Networked Power-Gated MRAMs for Memory-Based Computing

Jean-Philippe Diguet, Naoya Onizawa, Mostafa Rizk, Johanna Sepúlveda, Amer Baghdadi, Takahiro Hanyu

Networked Power-Gated MRAMs for Memory-Based Computing

Jean-Philippe Diguet, *Member, IEEE,* Naoya Onizawa *Member, IEEE,* Mostafa Rizk, Johanna Sepulveda *Member, IEEE*, Amer Baghdadi *Senior Member, IEEE,* and Takahiro Hanyu *Senior Member, IEEE*

***Abstract*—Emerging non-volatile memory technologies open new perspectives for original computing architectures. In this paper, we propose a new type of flexible and energy-efficient architecture that relies on power-gated distributed Magnetoresistive Random-Access Memory (MRAM). The proposed architecture uses a Network-on-Chip (NoC) to interconnect MRAM-based clusters, processing elements, and managers. The NoC distributes application-specific commands to MRAM devices by means of packets. Configurable Network Interfaces (NI) allow to transform MRAM devices into smart units able to respond to incoming commands. In this context, three types of MRAM designs are proposed with different power-gating policies and granularities. A relevant database search engine case study is considered to illustrate the benefits of this proposed architecture. It is implemented with a Sparse-Neural-Network (SNN) approach and simulated in SystemC with different scenarios including hundreds of database queries. Hardware designs and accurate power estimations have been conducted. The obtained results demonstrate important power reduction with database hit rates of about 94%. Targeting 65nm technology, energy savings reach 87% when compared with an SRAM-based implementation. Moreover, a new asymmetric read/write MRAM type provides from 39% to 50% energy reduction with respect to the other fixed-granularity models. This results in a low-power, highly scalable and configurable implementation of memory-based computing.**

***Index Terms*—MRAM, NOC, Power-gating, Database, Sparse-neural networks.**

I. Introduction

COMPUTING resources are continuously increasing with technology progress, yet the use of the computing capabilities is facing major limitations in terms of power dissipation, off-chip memory access time, and real available parallelism. These limitations lead to a partial use of the offered integration capabilities and increase the gap between accessible technology nodes and available design methodologies. Nevertheless, the evolution in memory design and on-chip memory integration, particularly the emerging non-volatile memory (NVM) technologies, open new design perspectives to balance the under-use of available hardware resources.

In fact, besides memory access time reduction due to on-chip integration, the silicon density offered by recent technology nodes can be efficiently used and significant reductions in power consumption can be achieved as long as the leakage power can be controlled and the memory bandwidth exploited. Such a configuration is made possible by applying memory-based computing (MBC), while considering a network-on-chip (NoC) to broadcast requests to NVM devices and full power-gating to switch-off unused memories.

Based on these observations, we first propose a novel NoC-Memory Based Computing (NMBC) architecture which relies on a NoC-based multicore architecture that allows to take full benefit from power-gating capabilities of distributed MRAM. We introduce a new computing paradigm, where the packets of a NoC are used as commands executed by configurable Network Interfaces (NI) to access multiple distributed memory blocks in parallel. These memory blocks are technologically independent from the NoC and the processing units. Furthermore, we introduce different implementations of Spin-Transfer-Torque Magnetoresistive Random-Access Memory (STT-MRAM) with cell and peripheral power-gating capabilities. Finally, the effectiveness of the proposed approach is demonstrated through a relevant case study of a database search application implemented with neuromorphic architecture based on Sparse-Neural-Network (SNN) [1].

This paper presents five new contributions that improve significantly the performances compared to our previous work [2]: i) an original type of power-gated MRAM memory called *Type III*. It is based on an asymmetric read/write scheme and allows for significant power reductions with respect to previous solutions; ii) rather than a single manager, multiple manager modules can be working concurrently; iii) the processing elements are not statically but dynamically allocated; iv) processing elements are implemented as dedicated hardware components and they are included in the global power budget; v) design and synthesis of all modules, including MRAM devices, are done using 65nm technology node.

The rest of the paper is organized as follows. First we present the proposed NMBC architecture model and the considered application case study of SNN-based database search engine in Section III. Then, Section IV introduces the MRAM design models including the proposed cell and peripheral power-gating policies and the three types of power-gating granularity. Section V provides the detailed evaluation setup in terms of selected database for the application case study, NoC implementation, and SystemC simulation of the proposed NMBC architecture. Section VII summarizes and discusses the obtained results before concluding the paper in Section VIII.

J. Diguet is with CNRS, Lab-STICC UMR 6285, Lorient, France; M. Rizk is with Lab-STICC and the Lebanese International University, Beirut, Lebanon; A. Baghdadi is with Lab-STICC and IMT Atlantique, Brest, France; J. Sepulveda is with TUM, Munich, Germany; N. Onizawa and T. Hanyu are with Tohoku University, Sendai, Japan

II. Related work

A. Memory-based computing

MBC can have several meanings. The first one is a method to solve the memory-wall challenge, for applications dominated by data transfers, by migrating computing within the memory. A first implementation of full adder was introduced in [3] and a summary of a research decade on this topic is presented in [4]. The solution consists in executing some greedy operations with logic implemented in the memory device in order to reduce the bandwidth demand and latency that limit the use SRAM and DRAM. The logic-in-memory concept allow to implement logic function (F) to apply transformations and data transfer in the memory as presented in Fig.1-b). This solution is especially efficient with NVM technologies such as spin-orbit torque magnetic random access memory (SOT-RAM) or Domain Wall Motion (DWM) memory. In [5] the authors compare these technologies with a AES encryption application, which is a typical in-place computing scheme that can take full benefit from this approach. The idea of moving computing close to memory to reduce latency and increase the bandwidth compare to solution based on SRAM or DRAM. Memristive technologies are also candidate for programmable approaches. This is presented with programmable bit-level logical function in memories in [6], but power-gating is not considered.

Our approach aims also to distribute and move computing resources closer to memory but with a different approach. Actually we consider distributed memories to increase the bandwidth. This solution is made possible jointly by the increase of on-chip memory density and by the emerging manycore architectures based on NoC. For cost and efficiency reasons, we also want to reuse existing (or emerging) NVM memories without modification except the implementation of efficient power-gating techniques. So our solution relies on the enhancement of NI as illustrated in Fig.1-a).

The second meaning of MBC is the idea that computing can be replaced by memory. Replacing logic by memories is efficient in case of computation redundancy. Such patterns can be extracted from signal processing algorithms that allow partial pre-processing as introduced in [7]. In [8] the authors consider dense memory, based on molecular crossbars, to implement logic functions in multi-input-multi-output LUT in a memory array. In this approach, a partitioning of the application graph is applied to map computing elements on memory arrays. In [9] the idea is extended to coarse-grain implementations. Both cases are illustrated in Fig.1-c) and d) respectively. It worth noting that the two ideas can be combined if in-memory logic functions are implemented as LUTs.

Our solution also aims to replace computing by memory like Fig.1-c), however we do not implement logic in the memories but in the NI to turn power-gated memories into content-addressable memories. It is also important to note that our NMBC concept can be generalized and additional functionality (e.g. encryption, max, sum, sigmoid, etc.) can be implemented in the NIs.

The third meaning is an extension of the second one and is based on the principle of memory cache applied to the results of frequent computations with identical inputs. It is illustrated in Fig.1-e). If the same computation with the same operands occurs then the result is available from the memory without repeating the computation. Such methods have shown significant energy improvement in application domains with important opportunities of result reuse. This technique has been implemented with associative memristive memories (AMM) in the context of heterogeneous reconfigurable multicore [9] and GPGPU architectures [10].

Our approach considers the replacement of results stored in memory to take advantage of dense, low power memories. In practice, all memories are not addressed at the same time leading to inherently partial resource usage in time. Therefore, applying power-gating techniques to unused resources can offer important power savings. In this context, emergent NVM technologies such as MRAM with low-leakage characteristics and power-gating capabilities represent a great opportunity. They allow to design energy efficient architectures where unused memory resources can be switched off. In other words, increasing on-chip resources with low-power behaviors contribute to the global architecture and energy efficiency. Moreover, such architectures can now take real advantage of more than one decade of research and achievements in NoC design. NoC allows configurable accesses to grid of memories as well as the implementation of pre/post-processing steps in NI [11].

