Fabrice Bernard-Granger, and Guillaume Prenat. 2014. Spin orbit torque non-volatile flip-flop for high speed and low energy applications. *IEEE Electr. Device Lett.* 35, 3 (2014), 408–410.

Guenole Jan, Luc Thomas, Son Le, Yuan-Jen Lee, Huanlong Liu, Jian Zhu, Ru-Ying Tong, Keyu Pi, Yu-Jen Wang, Dongna Shen, and others. 2014. Demonstration of fully functional 8Mb perpendicular

STT-MRAM chips with sub-5ns writing for non-volatile embedded memories. In *Digest of Technical Papers, 2014 Symposium on VLSI Technology (VLSI-Technology)*. IEEE, 1–2.

Bojan Jovanovic, Raphael M. Brum, and Lionel Torres. 2015. Comparative analysis of MTJ/CMOS hybrid cells based on TAS and in-plane STT magnetic tunnel junctions. *IEEE Trans. Magnet.* 51, 2 (2015), 1–11.

N. Jovanović, O. Thomas, E. Vianello, J. M. Portal, B. Nikolić, and L. Naviner. 2014. OxRAM-based non volatile flip-flop in 28nm FDSOI.

Stamatis Karnouskos, Pedro José Marrón, Giancarlo Fortino, Luca Mottola, and José Ramiro Martínez-de Dios. 2014. *Applications and Markets for Cooperating Objects*. Springer.

Saarthak Khanna, Steven Bartling, Michael Clinton, Scott Summerfelt, Jose Rodriguez, and Hugh McAdams. 2013. Zero leakage microcontroller with 384ns wakeup time using FRAM mini-array architecture. In *2013 IEEE Asian Solid-State Circuits Conference (A-SSCC)*. IEEE, 21–24.

Saarthak Khanna, Steven C. Bartling, Michael Clinton, Scott Summerfelt, John A. Rodriguez, and Hugh P. McAdams. 2014. An FRAM-based nonvolatile logic MCU SoC exhibiting 100% digital state retention at 0 V achieving zero leakage with 400-ns wakeup time for ULP applications. *IEEE J. Solid-State Circ.* 49, 1 (2014), 95–106.

A. V. Khvalkovskiy, D. Apalkov, S. Watts, R. Chepulskii, R. S. Beach, A. Ong, X. Tang, A. Driskill-Smith, W. H. Butler, P. B. Visscher, and others. 2013. Basic principles of STT-MRAM cell operation in memory arrays. *J. Phys. D: Appl. Phys.* 46, 7 (2013), 74001–74020.

Hiroki Koike, Sadahiko Miura, Hiroaki Honjo, Toshinari Watanabe, Hideo Sato, Soshi Sato, Takashi Nasuno, Yasuo Honjo, Naguchi, and others. 2016. Demonstration of yield improvement for on-via MTJ using a 2-Mbit 1T-1MTJ STT-MRAM test chip. In *2016 IEEE 8th International Memory Workshop (IMW)*. IEEE, 1–4.

Hideaki Koike, Takashi Ohsawa, Shoji Ikeda, Takahiro Hanyu, Hideo Ohno, Tetsuo Endoh, Noboru Sakimura, Ryusuke Nebashi, Yukihide Tsuji, Ayuka Morioka, and others. 2013. A power-gated MPU with 3-microsecond entry/exit delay using MTJ-based nonvolatile flip-flop. In *2013 IEEE Asian Solid-State Circuits Conference (A-SSCC)*. IEEE, 317–320.

K. Lewotsky. 2013. Tech trends: Details on everspins ST-MRAM. Retrieved from http://www.eetimes.com/document.asp?doc_id=1280267

Kaisheng Ma, Yang Zheng, Shuangchen Li, Karthik Swaminathan, Xueqing Li, Yongpan Liu, Jack Sampson, Yuan Xie, and Vijaykrishnan Narayanan. 2015. Architecture exploration for ambient energy harvesting nonvolatile processors. In *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 526–537.

Jagan Singh Meena, Simon Min Sze, Umesh Chand, and Tseung-Yuen Tseng. 2014. Overview of emerging nonvolatile memory technologies. *Nanoscale Res. Lett.* 9, 1 (2014), 1–33.

Rino Micheloni, Luca Crippa, and Alessia Marelli. 2010. *Inside NAND Flash Memories*. Springer Science & Business Media.

Jagadeesh Subbaiah Moodera, Lisa R. Kinder, Terrilyn M. Wong, and R. Meservey. 1995. Large magnetoresistance at room temperature in ferromagnetic thin film tunnel junctions. *Phys. Rev. Lett.* 74, 16 (1995), 3273.

Taehui Na, Kyungho Ryu, Jisu Kim, Seung-Hyuk Kang, and Seong-Ook Jung. 2013. A comparative study of STT-MTJ based non-volatile flip-flops. In *2013 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 109–112.

Hiroki Noguchi, Kazutaka Ikegami, Keiichi Kushida, Keiko Abe, Shogo Itai, Satoshi Takaya, Naoharu Shimomura, Junichi Ito, Atsushi Kawasumi, Hiroyuki Hara, and others. 2015. 7.5 a 3.3 ns-access-time 71.2μW/MHz 1Mb embedded STT-MRAM using physically eliminated read-disturb scheme and normally-off memory architecture. In *2015 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 1–3.

Hiroki Noguchi, Keiichi Kushida, Kenshin Ikegami, Kiyohiko Abe, Eiji Kitagawa, Shintaro Kashiwada, Chikayoshi Kamata, Atsushi Kawasumi, Hideki Hara, and Shinobu Fujita. 2013. A 250-MHz 256b-I/O 1-Mb STT-MRAM with advanced perpendicular MTJ based dual cell for nonvolatile magnetic caches to reduce active power of processors. In *2013 Symposium on VLSI Technology (VLSIT)*. IEEE, C108–C109.

