Energy saving in a multi-context coarse grained reconfigurable array with non-volatile flip-flops

Aika Kamei, Takuya Kojima, Hideharu Amano,
Keio University, Yokohama, Japan
Email: wasmii@am.ics.keio.ac.jp
Daiki Yokoyama, Hisato Miyauchi, Kimiyoshi Usami,
Shibaura Institute of Technology, Tokyo, Japan
Keizo Hiraga, Kenta Suzuki, Kazuhiro Bessho
Sony Semiconductor Solutions Corporation, Atsugi, Japan

***Abstract*—In this study, a second-generation coarse-grained reconfigurable array with non-volatile flip-flops (NVFFs), known as the non-volatile cool mega array with multi-context (NVCMA/MC), is proposed. Similar to the previous NVCMA, verify-and-retriable NVFFs (VR-NVFFs) are provided for their configuration memory, constant memory, data memory, and instruction memory. The dedicated instructions for controlling the store, verify, and restore operations of the NVFFs are provided to the microcontroller in addition to power gating functions. Based on experience of the NVCMA, four hardware contexts are introduced to maintain the configuration data for four tasks, without the sacrifice of memory leakage. The array size is expanded, and pipeline registers are introduced to reduce the trade-off between the performance and power consumption. This study mainly focuses on the energy-saving effect of the VR-NVFFs and the multi-context facility of the NVCMA/MC, including the measurement of the break-even point. The evaluation of a real chip implemented with 40 nm MTJ/MOS hybrid process technology demonstrates that the store energy is reduced by 65% with the two-step store control of the VR-NVFFs. Moreover, applications that run intermittently for intervals as short as approximately 3 μs can benefit from the multi-context power gating.**

I. Introduction

The Internet of Things (IoT) and edge computing have attracted substantial attention in recent years, and the expectations of low-energy, high-performance, and flexible devices have been increasing. Coarse-grained reconfigurable arrays (CGRAs) are promising accelerators for such applications. Unlike field programmable gate arrays (FPGAs), which reconfigure themselves on bit-level with look-up tables (LUTs), they provide word-level reconfigurability. Therefore, the hardware overhead for the reconfiguration of CGRAs is much lower than that for FPGAs. This makes them nearly as highly energy efficient as application-specific integrated circuits (ASICs), while maintaining a certain degree of flexibility.

Many IoT applications require edge devices to operate intermittently. To maximize the battery life, such devices are controlled to be active during short operation times and remain in standby mode for the remainder of the time. However, leakage power is still exhausted, even during the standby period. This cannot be ignored when employing CGRAs, owing to the large array of processing elements (PEs) and memory cells for storing configuration information, although the same problem is more serious for FPGAs.

One practical solution is power gating (PG) [1], which is an effective power management technology that suppresses the leakage power of inactive circuits by shutting off the power grid with high-threshold, low-leakage transistors for power switches. PG is particularly useful for IoT applications that require a long time to sleep. However, the disadvantage of this technology is that the data that are stored in volatile memory, such as flip-flops (FFs) and SRAM, are lost when the power is turned off. The recovery of the lost data following wake-up requires many state transitions and significant overheads in terms of the execution time and energy, particularly when the CGRAs contain voluminous configuration data.

To address this issue, several reconfigurable devices that leverage emerging non-volatile memory (NVM) technology have been proposed [2], [3]. As the data that are stored in NVM are maintained during the sleep period, the leakage power in the idle state can be drastically suppressed.

In particular, spin-transfer torque magneto-resistive RAM (STT-MRAM), which uses a magnetic tunnel junction (MTJ) as its memory element, is a very promising NVM technology. It offers various advantages over other NV technologies, such as fast access speed, maximum rewritable times, and operating voltage compatibility with CMOS. However, MTJ NVM exhibits a drawback in that it requires a long time for a secure store operation owing to both the process variation in the MTJ/MOS hybrid process and the variation in the local supply voltage. A time-consuming store operation directly causes high energy consumption, which reduce the energy that is saved during the sleeping time.

We previously proposed the non-volatile cool mega array (NVCMA), which was the first CGRA to introduce STT-MRAM-based NVFFs for all memory elements. The employed NVFF, which is known as the verify-and-retriable NVFF (VR-NVFF) [4], minimizes the energy required for writing data by its verify-and-retriable mechanism, taking into account the variations. The microcontroller that is built into the CMA contains additional instructions to control the NVFF and manages the store, verify, and restore operations as well as the PG. A real chip evaluation demonstrated the capability of the VR-NVFF to reduce the writing energy [5].

In this study, based on experience of NVCMA, we developed a second-generation CGRA with NVFFs known as NVCMA with multi-context (NVCMA/MC), and implemented a real chip with a 40 nm CMOS/MTJ hybrid technology. To make use of the NVFFs, NVCMA/MC provides four hardware contexts and can hold configuration data for multiple tasks without the sacrifice of memory leakage. Furthermore, it provides a larger PE array structure and pipelined registers inside the PE to widen the possibility of the optimization of the performance and power consumption. The evaluation of this study mainly focused on the energy-saving effect of the verify-and-retriable mechanism of the NVFFs and the multi-context facility of NVCMA/MC.