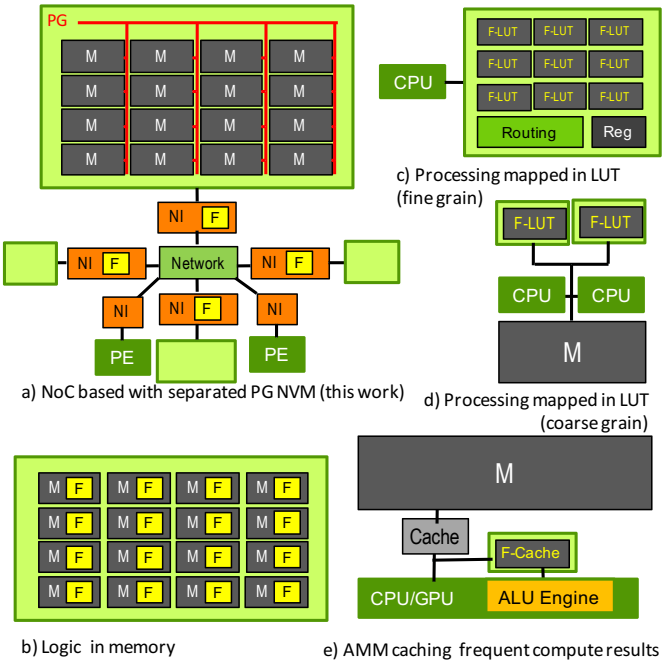


Figure 1: Types of memory-based computing

Data-centers related applications represent a typical case where the energy efficiency constraints lead to the rise of heterogeneous architectures [12]. It covers applications dominated by memory resources such as database accesses, search engines, data analytics and so on. In that context, neuromor-

phic architectures can play a major role. It is worth noting here that this class of applications usually implies much more reading than writing accesses in the operational phase after the learning stage. Different approaches have been explored in the literature. In [13], the authors present an efficient run-time configurable Coarse Grain Reconfigurable Architecture (CGRA) for convolutional neural networks (CNN). A smart DMA and dedicated switches are used to feed computing nodes with data from an off-chip memory. The architecture is computation intensive and doesn't fit with memory-based computing except if produced data exhibits reuse. The True North architecture [14] is different and designed for Spiking Neural Networks. It is based on a NoC to distribute synapses states to local computing cells. The chip is based on a dedicated architecture that implements 5 Mb of on-chip memory but with no power-gating technique. Logic-in-memory is also explored for neural network (NN) applications, in [15] the authors implement the different NN function including Sigmoid in a NVM. Finally a Sparse Neural Networks (SNN) dedicated architecture is presented in [16]. It is designed using a logic-in-memory approach and MTJ devices, with bit-level interconnects. It is highly optimized but at the cost of a very specific design approach. A NoC-based architecture for SNN is presented in [17], yet it is a dedicated architecture and the paper is mainly devoted to the NoC topology and the memory optimisation.

Finally, one can think about the concept of active memory, which also enhances memory with new functionality. This is the capacity to manage high-level transactions requests with a local processor executing transfer operations [18]. This type of solution leads to specific designs, where logic and memory are tightly coupled.

We also consider the case of neuromorphic computation as a typical case of data reuse that can take a great benefit of a MBC model. But as previously mentioned, we consider MRAM with efficient power gating and implement additional logic resources in the NI and not in the memories. Our objective is to exploit the bandwidth and the flexibility of packet-switching NoC architecture, to efficiently access the target data distributed in arrays of memories that can be switched on when necessary.

B. Power-gated MRAM devices

Magnetic tunnel junction (MTJ) devices [19] represent a main class of emerging nonvolatile technologies for low-power memory circuits. Recently, several MRAM designs have been presented [20], [21], [22] that exhibit significantly low leakage currents realized by power-gating, thanks to the non-volatility feature. These MRAM designs are designed using relatively large device technologies, such as 90 nm, yet the MTJ technology allow scaling down to 11 nm as for CMOS devices [23]. The performance of MRAM devices in such recent technology nodes are also estimated in [24], [25].

MRAM devices are designed using different MRAM cells, such as 1T-1MTJ, 2T-2MTJ, and 4T-2MTJ. This corresponds to performance trade-offs between area efficiency and read time. For example, the 1T-1MTJ MRAM exhibits the highest area efficiency with a relatively slow read time, while the 4T-2MTJ MRAM realizes faster read time with lower area efficiency. In terms of power-gating, a simple strategy that shutdowns the power of the whole MRAM device at the idle state is generally used. In the 4T-2MTJ MRAM [21], a cell-level power-gating technique is used that activates only a row at the read/write state, leading to low leakage currents in the cells without any additional power-gating transistor. We use this 4T-2MTJ MRAM model in our work, more details are provided in Section IV.

However, advanced power-gating policies and granularities including peripheral circuits (e.g. row and column decoders) have not been presented or discussed in the available literature. It is worth noting that conventional power-gating techniques that can be applied to processing units and NoCs are out of the scope of this paper. These methods are independent and complementary to our proposed approach and can be efficiently combined.

III. NMBC Flexible Architecture

A. Architecture principle

The proposed architecture model is presented in Fig.2(a). It is based on a NoC that interconnects three types of IP blocks: memory clusters, processing elements and managers. In the proposed approach we consider NoC packets as commands initiated by managers as depicted in Fig.2(b). Memories are distributed in M_C clusters, each cluster has a unique NI and is composed of M_B memory blocks. The managers are in charge of a set of requests to process and send packets to memory clusters in a uni-, multi- or broad-cast way. They also apply a processor selection for post-processing and can so perform load balancing. All the managers can work in parallel. The decoding of instructions is carried out by smart NIs that implement additional logic elements . NIs are in particular aware of a data mapping and manage the communication of results to processors. NIs also implement some specific bit-level operations such as bit selection that will be detailed in section V-B3. NIs can also provide managers with monitoring data about bandwidth and processor usage, this type of information is sent as monitoring packets to managers.

B. Application case study of SNN-based database

1) SNN concept: As introduced before, database search engine is one of our target application domain. Hereafter, we explain the use of NMBC for database search engine implementation based on the SNN algorithm [1]. SNN is a neural network model inspired by information theory and error correcting codes. It is especially efficient for the implementation of associative memories, so it can be used to process database queries as shown in [16]. SNN relies on two phases: message learning and information retrieval from partial knowledge.

It is worth noting that the learning phase consists simply in writing each new record "one time" at the right place in the memories. Therefore, this phase presents a linear complexity with the database size for any devised network architecture. This is fundamentally different from the the greedy learning phase of deep learning approaches. So, the Write operations

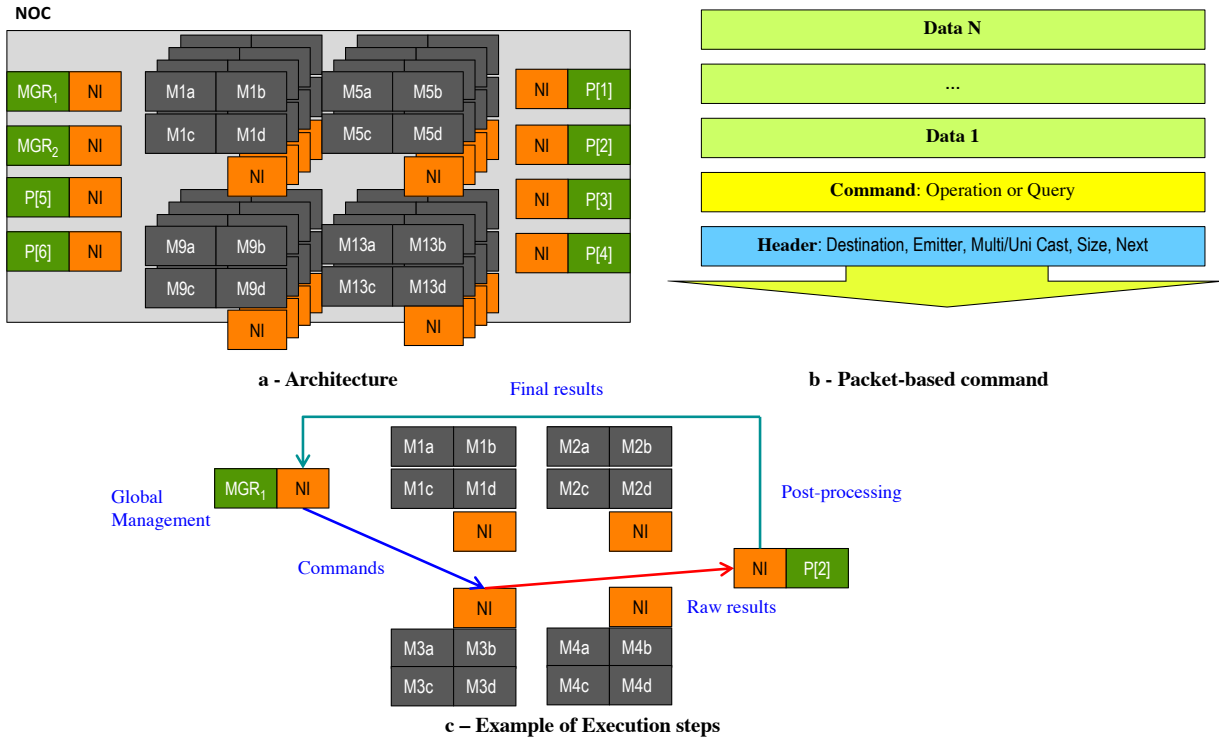


Figure 2: NMBC architecture example with M_C = 16 memory clusters composed of M_B = 4 memory modules (Mij), 6 processing elements ($P[x]$), and 2 managers (MGR_i)

cost is negligible compared to Read operations cost which constitutes the major challenge in database search applications.

