Towards a Nonvolatile Implementation of Neander, an Accumulator Based 8-bit Processor

Bruna C. Cagliari, Paulo F. Butzen and Raphael M. Brum
Departamento de Engenharia Elétrica
Universidade Federal do Rio Grande do Sul
Av. Osvaldo Aranha, 103 - 90035-190 Porto Alegre, RS, Brazil
bccagliari@inf.ufrgs.br, paulo.butzen@ufrgs.br, brum@ufrgs.br

***Abstract*—The rise of the Internet of Things (IoT) and the constant growth of portable electronics have leveraged the concern with energy consumption due to the use of batteries in these devices. The nonvolatile memory (NVM) emerged as a solution to mitigate the problem due to its ability to retain data on sleep mode without a power supply. Nonvolatile processors (NVPs), in turn, have improved performance in centralized NVM architectures, providing nonvolatile flip-flops (NVFFs) that store the system states in parallel, allowing a quicker resumption of the system after power on. Neander is a hypothetical processor created for educational purposes. This work proposes to validate the first step to implement the nonvolatile Neander: describe it through the Register Transfer Level (RTL) design — which was successfully performed. This approach was made possible by separating the combinational logic, implemented through control signals, from the sequential logic composed by the analog D flip-flops at the transistor level.**

***Index Terms*—Register Transfer Level (RTL), nonvolatile processor (NVP), Neander**

I. Introduction

The growth of the Internet of Things (IoT) has increased a search for efficient energy solutions [1]. Energy harvesting has become a viable solution for self-powered systems, thus replacing batteries and providing remarkably long life, negligible maintenance cost and sustainable factor [2]. However, the energy from the environment is inherently unstable, causing great concern about data loss [2]. The nonvolatile processor (NVP) is proposed as a way to deal with unexpected failures, allowing the use of intermittent sources[2].

NVP mitigates energy and performance overheads in systems with centralized nonvolatile memory (NVM) [3]. The NVP has nonvolatile flip-flops (NVFFs) that store the system states every n cycles, being able to recall pre-stored data after a suspension or failure [4]. Backing up and restoring all states in parallel, the NVP enables a quick resumption [4].

The increasing use of portable electronic devices causes a new concern about battery extension [3]. The use of NVP devices allows the memory cell to maintain the states on sleep mode without the need for energy, demonstrating a great advantage [3].

Due to their qualities, NVPs have been widely investigated [3]. In 2012, Wang et al. [4] presented the first fabricated nonvolatile processor based on ferroelectric flip-flops. After that, Koike et al. in 2013 [5] presented an evaluation of a nonvolatile microprocessor unit (MPU) based on STT-MRAM.

Neander [6] is a hypothetical processor created for educational purposes. The final objective of this project is to implement a nonvolatile Neander. NVMs are still experimental technologies, and the behavior of nonvolatile cells must be carefully tuned to match the desired specification. CMOS logic behavior, on the other hand, is well-known and reliable. For this reason, we chose to implement the NVFFs in circuit-level, while keeping a high-level abstraction for combinational logic. This work's proposal is to validate a mixed-signal Neander implementation, combining an analog model for flip-flops with a purely digital implementation of the Neander's random logic.

This paper is organized as follows: In Section II, important concepts are presented for the understanding of this work. The work proposal is presented in Section III and the Section IV shows a brief overview of the combinational implementation proposed. Results are in Section V and the conclusions and future work are presented in Section VI.

II. Background

A. Neander

Neander is a simple hypothetical processor designed to introduce undergraduates to computer architecture. In its simplicity it has a small set of 11 instructions and two status registers: negative (N) and zero (Z) used in conditional bypass operations. Fig. 1 [7] illustrates the Neander architecture.

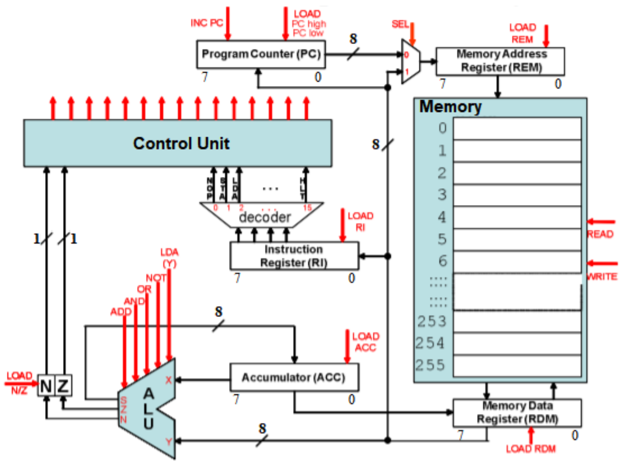

Fig. 1. Neander architecture (adapted from [7])