Janusz J. Nowak, Ray P. Robertazzi, Jonathan Z. Sun, Guohan Hu, Jeong-Heon Park, JungHyuk Lee, Anthony J. Annunziata, Gen P. Lauer, Raman Kothandaraman, Eugene J. OSullivan, and others. 2016. Dependence of voltage and size on write error rates in spin-transfer torque magnetic random-access memory. *IEEE Magnet. Lett.* 7 (2016).

D. Pala, G. Causapruno, M. Vacca, F. Riente, G. Turvani, M. Graziano, and M. Zamboni. 2015. Logic-in-memory architecture made real. In *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1542–1545.

Somnath Paul, Saibal Mukhopadhyay, and Swarup Bhunia. 2011. A circuit and architecture codesign approach for a hybrid CMOS–STTRAM nonvolatile FPGA. *IEEE Trans. Nanotechnol.* 10, 3 (2011), 385–394.

I. L. Prejbeanu, S. Bandiera, J. Alvarez-Hérault, R. C. Sousa, B. Dieny, and J. P. Nozieres. 2013. Thermally assisted MRAMs: Ultimate scalability and logic functionalities. *J. Phys. D: Appl. Phys.* 46, 7 (2013), 074002.

I. L. Prejbeanu, M. Kerekes, R. C. Sousa, H. Sibuet, O. Redon, B. Dieny, and J. P. Nozieres. 2007. Thermally assisted MRAM. *J. Phys.: Condens. Matter* 19, 16 (2007), 165218.

Noboru Sakimura, Yukihide Tsuji, Ryusuke Nebashi, Hiroaki Honjo, Ayuka Morioka, Koichi Ishihara, Keizo Kinoshita, Shunsuke Fukami, Shun Miura, Naoki Kasai, and others. 2014. 10.5 A 90nm 20MHz fully nonvolatile microcontroller for standby-power-critical applications. In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. IEEE, 184–185.

Conor Santifort. 2010. Amber ARM-compatible core. Retrieved from *OpenCores.org*.

STM32L1. 2016. MCU. Datasheet. (MCU). Retrieved from http://www.st.com/web/en/resource/technical/document/datasheet/CD00277537.pdf.

Ogun Turkyilmaz, Santhosh Onkaraiah, Marina Reyboz, Fabien Clermidy, Costin Anghel, Jean-Michel Portal, Marc Bocquet, and others. 2014. RRAM-based FPGA for normally off, instantly on applications. *J. Parallel Distrib. Comput.* 74, 6 (2014), 2441–2451.

I. Wali, Arnaud Virazel, A. Bosio, P. Girard, S. Pravossoudovitch, and M. Sonza Reorda. 2016. A hybrid fault-tolerant architecture for highly reliable processing cores. *J. Electron. Test.* (2016), 1–15.

Jue Wang, Yongpan Liu, Huazhong Yang, and Hui Wang. 2010. A compare-and-write ferroelectric nonvolatile flip-flop for energy-harvesting applications. In *2010 International Conference on Green Circuits and Systems (ICGCS)*. IEEE, 646–650.

Yiqun Wang, Yongpan Liu, Shuangchen Li, Daming Zhang, Bo Zhao, Mei-Fang Chiang, Yanxin Yan, Baiko Sai, and Huazhong Yang. 2012. A 3us wake-up time nonvolatile processor based on ferroelectric flip-flops. In *2012 Proceedings of the ESSCIRC (ESSCIRC)*. IEEE, 149–152.

Reinhold P. Weicker. 1984. Dhrystone: A synthetic systems programming benchmark. *Commun. ACM* 27, 10 (1984), 1013–1030.

Mimi Xie, Mengying Zhao, Chen Pan, Jingtong Hu, Yongpan Liu, and Chun Jason Xue. 2015. Fixing the broken time machine: Consistency-aware checkpointing for energy harvesting powered non-volatile processor. In *Proceedings of the 52nd Annual Design Automation Conference*. ACM, 184.

Jing Yu, María Jesús Garzarán, and Marc Snir. 2008. Efficient software checking for fault tolerance. In *IEEE International Symposium on Parallel and Distributed Processing, 2008. IPDPS 2008*. IEEE, 1–5.

Ye Zhang, E. Y. Deng, J. O. Klein, Damien Querlioz, Dafine Ravelosona, Claude Chappert, Weisheng S. Zhao, M. Moreau, J. M. Portal, Michael Bocquet, and others. 2013. Synchronous full-adder based on complementary resistive switching memory cells. In *2013 IEEE 11th International New Circuits and Systems Conference (NEWCAS)*. IEEE, 1–4.

Weisheng Zhao, Eric Belhaire, Claude Chappert, and Pascale Mazoyer. 2009. Spin transfer torque (STT)-MRAM–based runtime reconfiguration FPGA circuit. *ACM Trans. Embedd. Comput. Syst.* 9, 2 (2009), 14.

Weisheng Zhao, Mathieu Moreau, Erya Deng, Yue Zhang, J.-M. Portal, Jacques-Olivier Klein, Michael Bocquet, Hassen Aziza, Damien Deleruyelle, Candice Muller, and others. 2014. Synchronous non-volatile logic gate design based on resistive switching memories. *IEEE Trans. Circ. Syst. I: Regul. Pap.* 61, 2 (2014), 443–454.

Received October 2015; revised August 2016; accepted September 2016