Basically, a record is a clique that fully links neurons that belong to different clusters. The principle can be compared to low-density parity-check codes (LDPC). If we consider the database application, a record M is composed of F fields (a field $f(i)$ per cluster). The learning phase is the mapping function that associates each field of each record to a single neuron in each cluster. Thus, each field $f(i)$ is coded with $k_i = \lceil log_2(l_i) \rceil$ bits where l_i is the number of neurons in cluster $f(i)$. So, a record is a binary sequence of $\sum_{i=1}^{F} k_i$ bits and is associated to a clique of neurons, each clique materializes a codeword. The information retrieval or query consists in the selection of the best candidate neuron for each cluster that corresponds to an unknown field by means of known neurons is other fields. This choice is made according to a ranking method, which is a winner-take-all algorithm in the original proposal. All clique edges are specified with binary weights such as $C_{XY}(ij) = 1$ means that at least one clique contains an edge from neuron i in cluster X to neuron j in cluster Y. The score of each neuron, in the target cluster, equals to the number of clique edges that exist between this neuron and known neurons in other clusters. In the original algorithm, additional iterations can be executed if multiple winners have the same score. However, in the case of database search engine we will consider multiple answers to a given query. Other algorithmic options are possible but are out of the scope of the paper.

2) SNN on NMBC architecture: The very dominant cost in SNN implementation is memory. Actually it is necessary to store all the connections between every pair of clusters. We call $m_{i,j}$ the connection memory between clusters i and j. We demonstrated in another application domain that it is possible, by means of a dedicated memory mapping, to use a single memory for implementing $m_{i,j}$ and $m_{j,i}$ [17]. However in this work, we consider distinct $m_{i,j}$ and $m_{j,i}$ for two reasons. First, the NMBC architecture must be programmable to match with different configurations so the size of memory cannot fit exactly with the size of the cluster. Secondly, our objective is to take advantage of future MRAM memory with an low static power consumption so that unused memories do not impact significantly the power consumption. Therefore memories are used to store the $F \cdot (F-1)/2$ connection memories. Regarding the number of fields and the size of each cluster, the mapping of $m_{i,j}$ on the set of distributed memories results from an optimisation process to maximise the use of each memory. The managers send queries to memories by means of packets where known and unknown fields are specified. Then the processors collect the memories answers and apply a simple winner-take-all algorithm to select records to be send back to managers.

IV. MRAM MODEL

A. 4T-2MTJ MRAM cell

MRAM circuits are designed using spin-transfer-torque (STT) magnetic tunnel junction (MTJ) devices [19] with CMOS transistors. Fig.3(a) shows an STT MTJ device stacked over a CMOS layer. It mainly consists of three layers: free layer, tunnel barrier, and fixed layer. The resistance of the

Table I: Power-gating policy in MRAMs

Policy	Only cell power-gating (OCPG)	Full power-gating (FPG)
PG in cell array	Yes	Yes
PG in peripheral	No	Yes
Wake-up operation	No	Yes

MTJ device depends on a spin direction of the free layer. If the spin direction of the fixed layer is the same as that of the free layer, the MTJ is at the parallel state, R_P (low resistance). Otherwise, it is the anti-parallel state, R_{AP} (high resistance). The resistance state can be changed using a current signal, I_{MTJ}, as shown in Fig.3(b).

Fig.4(a) shows a 4T-2MTJ MRAM cell with power-gating (PG). The MRAM designed based on [21] consists of four transistors and two MTJ devices controlled by the power line (PL), the word line (WL), and the bit lines (BL, $\overline{\text{BL}}$). A data bit is represented by complementary signals, b and $\overline{b}$, stored in the two MTJ devices. Fig. 4 (b) shows a timing diagram of the MRAM cell. In the write operation, a data bit is stored to two MTJ devices for two steps. First, a current signal is generated from PL to one of bit lines (BL, $\overline{\text{BL}}$) depending on a data bit stored. In this example, $\overline{b}$ is written because $\overline{\text{BL}}$ is low. Then, b is written using a current signal from BL to PL, where the resistance is changed from R_P to R_{AP}. In the read operation, one of BL and $\overline{\text{BL}}$ are low depending on a stored bit. In the idle state (PG), both PL and WL are grounded which totally eliminates the static power in the 4T-2MTJ cell as no leakage path exists in this case.

B. Power-gating policy

Fig. 5 shows an MRAM with cell PG and selective peripheral PG. It consists of the MRAM-cell array with the peripheral circuits: row decoder, row drivers, column decoder, and sense amplifiers (S/A). In the MRAM-cell array, only MRAM cells in a row are active in read/write operation, while the other MRAM cells are power-gated using the row drivers. The cell PG significantly reduces the leakage current of the cell array in comparison with that of an SRAM cell array. The peripheral circuits can be power-gated using the PMOS transistor when the MRAM is at the idle state.

For the PG policy, two different policies of PG are exploited as summarized in Table I. The 1st policy is *only cell power-gating (OCPG)*, where the MRAM cells are power-gated at

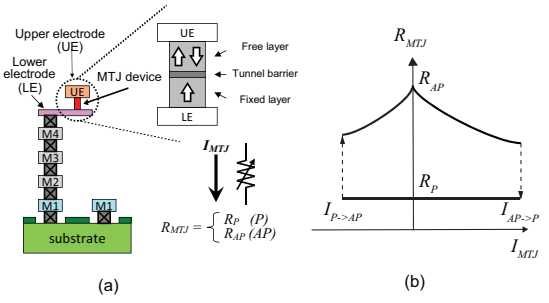


Figure 3: Spin-transfer-torque (STT) magnetic tunnel junction (MTJ): (a) structure and (b) R-I characteristic

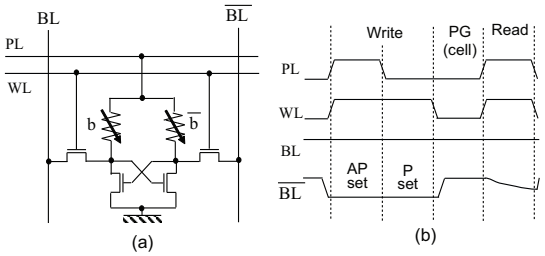


Figure 4: 4T-2MTJ MRAM cell with power-gating (PG): (a) circuit diagram and (b) timing diagram.

Table II: Performances of 256×256 MRAMs using TSMC 65nm CMOS process and MTJ model [26].

	Type I	Type II	Type III
Write bit width	256	128, 256	32
Read bit width	256	128, 256	32, 64, 128, 256
# of PG switches	1	4	1
Normalized PG switch size	1	1	0.5
Normalized total PG switch size	1	2	0.5
Read power/bit@100 MHz [mW]	1.30	1.16	1.03
Write power/bit@ 100 MHz [mW]	2.79	2.48	2.38
Static power w/o PG [mW]	51.3	62.2	43.2
Static power w/ PG [mW]	0.679	0.980	0.300
Wakeup energy [nJ]	0.934	1.013	0.648
Wakeup time [ns]	0.072	0.0045	0.072

the idle state and the peripheral circuits are always at the active state. Therefore, the OCPG-MRAM requires no wake-up operation with negligible leakage current in the MRAM cell array The 2nd policy is *full power-gating (FPG)*, where the whole MRAM is power-gated at the idle state. It reduces the leakage current in comparison with the OCPG-MRAM while the wake-up operation is required for the peripheral circuits. The wake-up operation causes extra energy dissipation and delay time. Hence, there is a trade-off between the OCPG-MRAM and FPG-MRAM. In the next Section, the performance of the two different MRAMs with the two different PG policies is exploited for system-level simulations of NMBA.

C. Power-gating granularity

For the PG granularity, three different 256-bit × 256-word MRAMs (Type I, Type II and Type III) are designed as shown in Fig. 6. The detailed specifications and peformance are summarized in Table II. The MRAM circuits are designed and evaluated using TSMC 65nm CMOS process and MTJ model [26] by HSPICE.