Neander is formed by a memory, a decoder, an accumulator (AC), an arithmetic logic unit (ALU) that performs basic operations on the accumulator, a multiplexer (MUX) and a program counter (PC) which points to the current memory address. There are also a number of registers: a Memory Address Register (REM), a Memory Data Register (RDM) an Instruction Register (RI) and a State NZ Register (evaluates whether AC is negative or zero). All these components operate together through control signals determined by the Control Unit that guarantee the correct logic functioning.

Data is 8-bit wide and represented in two's complement. There is only one addressing mode: direct. All instructions have at least one byte, where the first four bits indicate the instruction opcode (operation code) and the last ones don't matter. In STA, LDA, ADD, AND, OR statements the second byte indicates the memory address of the operand. In the JMP, JN and JZ instructions, the second byte indicates the memory address to be pointed by the PC. In NOP, NOT and HLT statements there is not a second byte. Table 1 presents the instructions, respective opcodes and descriptions.

TABLE I
INSTRUCTIONS OF NEANDER

Opcode	Mnemonic	Description
0000	NOP	Initializes memory
0001	LDA	AC $\leftarrow$ Memory Data
0010	STA	Memory $\leftarrow$ AC
0011	ADD	AC $\leftarrow$ AC + Memory Data
0100	OR	AC $\leftarrow$ AC OR Memory Data
0101	AND	AC $\leftarrow$ AC AND Memory Data
0110	NOT	AC $\leftarrow$ NOT AC
1000	JMP	PC $\leftarrow$ Memory Data
1001	JN	If AC $<$ 0 then PC $\leftarrow$ Memory Data
1010	JZ	If AC $=$ 0 then PC $\leftarrow$ Memory Data
1111	HLT	Ends execution

To illustrate the Neander operation, the data flow associated to the ADD operation is described below. When the PC points to the memory address where the ADD instruction is located, the opcode of the instruction will pass from memory to RI. After the operation is identified, the counter will add one more to its output so that the next memory address to be read will indicates the operand location. The memory will read the second byte of the instruction and the MUX will pass the memory out to the REM. After that, the memory will receive the operand address and then the data located will be transferred to the ALU and added to the accumulator content.

B. Magnetic Tunnel Junction

Magnetoresistive random access memory (MRAM) is one of the most promising spintronics applications [8]. It was the first commercial use of the magnetic tunnel junction (MTJ) [9], the nonvolatile spintronic device that can store data without a power supply and is the key technology in each memory cell [10].

MTJ is basically composed of two ferromagnetic layers, one fixed and the other one free, separated by a very thin tunnel barrier ($\sim$ 1 nm) [9]. The relative orientation of the two ferromagnetic layer determines the variable resistance of the device: low (parallel) or high (antiparallel) [9]. The high resistance represents the logical value '1' and the low resistance represents the '0' logical value [9].

The switching methods of magnetic state influence directly the power, speed and area performance of hybrid MTJ / CMOS circuits [11]. The magnetic tunneling junction based on the spin transfer torque (STT-MTJ) is the most researched MTJ variant and one of the most promising MRAM technologies, presenting low energy consumption, compact design and CMOS compatibility [10], [12]. The operation of the STT-MTJ is illustrated in Fig. 2 [13]. To change the magnetization state of MTJ in this variant, the switching current (I) is passed through the device [13]. The MTJ state changes from parallel (P) to antiparallel (AP) if the forward biased current is greater than the critical current (Icrit) and its state will return if the reverse biased current is greater than the critical current [13].

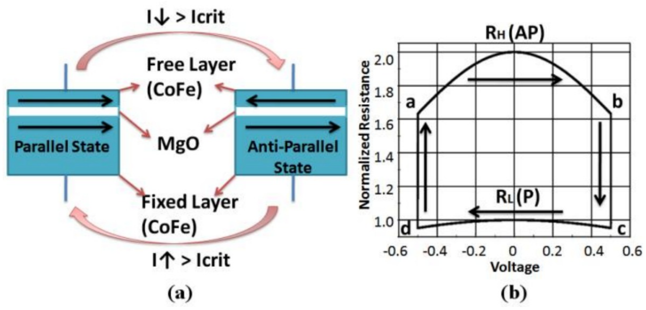

Fig. 2. (a) STT-MTJ structure and state change diagram; and (b) V-R curve of STT-MTJ device, both published in [13]

The hybrid integration of spintronic devices, such as MTJs, with CMOS (Complementary metal oxide semiconductor) technologies is of great importance due to the viability of ultra-low power solutions, increasingly necessary due to the growth of IoT and the use of portable electronics [1].