First, let us explain the differences between Type I and Type II. Type I uses a PG transistor for a 256×256 MRAM. Type II uses four PG transistors for four 128×128 MRAM subblocks. The widths of the PG transistors are determined so as to achieve similar write and read times. Compared with Type I, the advantage of Type II is lower power dissipation of reading/writing, if the read/write bitwidths are smaller than or equal to 128 bits. In fact, read and write power are reduced because parastic capacitances of BLs and WLs are reduced, as these capacitances depend on the sizes of the MRAM subblocks. In contrast, the disadvantage of Type II is larger leakage current as the normalized total PG switch size required

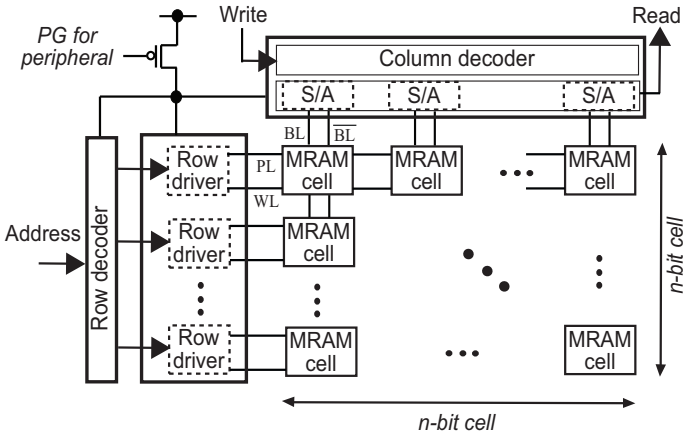


Figure 5: MRAM with cell PG and selective peripheral PG

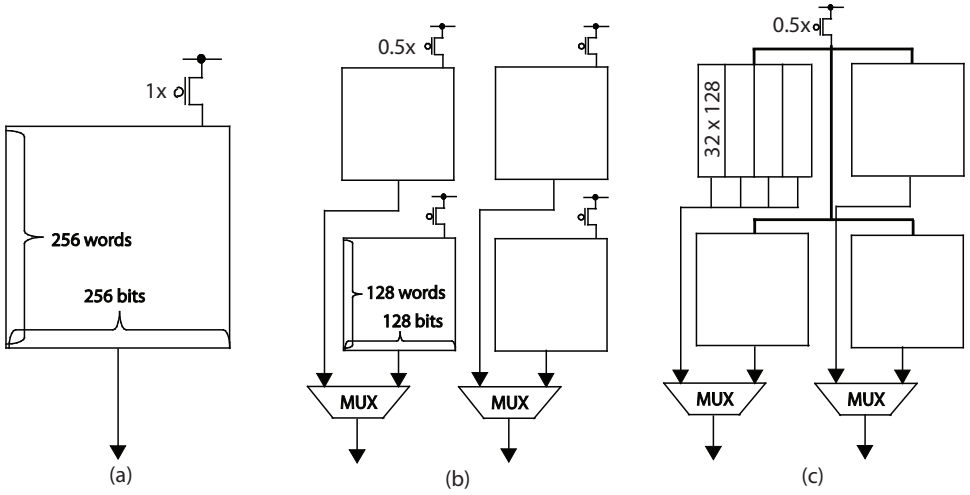


Figure 6: Power-gating types and granularity in 256-bit $\times$ 256-word MRAMs: (a) Type I, (b) Type II and (c) Type III.

is double. We observe that in MBC applications in general and search applications in particular, data are mainly read and rarely written. So we introduce Type III in order to meet both advantages of Type I and II. Type III is designed with the restriction that the maximum write bit widths is 32 bits, whereas the maximum read bit width is 256 bits as for Type I and II. Considering the asymmetry of read and write occurrences in the target application domain, this restriction does not significantly affect the execution time. As only 32 bits are written in parallel in Type III, the normalized total PG switch size required is 1/4 of the Type II, leading to a lower leakage current. If write capability is limited to 32bits, Type III offers on the other hand opportunities to extend read configurations to 32, 64, 128 or 256 bits and so to adapt the read power according to the application demand as shown in Table II.

V. Case Study

A. Yeast database case

As a representative example we consider the Yeast database (Cellular Localization in Proteins) from the UCI Machine Learning Repository [27]. The original database is specified with 10 fields that we transform into 11 fields by splitting the first oversized field so that we get a more homogenized network. Table III shows the number of connections per field, i.e. neurons per cluster. The number of available memories is fixed by the architecture, yet the number of connection memories $m_{i,j}$ depends on the application. Therefore, the first design step is the memory mapping, that results from an optimisation process, which is out of the scope of this paper. In our case study we map $10\times11 = 110$ connection memories in six 256×256-bit memory modules. Fig. 9 illustrates the proposed mapping for one of these six memories (M5, as referenced in Fig. 7).

B. NoC Implementation

Fig. 7 presents the structure of the NoC we consider to demonstrate the NMBC concept. The NoC is a 4×4 mesh-based network which interconnects 18 IP cores (6 memory clusters composed of single modules, 10 processing elements, and 2 managers). It means $M_C = 6$ and $M_B = 1$ if we refer to the general model described in Fig.2 The NoC employs the wormhole packet switching mode, the deterministic XY routing algorithm, and a flow control policy without virtual

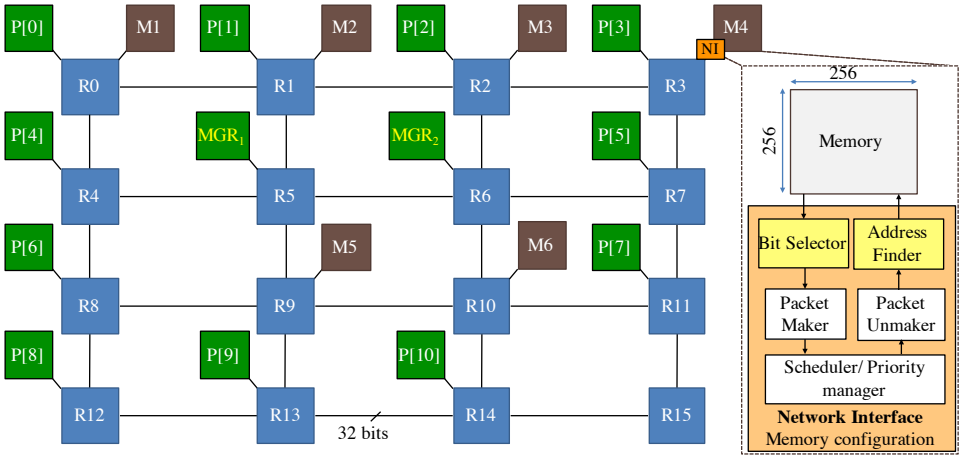


Figure 7: The structure of the used NoC

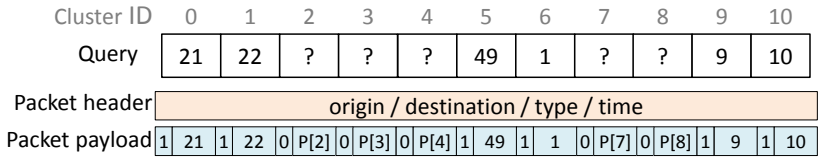


Figure 8: The structure of the command packet

channels. The implemented routers have one buffer of 3 flits per input port and use distributed arbitration logic (one arbiter per port). The back-end part of the NI is typical and includes a packet maker / un-maker, and a priority manager to synchronize packet transmission and reception.

In addition to usual interface logic, the front-end of a NI associated to a memory module, implements the addressing and bit selector modules. This choice is important since it allows to remain compliant with any existing memory and to be independent from NoC parameters (topology, router buffer depth and routing policy). In a fully flexible NMBC architecture, a NI includes an area of reconfigurable hardware that can be configured to implement application-specific functions like addressing and bit selector modules in our case study. The functionality of each IP core and the structure of the additional elements are detailed in the following paragraphs.

1) Manager: Upon loading new query (e.g. from a host machine), the manager associates missing fields to clusters with unknown neurons and well specified fields to identified neurons in other clusters. Consequently, it transmits to all memory modules the command to send to allocated PEs, the available connection information between known neurons in well defined clusters and all neurons of the undefined clusters. The payload of sent packets is divided into segments, which are relative to definite clusters.

In the adopted application, 11 different clusters exist. Each segment has two parts. The first is one-bit flag, which is used to indicate the status of the corresponding cluster. Respectively, "0" or "1" are placed to specify if the cluster neuron is missing or known. The content of the second part depends on the cluster status. If the cluster has known neuron (i.e. known field of the query), this part contains the initial address of the neuron which is available in the loaded query. Whereas, if the cluster's neuron is missing, it will be filled by the address of the processing element that is in charge of processing the winner take all for this cluster. Fig. 8 presents an example of the sent packet. The loaded query in this example have five clusters with missing neurons (2, 3, 4, 7, and 8). The flags corresponding to these clusters is set to "0". Also, the addresses of the processing modules (P[2], P[3], P[4], P[7], and P[8]) are specified in the second part of each segment. The choice of the PE is made dynamically with a fairness load-balancing policy. Whereas, for the other clusters (0, 1, 5, 6, 9, and 10), the flags are adjusted to "1" and the second part of each segment is loaded by the values initially available in the query. By arranging this data into 32-bit flits, the sent packet will be composed of 4 flits only. On the other hand, the manager is firstly responsible to collect the results corresponding to each cluster from its dedicated processing module. Then it decides to iterate in order to enhance results or to simply deliver the achieved ones.

2) Processing Elements (32 bits in parallel): All processing element (PE) apply the winner-take-all computational principle in order to find the neurons with the highest score. The score of each neuron is incremented by the number of available connections with all known neurons, which belong to the other clusters. Each PE is assigned to one cluster with a missing field, the assignment is made dynamically by the managers and the PE is active until the solutions are found. At last, each PE informs, by means of packets, the managers by the identity(ies) of the winner neuron(s).

In a fully flexible NMBC architecture, a PE could be a small-size embedded processor or a reconfigurable hardware. In this study we have designed a specific PE in VHDL so that we can estimate the power consumption and the area. Each PE can process 32bits (32 neurons) in parallel. It takes then 3 cycles to update in the local memory the score of the cluster neurons and the best score neuron. The PE receives the connection information related to their cluster from all memory modules. The total number of cycles depends on the number and size of known/unknown fields that determine the number of packets to be processed.

3) Memory module: The memory modules store the connection information between the neurons of all clusters at the learning phase. For the adopted case study, six 256×256-bits memory modules are used to accommodate the contents of 110 connection memories. Each module includes a memory that returns the data allocated at its specified address. Each memory module should respond to the reading request and deliver all connection information between all known neurons and missing ones to the specified processing modules. The manager is not aware of the local mapping of connection memories and the model is independent from the type of memories. Therefore, the matching between requested connections and physical addresses is performed by NIs. Since the command packet arriving from the manager does not include implicitly the specifications of the data which is required to be retrieved (address, width, and starting bit), new tasks such as addressing and arranging the output bits into flits are required. Independently from memory and NoC architectures, these new services are implemented in separate elements in the front-end of the NI associated to each memory module as shown in Fig. 7. Their architectures and functionalities are described below:

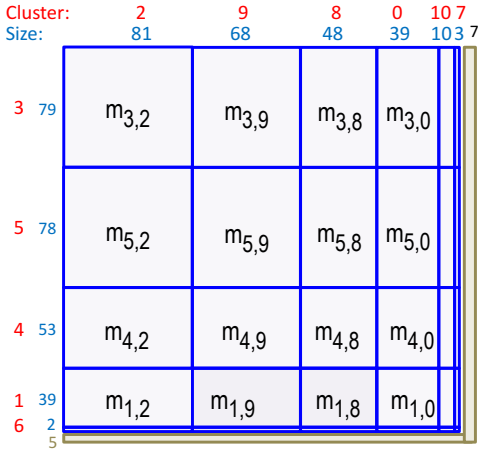


Figure 9: Yeast database: $m_{i,j}$ mapping in Memory M5

Table III: Number of neurons per cluster in Yeast database

Cluster	0	1	2	3	4	5	6	7	8	9	10
Neurons	39	39	81	79	53	78	2	3	48	68	10

Address finder: It is a simple hardware element, presented in Fig. 10, that determines the addresses that must be used to access the right connection memories. These addresses are based on the ID of unknown filed and the value of known fields indicated in the command packet. It is composed of an adder and a configurable look-up table which includes the starting addresses of all clusters in the associated memory. When a command packet arrives, the packet un-maker disassembles the packet and delivers the payload to the address finder. The latter identifies the clusters with known neurons according to their flag bits. For each of those, its identity is used directly as the look-up table index (address) to retrieve the starting address of the cluster. This address is considered as an offset and is used by the adder to determine the address value to be delivered to the memory. For example, in Fig. 8, cluster 5 has known neuron address 49. The starting address of cluster 5 in memory M_5 is 79 as shown in Figure. 9. Therefore, the address finder will deliver the value of $128 = 79 + 49$ to the memory.

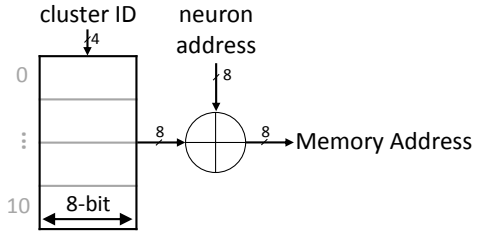


Figure 10: The architecture of the address finder

Bit selector: Fig. 11 illustrates the architecture of the bit selector. It is composed of one 256-bit barrel shifter, one 256-bit register to store input data from memory, three 32-bit registers to store output data according to a 32-bit flit size, and two look-up tables that deliver most significant bit (MSB) and control output data width. The first includes the MSB values corresponding to all clusters. The second controls the output data width. Both look-up tables are controlled by the cluster IDs. When a new data is retrieved from the memory, the barrel shifter shifts, for each cluster, the data by $255 - MSB$ steps and then delivers the result to the three output registers. The registers are then loaded according to the enable signal delivered by the second look-up table. The algorithm of the address finding and the bit selection algorithm is outlined in Algo. 1.

Algorithm 1 Address finding and bit selection

Require: k the number of clusters with known neurons,
m the number of clusters with missing neurons
for all $c \in \{0, ..., k-1\}$ **do**
$Address = offset_c + neuron_c$
$data = Memory[Address]$
for all $c \in \{1, ..., m-1\}$ **do**
$out = data << MSB_c$
$out0 = out[255:224]$
$out1 = En_c[0] \wedge out[223:192]$
$out2 = En_c[1] \wedge out[191:160]$
end for
end for

Finally, for all clusters with missing neurons, the retrieved connection information with known neurons is gathered ac-

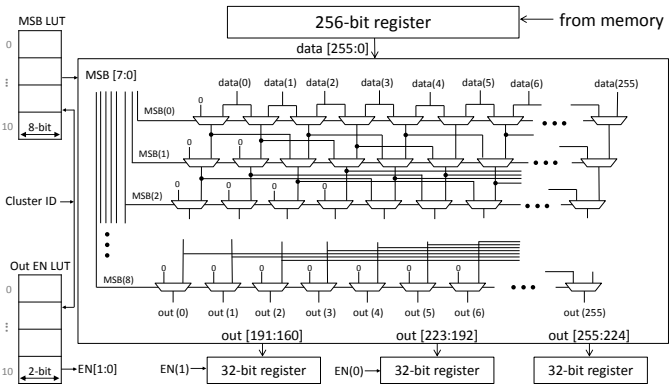


Figure 11: The architecture of the bit selector

cording to the cluster identities. Then, each group which is relative to a definite cluster is sent consecutively to the packet maker. The latter assembles a packet that is then transmitted to the processing module dedicated to each cluster.

VI. Experimental Setup

In order to check the relevance of the proposed approach, the adopted NoC implementing MRAM memories has been described in SystemC TLM model as a proof of concept. The devised model executes the database search engine application using the Yeast benchmarks. The model has been developed to mimic the exact behavior of the target architecture by designing hierarchical modules that work concurrently and intercommunicate via ports. In addition, the SytemC model is cycle accurate at the NoC level including NIs. To ensure accuracy, the timing features of memory reading response and wake ups are considered in the model. Furthermore, the execution time per paquet of winner-take-all computations are based on HDL simulation and synthesis that provide real-time environment modeling.

Multiple simulations have been conducted by using different numbers of queries with various number of missing clusters. In this study we consider four cases with 4, 5, 6 and 7 missing fields out of 11. The index of missing fields have been chosen randomly. To evaluate the efficiency of the proposed MRAM designs, the results are compared to those obtained when SRAM-based design is adopted. The SRAM on the iso-capacity condition is used. In both SRAM and MRAM cases, the total memory size is 384 kb, where six 256x256 distributed memories are used. Both models use identical NoC features, including packet sizes, routing algorithm, processing elements features, mapping strategy and a common clock set to 500 MHz. We consider the two power-gating policies defined in Sec.IV-B. FPG is implemented at the query level, it means that the peripherals are switched on when a packet is received and switched off when all the connection data, related to the command, are processed. We compare the three types of MRAM detailed in section IV-C. In the case of Type II and Type III memories, readings are limited to 128, 64 or 32 bits when it is possible, namely when all the requested data are located in one of the 4 128×128-bit memory block.