C. Nonvolatile Flip-Flops

NVFFs are key memory cells in NVPs, integrating CMOS technologies with a nonvolatile device, such as an MTJ [2]. It is also a promising solution to implement zero standby power dissipation and instant on/off of a system-on-a-chip [12]. One of its great advantages is the possibility of being disconnected from the power supply on sleep mode without losing the stored data, reducing the leakage current and energy dissipation [14].

There also 3 more modes besides the sleep one: latch, write and read modes [12]. Due to the relevance of the STT-MTJ, it will be considered as the nonvolatile device in the example. During latch mode, a NVFF operates as a conventional flip-flop [12]. In the write mode, the latched data is written into the two MTJs with opposite states before the system power is turned off [12]. The reading of the data can be made by equalizing the voltages of the left and right branches of the latch, moving it to a metastable state [10]. After leave it, the state of MTJ resistances will determine the stable state [10]. An other way to read is to pass a read current through the device and compare it to a reference current, but this approach requires an additional circuit [12].

NVFF can be presented in two different structures: merged latch and sensing circuit (MLS) and separated latch and sensing circuit (SLS) [12]. STT-MTJ variant has great scalability, as the write current decreases in proportional to the size of the MTJ [12]. However, with a lower critical current, it also becomes a greater problem to pass a read current that does not cause any switching [12]. The SLS structure emerged so that the latch and the sensing circuit could be optimized simultaneously, in order to circumvent the STT-MTJ problem with the reading current [12].

D. Rollback Scheme

For a processor to be considered nonvolatile, it must have both abilities: Instant on / off (shutdown and return to the execution from the previously saved state) and Backward Error-Recovery (recover from an execution error). [10] Rollback is the technique that supports these abilities.

Every *n* cycles it is possible to make a checkpoint that consists of storing the current states of the processor [10]. If the processor goes through shutdown or if a fault has been found, the rollback process is triggered and the processor will receive the states of the last checkpoint considered safe [10].

III. Work Proposal

To implement the nonvolatile Neander processor, the first step is to refactor Neander in combinational logic. The objective of this work is to present the combinational implementation of the Neander through the RTL design abstraction. RTL code is described as a synchronous circuit in terms of analog registers and combinational logic between them. Hardware description languages (HDLs) were used to create high-level representations of a circuit, making the complexity of the project more manageable. The use of the RTL design requires separate the combinational logic from the sequential logic.

The sequential logic of the Neander implementation comprises analog D-type flip-flop (DFF) that act as registers for the system context. In the adopted configuration, the DFF passes the input on rising clock edge and keeps the same output the rest of the time, unless the reset receives '0' which will force the output to be '0'. There are signals responsible for all inputs of the registers. These signals are determined by the logic presented in Neander and depends on the operation to be performed, the memory position and the current state of the operation.

The combinational logic is the part that defines the signals that will determine the inputs of the registers, keeping all states consistent. It is essential that it is combinational so that the system can be turned off and resumed using nonvolatile memory cells since this logic depends only on the current inputs that, in this case, would be provide by the nonvolatile registers.

IV. Experimental Setup

The combinational implementation of the Neander was based on assign statements. The equation for assign statement is presented bellow. As a designation statement continues, whenever the expression on the right undergoes any change, the expression on the left is immediately changed.

```
assign < net_expression > = < expression of different
signals or constant value >
```

The assign statements makes it possible to determine the load signals and the inputs of the registers using the ternary conditional operator. A simple example is the operation of the Instructions Register (RI) described in combinational logic presented below, where: loadRI represents the load signal that decides the input of the register; state corresponds to one of the eight states implemented that guarantee the correct operation of the Neander; smem is the output of memory; e_ri is the input of the RI and sri is the output. In the second logic state of the Neander, the loadIR receives a high signal so the output of the memory is loaded in this register. This output corresponds to the opcode of the instruction that must be executed. When the loadRI is off, the input of the DFF must be kept the same. Assign statements were also used in decisions corresponding to ALU and MUX.

```
assign loadRI = (state == 2) ? 1 : 0;
assign e_ri = (loadRI == 1) ? smem : sri;
```

The analog DFFS cells were implemented using NCSU FreePDK 45nM synthetic CMOS technology. Analog to digital converter (ADC) and digital to analog converter (DAC) were used as bridges between the analog modules of the sequential part and the digital modules of the combinational part.