VII. Results and Analysis

A. Functional simulation

The data delivered by the manager is compared with the reference data from the Yeast database for the 4 cases with 600 random request experiment. As illustrated in Table IV, the obtained results meet our expectations with hit rates over 94% when adopting either one or two managers. The results show a slight degradation in the hit rate when increasing the number of missing fields. This is expected, and it is due to the fact that the ambiguity ratio increases with the decrease of the number of known fields. As indicated in Table V, the NoC injection rate doubles with two managers. This translates in an increase of the NoC activity and the time percentage FPG memories are ON.

Table IV: Hit rate

Number of missing fields	4 fields	5 fields	6 fields	7 fields
1 or 2 managers	94.83%	94.60%	94.22%	94.19%

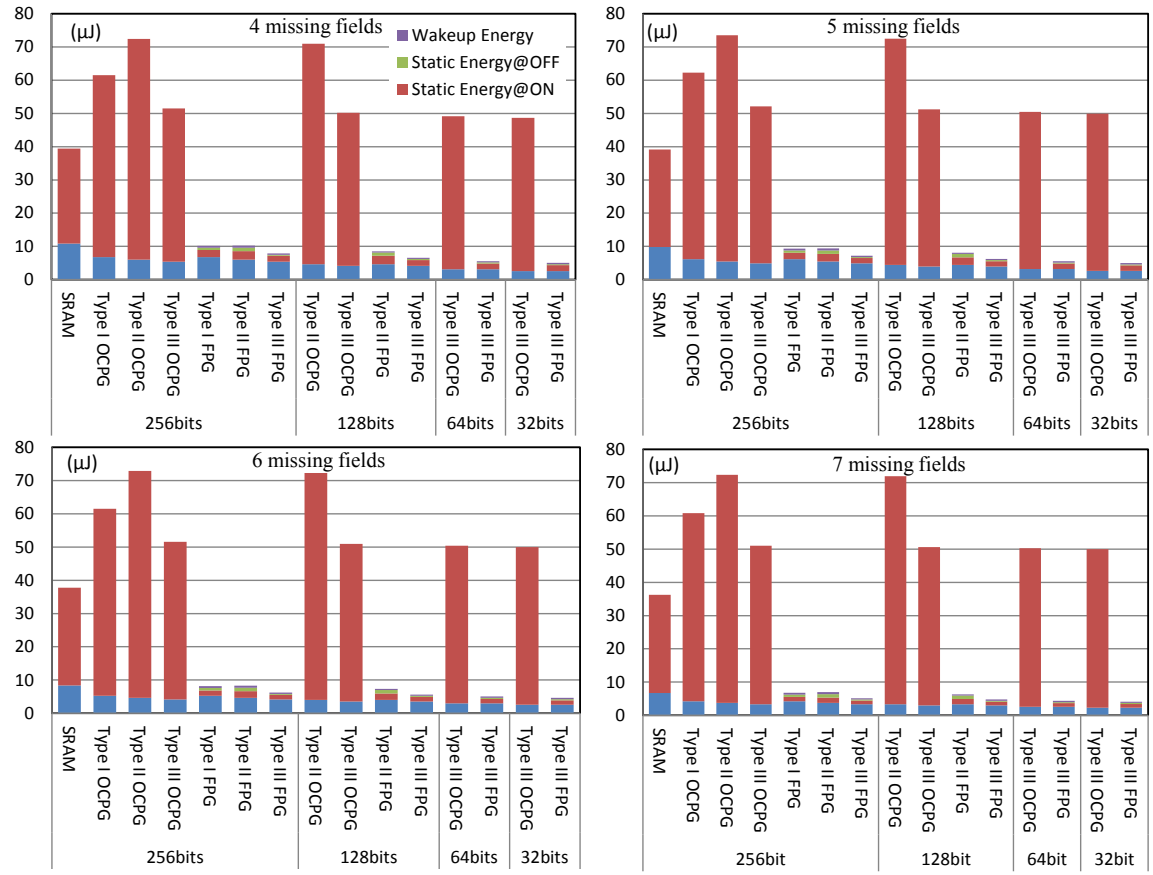
Table V: Injection rate (flits/clock cycle)

Number of missing fields	4 fields	5 fields	6 fields	7 fields
1 manager	0.482	0.524	0.552	0.567
2 managers	0.875	0.944	0.995	1.026

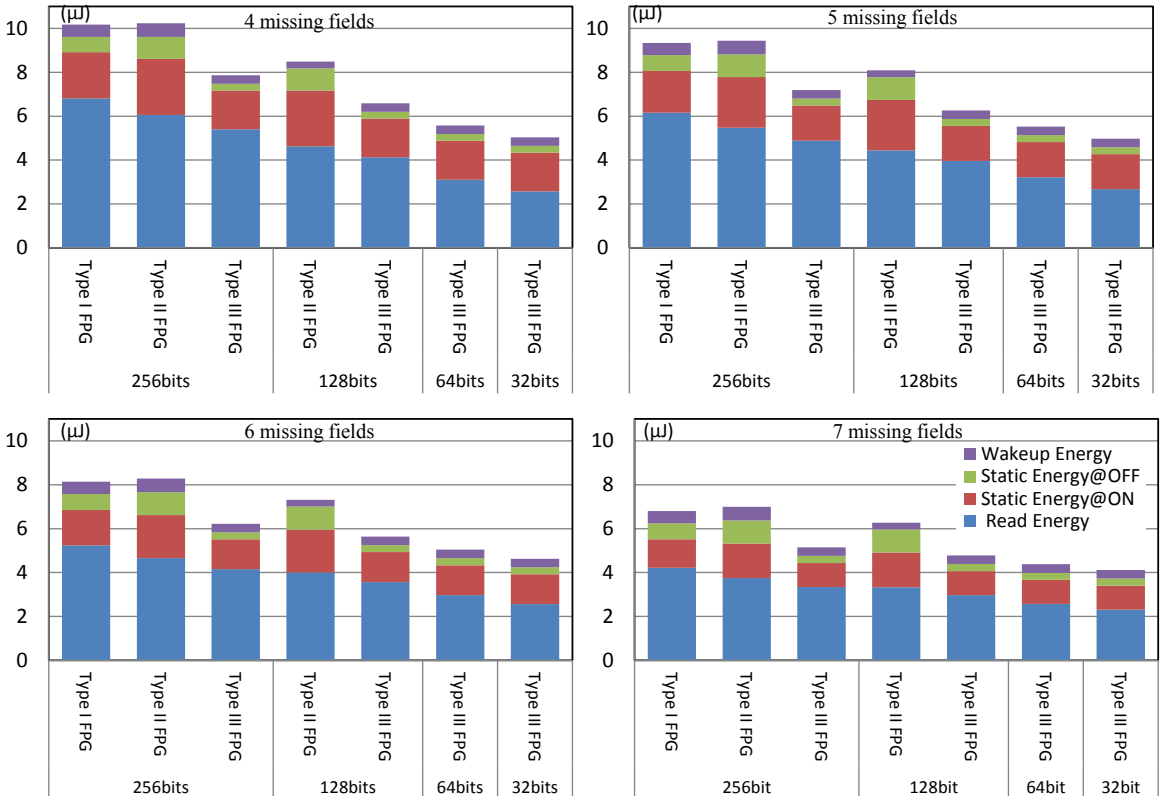
B. Power and Energy

1) Memories: Fig. 12 illustrates the energy dissipated by different types of memory modules in order to process 600 random queries while using the two proposed PG policies with 4, 5, 6 and 7 missing fields. The power estimation is based on the TSMC 65nm CMOS process and MTJ model given in Fig. II. From these results about the required energy budgets, we can draw several important conclusions:

- In 65nm, OCPG policy can not outperform SRAM case with any type of MRAM. The static power dissipation of OCPG-MRAM reaches 95% of the total power dissipation as shown in Table IX. Whereas in SRAM, the highest recorded percentage of static power dissipation is 81.5%. Hence, the power saving due to the reduction of dissipation while reading is not noteworthy although it is recorded as 1.6, 1.8, and 2 times less than that of SRAM for Type I, Type II, and Type III respectively. On the other hand, the static power dissipated in OCPG-MRAM is approximately 1.9, 2.3, and 1.6 times more than that of SRAM for Type I, Type II, and Type III respectively. This refers to the fact that in OCPG-MRAM only cells are power-gated at the idle state while the peripheral circuits are always at the ON state.
- All FPG solutions provide energy gains over 4 times with respect to SRAM. This refers to different factors. Firstly, the maximum recorded percentage of static power in FPG-MRAM is 42% from the total dissipated power which is about half of that recorded when SRAM is adopted. This can be explained by the significant reduction of static power in FPG-MRAM when the



a) 2 Managers - 600 requests, 4 configurations with 4, 5, 6 and 7 missing fields



b) 2 Managers - 600 requests, 4 configurations - Zoom on cases where $E < 10\mu J$ (all FPG policies)

Figure 12: Energy consumption for the different memory types and power-gating policies.

memory is in idle state and that the memories are at OFF state most of the time. Table VI shows the time percentage of ON state. The table shows that in FPG policy, the MRAMs are OFF more than 75% of the time. It is worthy to indicate that the static power dissipated in FPG-MRAM in idle state is approximately 39.5, 27.3, and 89.3 times less than that of SRAM for Type I, Type II, and Type III respectively. This fact demonstrates the reduced dissipation of static energy of FPG-MRAM compared to that of SRAM. Secondly,

Table VI: Time percentage of the ON state

Number of missing fields	4 fields	5 fields	6 fields	7 fields
1 manager	12.69%	11.27%	12.42%	7.61%
2 managers	23.03%	20.32%	17.21%	13.77%

the wake up feature in FPG policy has not added a significant overhead. The percentage of wake up consumed power is less than 10% as shown in Table VII. Thirdly, in FPG-MRAM, the dissipated power while reading is important since it dominates over the static power and wake up power. Its percentage approaches to 65% of total consumed power as shown in Table VIII. In addition, the SRAM power consumption while reading 256 bits is recorded about 1.6, 1.8, and 2.0 times more than that of MRAM for Type I, Type II, and Type III respectively. Whereas when 128 bits are retrieved, the SRAM reading power is 3.6 and 4.0 times more than that of MRAM for Type II and Type III respectively. This ratio increases significantly for the cases of reading 64 bits and 32 bits from MRAM Type III to reach 8.0 and 16.1 respectively.

Table VII: The percentage of power dissipated while waking up in FPG-MRAM from total power consumed in case of 7 missing fields and 2 managers

Memory Type	256-bit	128-bit	64-bit	32-bit
Type I FPG	8.24%	-	-	-
Type II FPG	8.86%	4.94%	-	-
Type III FPG	7.55%	8.14%	8.88%	9.45%

Table VIII: The percentage of power dissipated while reading in FPG-MRAM from total power consumed in case of 7 missing fields and 2 managers

Memory Type	256-bit	128-bit	64-bit	32-bit
Type I FPG	61.93%	-	-	-
Type II FPG	53.54%	53.13%	-	-
Type III FPG	64.92%	62.22%	58.75%	56.11%

If we look into details (Fig. 12-b) we observe the significant benefit of the proposed MRAM Type III:

- First we note that Type I outperforms Type II in case of 256 (about 2%). With 256 bits, the gain of read operations doesn't compensate the increase of the static power of the Type II switch.
- The balance becomes positive for Type II with respect to Type I when 128 accesses are possible. The application offers opportunities that can be exploited (from 7.7% to 16.5% for 7 and 4 missing fields respectively).
- Type III outperforms Type I and Type II in any case. Type III takes full benefit of the reduced read energy that doesn't impact the PG switch overhead thanks to the asymmetric scheme. Moreover Type III can also benefit from the bit-width granularity and efficiently exploit 32, 64, 128-bit cases opportunities.
- With the 32 bit Type III memory, the gains with respect to Type I go from 39.5% to 50.5% for 7 and 4 missing fields respectively, while the energy budget reduction is about 87% when compared with SRAM.
- The gains with respect to Type I and Type II decrease with the activity rate, this is logical since the PG impact is reduced as well. However it remains i) strongly significant with respect to Type I even with 7 unknown fields out of 11 and ii) extremely large with respect to SRAM in all cases. Therefore, we conclude that we obtain, as expected, important energy savings when the application fits with the MBC context namely when fully power-gated non volatile memories can take benefit of limited activity rates.

Table IX: The percentage of static power from total power consumed in case of 7 missing fields and 2 managers

Memory Type	256-bit	128-bit	64-bit	32-bit
SRAM	81.51%	-	-	-
Type I OCPG	93.07%	-	-	-
Type II OCPG	94.82%	95.37%	-	-
Type III OCPG	93.45%	94.13%	94.88%	95.38%
Type I FPG	29.83%	-	-	-
Type II FPG	37.60%	41.92%	-	-
Type III FPG	27.53%	29.65%	32.36%	34.44%

Table X: Average NoC power consumption (mW)

Number of missing fields	4 fields	5 fields	6 fields	7 fields
NoC (routers)	54.46	61.04	65.21	71.04
NoC (network interfaces)	38.17	42.78	45.7	49.79
Total	92.63	103.82	110.91	120.83

2) NoC: The NoC and NI clock frequency is 500 MHz in nominal case operating conditions (25 °C, 1V). The technology used is 65nm LP LowK Std Vt High Density Tapless Library from Virage Logic Corporation. The area and power results are obtained with the Cadence Encounter RTL Compiler tool. The RTL specification of the Router and the NI are synthesized in PLE (Physical Layout Estimation) mode. The total power was obtained by using the leakage and dynamic power of the NoC and NI components, relying on the switching activity traced by the SystemC simulation. Table X shows the static and dynamic power consumption of the used NoC. When compared to baseline NI, the area and power of the proposed NI, adopting the bit selector and address finder modules, present overheads of 14.3% and 12.6% respectively. If we consider the whole NoC, the area and power overheads are 6.1% and 5.2% respectively.

3) PEs: The power consumption of dedicated 32-bit PE is estimated with the following methodology. The RTL

Table XI: Global average power consumption - Worst case of 7 missing fields

Memory Average Power (mW)						
Memories types		**Static**	**Dynamic**	**Total**	**Ratio vs SRAM case**	**Ratio vs Best case** (32b Type III FPG)
256 bits	SRAM	160.80	36.49	**197.29**	1.00	8.82
	Type I OCPG	307.80	22.93	330.73	1.68	14.78
	Type II OCPG	373.20	20.38	393.58	1.99	17.59
	Type III OCPG	259.20	18.18	277.38	1.41	12.40
	Type I FPG	11.04	25.97	37.02	0.19	1.65
	Type II FPG	14.31	23.75	38.06	0.19	1.70
	Type III FPG	7.71	20.29	28.00	0.14	1.25
128 bits	Type II OCPG	373.20	18.14	391.34	1.98	17.49
	Type III OCPG	259.20	16.17	275.37	1.40	12.31
	Type II FPG	14.31	19.82	34.13	0.17	1.53
	Type III FPG	7.71	18.29	26.00	0.13	1.16
64 bits	Type III OCPG	259.20	13.99	273.19	1.38	12.21
	Type III FPG	7.71	16.11	23.81	0.12	1.06
32 bits	Type III OCPG	259.20	12.55	271.75	1.38	12.14
	Type III FPG	7.71	14.67	**22.38**	0.11	1
Total Average Power (mW)						
NoC		18.2	102.63	120.83	0.35	0.71
PE		4.11	22.60	26.71	0.08	0.16
Total (NoC + PE + SRAM)				344.83	1	2.03
Total (NoC + PE + 32b Type III FPG)				169.91	0.49	1

specification of PE are synthesized with the TSMC 65nm technology. The static and dynamic power consumption of a single PE running at 500 MHz are 0.32 and 3.79mW respectively. The activity of each PE is derived from the number of flits each PE is processing according to the simulation results. The power consumption of PEs are given in Table XI.

4) Global power budget: In this last section we analyse the power consumption of the whole NMBA considering a 65nm technology node. The whole architecture includes Memories, NoC (routers, NI with new modules) and PE (32-bit parallel version). In Table XI we consider the global power dissipation for seven missing fields, which is the worst case for MRAM versus SRAM since the idle time is reduced. All SRAM and MRAM-based architectures use the same NoC, the same SNN search algorithm and the same PE and managers. In addition to the significant power reduction, a main result lies in the major change of the power distribution. With SRAM memories, the NoC represents 35%, while the memories account for more than the half of the total power (57%). If the best MRAM type and PG policy are considered (Type III FPG), the NoC share rises to 71% and the memory one drops down to 13%. So, the use of the right power-gated MRAM leads to a situation where the NoC power consumption is higher than the memory one. This allows to increase the number of memory modules and so to target large-scale applications. In the considered case study, this number is limited to one 256×256-bit memory module. However, the general model includes multiple memory modules per cluster as depicted in Fig.2.

VIII. Conclusion

This study shows the high interest of a new type of architecture designed for memory-based computing that can fully benefit from distributed power-gated MRAMs. Memory blocks are efficiently accessed and controlled by means a NoC with smart NIs that are configured to implement application specific processing or content-addressable memory schemes. Although memory accesses dominate in such applications, the distributed nature of the proposed memory architecture exhibits idle states so opportunities for power gating.