For verification, Cadence Design Systems Incisive and Spectre tool sets were used. Due to the mix of analog and digital signals, all modules except ALU, MUX and memory were implemented in Verilog-AMS, which includes analog and mixed-signal extensions (AMS).

V. Results

All the 11 operations of the combinational Neander implementation were successfully validated through the RTL design. As the ADD operation was briefly explained in Section II, it was chosen for a representative simulation.

Table 2 presents the part of the memory used in this simulation and a brief explanation of each line. Fig. 3 shows the representative simulation: the operation of adding the value 248 (-8 in signed decimal) to the zeroed accumulator. Relevant signals that facilitate the reader's understanding were considered. The signals are inputs (e_ac, e_pc and e_ri) and outputs (sac, snz, spc, sri) of the registers, load signals (loadAC, loadNZ, loadPC and loadRI), the output of the memory (smem), the State and the incPC signal which is triggered when the PC output must be increased by one. The simulation presented in Fig. 3 took 8 nanoseconds and the data are represented in unsigned decimal. In Table 2 the data are represented in binary.

TABLE II
MEMORY ADDRESSES USED IN THE REPRESENTATIVE SIMULATION

Memory address	Data	Data meaning
0	00110000	ADD
1	00000101	Operand address
2	11110000	HLT
3	00000000	NOP
4	00000000	NOP
5	11111000	Operand

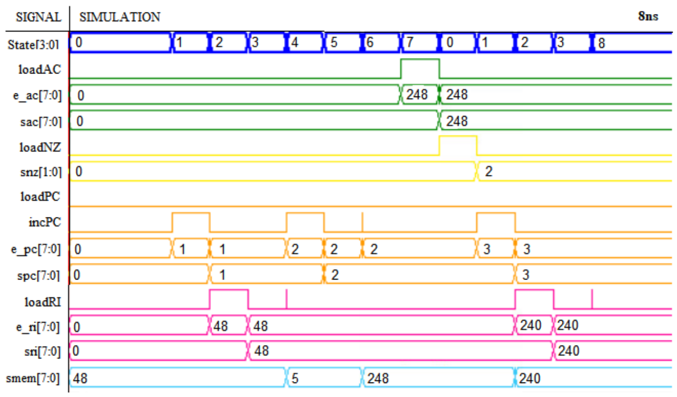

Fig. 3. Mixed-signal simulation of ADD operation.

Fig. 4 represents the sequential logic of the Neander through a mixed-signal simulation, showing the correctness of the DFF and ADC executions. In this simulation, a_clock is the analog clock signal and the affd_ac[7] is the first bit in the analog input of the AC that appears on the output in affo_ac[7] and converted to a digital signal in ffo_ac[7] after a clock cycle.

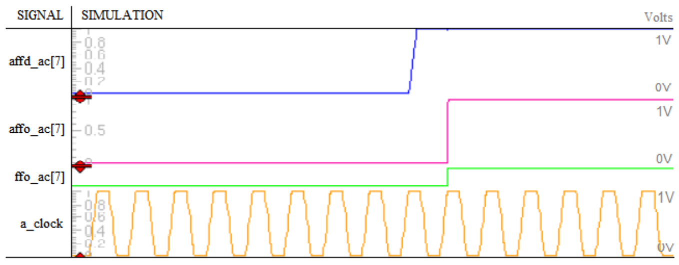

Fig. 4. Mixed-signal simulation validating the DFF and ADC executions.

VI. Conclusions and Future Work

Nonvolatile processors rely on nonvolatile memory cells that capture system states in parallel, making a quick recovery of the system context after power on. Thanks to its non-volatile factor, NVP is also energy efficient. In this work, a combinational implementation based on RTL was presented in order to complete the first stage for the implementation of the nonvolatile Neander. In addition, some concepts were presented to understand the proposal and some implementation techniques for combinational logic were discuss.

In the future, a NVFF cell will be validated using the Spice MTJ model from Harms et al. [9]. Once this validation has been made, it will be possible to make progress by placing the nonvolatile registers acting in parallel with the volatile registers. Then the rollback scheme will be implemented through checkpoints, making possible to implement the hypothetical nonvolatile Neander processor.

Acknowledgements

The authors would like to acknowledge the support of the National Council for Scientific and Technological Development (CNPq) and the State of Rio Grande do Sul Research Foundation (FAPERGS).

References

[1] T. Hanyu, T. Endoh, D. Suzuki, H. Koike, Y. Ma, N. Onizawa, M. Natsui, S. Ikeda, and H. Ohno, "Standby-power-free integrated circuits using mtj-based vlsi computing," *Proceedings of the IEEE*, vol. 104, no. 10, pp. 1844–1863, 2016. doi: 10.1109/jproc.2016.2574939

[2] F. Su, Z. Wang, J. Li, M.-F. Chang, and Y. Liu, "Design of nonvolatile processors and applications," in *2016 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*. IEEE, 2016. doi: 10.1109/vlsi-soc.2016.7753543 pp. 1–6.

[3] F. Su, K. Ma, X. Li, T. Wu, Y. Liu, and V. Narayanan, "Nonvolatile processors: Why is it trending?" in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE, 2017. doi: 10.23919/DATE.2017.7927131 pp. 966–971.

[4] Y. Wang, Y. Liu, S. Li, D. Zhang, B. Zhao, M.-F. Chiang, Y. Yan, B. Sai, and H. Yang, "A 3us wake-up time nonvolatile processor based on ferroelectric flip-flops," in *2012 Proceedings of the ESSCIRC (ESSCIRC)*. IEEE, 2012. doi: 10.1109/esscirc.2012.6341281 pp. 149–152.

[5] H. Koike, T. Ohsawa, S. Ikeda, T. Hanyu, H. Ohno, T. Endoh, N. Sakimura, R. Nebashi, Y. Tsuji, A. Morioka *et al.*, "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, 2013. doi: 10.1109/asscc.2013.6691046 pp. 317–320.

[6] R. F. Weber, *Fundamentos de arquitetura de computadores*. Sagra Luzzatto, 2000.

[7] F. R. Schneider, R. E. Poli, D. Barden, D. K. Leonel, D. A. Fiorentin, F. M. Trindade, F. S. de Vasconcellos, M. C. d. B. O. Neto, and R. P. Ribas, "Design of a 4-bit processor for evaluating of the e/d nmos technology from ccs/unicamp," in *20th Microelectronics Students Forum*, 2004.

[8] W. Zhao, E. Belhaire, C. Chappert, B. Dieny, and G. Prenat, "Tas-mram-based low-power high-speed runtime reconfiguration (rtr) fpga," *TRETS*, vol. 2, 06 2009. doi: 10.1145/1534916.1534918

[9] J. D. Harms, F. Ebrahimi, X. Yao, and J.-P. Wang, "Spice macromodel of spin-torque-transfer-operated magnetic tunnel junctions," *IEEE transactions on electron devices*, vol. 57, no. 6, pp. 1425–1430, 2010. doi: doi:10.1109/ted.2010.2047073

[10] R. M. Brum, "On the design of hybrid cmos and magnetic memories, with applications to reconfigurable architectures," Ph.D. dissertation, University of Montpellier, Montpellier, France, 2014.

[11] L.-B. Faber, W. Zhao, J.-O. Klein, T. Devolder, and C. Chappert, "Dynamic compact model of spin-transfer torque based magnetic tunnel junction (mtj)," in *2009 4th International Conference on Design & Technology of Integrated Systems in Nanoscal Era*. IEEE, 2009. doi: 10.1109/dtis.2009.4938040 pp. 130–135.

[12] T. Na, K. Ryu, J. Kim, S. H. Kang, and S.-O. Jung, "A comparative study of stt-mtj based non-volatile flip-flops," in *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*. IEEE, 2013. doi: 10.1109/iscas.2013.6571794 pp. 109–112.

[13] Y. Shang, W. Fei, and H. Yu, "Analysis and modeling of internal state variables for dynamic effects of nonvolatile memory devices," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 59, no. 9, pp. 1906–1918, 2012. doi: 10.1109/tcsi.2011.2180441

[14] M. Kazemi, E. Ipek, and E. G. Friedman, "Energy-efficient nonvolatile flip-flop with subnanosecond data backup time for fine-grain power gating," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 62, no. 12, pp. 1154–1158, 2015. doi: 10.1109/tcsii.2015.2468931