Considering the general architecture model, the paper explores different power-gating schemes as well as memory block and R/W granularities.

The paper shows first that limiting the power-gating to MRAM cells does not reduce energy consumption with respect to the use of SRAM devices. Only a full power gating including peripherals is efficient and provide impressive energy reduction, higher than 87% in the considered database search application case study.

This significant energy reduction is obtained by considering the specificity of the application domain (database search applications, neuromorphic architectures), where the number of memory write accesses is extremely reduced compared to the number of read accesses. In this context, an original MRAM device model (Type III) is proposed with asymmetric write bitwidth (32 bits) and read bitwidth (32, 64, 128, 256 bits). It allows to jointly limit the leakage of the power-gate switch and the read/write power consumption. This new approach brings 39% to 50% more improvement compared to symmetric fixed-granularity model.

Furthermore, the validity of the proposed architecture is functionally demonstrated with an application that can take advantage of memory-based computing, namely a database search application implemented with a Sparse-Neural-Network approach. SystemC simulations have been conducted targeting hundreds of database queries with a high hit ratio of about 94%.

Finally, the results show that such an approach can significantly modify the power budget breakdown between NoC and memory. This allows to increase the number of memory modules and so to target large-scale applications.

Future work should address the reduction of NoC power

consumption and the use of programmable logic in NI to allow their dynamic configuration according to the application requirements.

ACKNOWLEDGMENT

This work was supported in Japan by JSPS KAKENHI Grant Numbers JP16H06300 and in France by the Region Bretagne CyAM project and by the MFC project of Future & Rupture IMT program.

REFERENCES

[1] V. Gripon and C. Berrou, "Sparse neural networks with large learning diversity," *IEEE Trans. on Neural Networks*, vol. 22, no. 7, Jul. 2011.

[2] M. Rizk, J. Diguet, N. Onizawa, A. Baghdadi, M. Sepulveda, Y. Akgul, V. Gripon, and T. Hanyu, "NoC-MRAM architecture for memory-based computing: database-search case study," in *15th IEEE Int. NEWCAS Conf.*, Strasbourg, France, June 2017.

[3] S. Matsunaga, J. Hayakawa, S. Ikeda, K. Miura, H. Hasegawa, T. Endoh, H. Ohno, and T. Hanyu, "Fabrication of a nonvolatile full adder based on logic-in-memory architecture using magnetic tunnel junctions," *Applied Physics Express*, vol. 1, no. 9, 2008. [Online]. Available: http://stacks.iop.org/1882-0786/1/i=9/a=091301

[4] 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," *Proc. of the IEEE*, vol. 104, no. 10, pp. 1844–1863, Oct 2016.

[5] D. Fan, S. Angizi, and Z. He, "In-memory computing with spintronic devices," in *2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, July 2017, pp. 683–688.

[6] P.-E. Gaillardon, L. Amarú, A. Siemon, E. Linn, R. Waser, A. Chattopadhyay, and G. De Micheli, "The programmable logic-in-memory (plim) computer," in *DATE*, March 2016. [Online]. Available: http://dl.acm.org/citation.cfm?id=2971808.2971907

[7] H. R. Lee, C. W. Jen, and C. M. Liu, "On the design automation of the memory-based VLSI architectures for FIR filters," *IEEE Trans. on Consumer Electronics*, vol. 39, no. 3, pp. 619–629, Jun. 1993.

[8] S. Paul and S. Bhunia, "A scalable memory-based reconfigurable computing framework for nanoscale crossbar," *IEEE Trans. on Nanotechnology*, vol. 11, no. 3, pp. 451–462, May 2012.

[9] K. Rahmani, P. Mishra, and S. Bhunia, "Memory-based computing for performance and energy improvement in multicore architectures," in *Great Lakes Symp. on VLSI*. ACM, 2012.

[10] A. Rahimi, A. Ghofrani, M. A. Lastras-Montano, K. T. Cheng, L. Benini, and R. K. Gupta, "Energy-efficient GPGPU architectures via collaborative compilation and memristive memory-based computing," in *51st DAC*, June 2014.

[11] K. Martin, M. Rizk, M.-J. Sepulveda, and J.-P. Diguet, "Notifying memories: a case-study on data-flow applications with NoC interfaces implementation," in *53rd DAC*, June 2016.

[12] J. Cong, M. Huang, D. Wu, and C. H. Yu, "Invited - heterogeneous datacenters: Options and opportunities," in *DAC*, 2016. [Online]. Available: http://doi.acm.org/10.1145/2897937.2905012

[13] P. H. Pham, D. Jelaca, C. Farabet, B. Martini, Y. LeCun, and E. Culurciello, "Neuflow: Dataflow vision processing system-on-a-chip," in *55th Int. Midwest Symp. on Circuits and Systems (MWSCAS)*, Aug. 2012.

[14] P. Merolla et al., "A million spiking-neuron integrated circuit with a scalable communication network & interface," *Science*, vol. 345, no. 6197, 2014.

[15] Y. Wang, H. Yu, L. Ni, G. B. Huang, M. Yan, C. Weng, W. Yang, and J. Zhao, "An energy-efficient nonvolatile in-memory computing architecture for extreme learning machine by domain-wall nanowire devices," *IEEE Trans. on Nanotechnology*, vol. 14, no. 6, pp. 998–1012, Nov 2015.

[16] H. Jarollahi et al., "A nonvolatile associative memory-based context-driven search engine using 90 nm CMOS/MTJ-Hybrid logic-in-memory architecture," *IEEE Jour. on Emerging and Selected Topics in Circuits and Systems*, vol. 4, no. 4, pp. 460–474, Dec. 2014.

[17] J.-P. Diguet, M.-J. Sepulveda, M. Strum, N. L. Griguer, and L. Caetano, "Scalable NoC-based architecture of neural coding for new efficient associative memories," in *Int. Conf. on HW/SW Codesign and System Synthesis (CODES+ISSS)*, Montreal, Oct. 2013.

[18] J. Yoo, S. Yoo, and K. Choi, "Active memory processor for network-on-chip-based architecture," *IEEE Trans. on Computers*, vol. 61, no. 5, May 2012.

[19] S. Ikeda et al., "A perpendicular-anisotropy CoFeB-MgO magnetic tunnel junction," *Nature Materials*, vol. 9, pp. 721 – 724, 2010.

[20] R. Takemura, T. Kawahara, K. Miura, H. Yamamoto, J. Hayakawa, N. Matsuzaki, K. Ono, M. Yamanouchi, K. Ito, H. Takahashi, S. Ikeda, H. Hasegawa, H. Matsuoka, and H. Ohno, "A 32-Mb SPRAM with 2T1R memory cell, localized bi-directional write driver and '1'/'0' dual-array equalized reference scheme," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 4, April 2010.

[21] T. Ohsawa et al., "A 1 Mb nonvolatile embedded memory using 4T2MTJ cell with 32 b fine-grained power gating scheme," *IEEE Jour. of Solid-State Circuits*, vol. 48, no. 6, Jun. 2013.

[22] C. Kim, K. Kwon, C. Park, S. Jang, and J. Choi, "A covalent-bonded cross-coupled current-mode sense amplifier for STT-MRAM with 1T1MTJ common source-line structure array," in *2015 IEEE International Solid-State Circuits Conference - (ISSCC) Digest of Technical Papers*, Feb 2015, pp. 1–3.

[23] H. Sato, E. C. I. Enobio, M. Yamanouchi, S. Ikeda, S. Fukami, S. Kanai, F. Matsukura, and H. Ohno, "Properties of magnetic tunnel junctions with a MgO/CoFeB/Ta/CoFeB/MgO recording structure down to junction diameter of 11 nm," *Applied Physics Letters*, vol. 105, no. 6, p. 062403, 2014.

[24] K. C. Chun, H. Zhao, J. D. Harms, T. H. Kim, J. P. Wang, and C. H. Kim, "A scaling roadmap and performance evaluation of in-plane and perpendicular MTJ based STT-MRAMs for high-density cache memory," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 2, pp. 598–610, Feb 2013.

[25] S. Peng, W. Kang, M. Wang, K. Cao, X. Zhao, L. Wang, Y. Zhang, Y. Zhang, Y. Zhou, K. L. Wang, and W. Zhao, "Interfacial perpendicular magnetic anisotropy in sub-20 nm tunnel junctions for large-capacity spin-transfer torque magnetic random-access memory," *IEEE Magnetics Letters*, vol. 8, pp. 1–5, 2017.

[26] N. Sakimura et al., "High-speed simulator including accurate mtj models for spintronics integrated circuit design," in *2012 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2012, pp. 1971–1974.

[27] "Ml repository," https://archive.ics.uci.edu/ml/.