The remainder of this paper is organized as follows. First, the VR-NVFF structure with its two-step store (TSS) control flow is outlined in Section II. Following the review of related work in Section III, NVCMA/MC is introduced in Section IV. Section V presents the results of energy reduction by the TSS, and the energy-saving effect of the multi-context with PG is evaluated in Section VI. Finally, section VII concludes the paper.

II. VR-NVFF

NVCMA/MC employs an STT-MRAM-based NVFF for all of the NVM elements. The MTJ, which is the core storage element for the STT-MRAM, stores information with the property that the magnetization direction is changed by a certain amount of electric current. The STT-MRAM requires significantly high power and a sufficiently long time to write data into the MTJ safely. Improved NVFFs [6], [7] have been proposed to eliminate the store energy. Before storing the data, the NVFF compares the currently stored data with the data to be written. The store process is executed only if these data differ; otherwise, no action is taken for the NVFF.

The VR-NVFF [4] is a further improved NVFF, which considers the variations in the required writing time owing to both the process variations in the MTJ/MOS hybrid process and the variations in the local supply voltage. Many NVFFs can be stored in a short time, whereas others on the same chip may require a considerably long time. Thus, the application of a large current to all NVFFs for a sufficiently long time is a substantial waste of energy.

The VR-NVFF circuit is presented in Fig. 1. It consists of a common master latch and a slave latch, as well as two MTJs. The VR-NVFF is distinguished by its balloon latch and independent paths from the MTJs for verification and retry features. Comparisons between the data in the MTJs and those in the slave latch are performed with an XOR gate after reading out the store data in the MTJs to the balloon latch. If CMP_OUT, the output from the XOR gate is ”0,” which means that the data are written into the MTJs correctly. Therefore, TR1 and TR2 are forced to be turned off, and no further current will flow into the MTJs. In this manner, the VR-NVFF can validate whether or not the store operation is required in bit-level units.

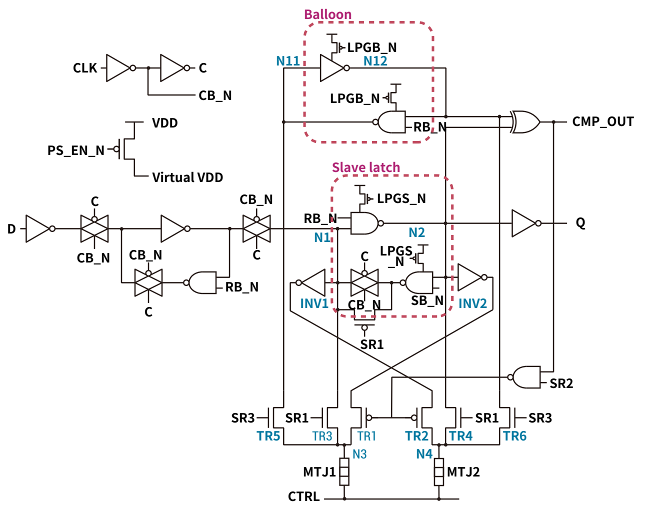


Fig. 1: VR-NVFF circuit.

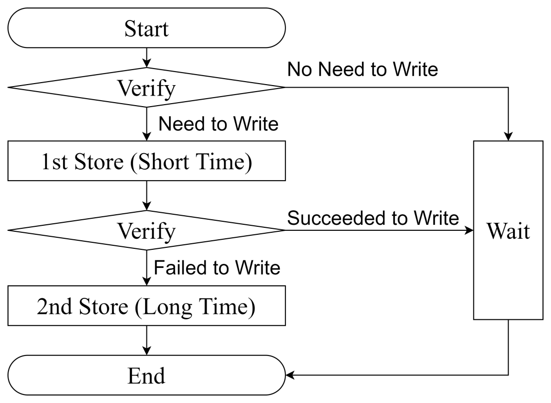


Fig. 2: TSS flow of VR-NVFF.

Using the VR-NVFF, an efficient store operation can be performed by first storing the majority of the NVFFs in a short time (T_{short}), and subsequently, following verification, by executing a second store operation for a long time (T_{long}) for the remaining NVFFs as a retry. This set of store control is known as TSS control, as illustrated in Fig. 2. The current tends to be high in the first store operation, but it is only required for a short time. In the second operation, the duration of the time to store is set to be longer than the first one, but the current is substantially lower because the number of NVFFs to write decreases significantly after the first trial. The portion of successfully stored NVFFs after the first store is considered to vary with T_{short}. Therefore, by optimizing the duration of T_{short} and T_{long}, the energy reduction effect of the TSS is expected to be maximized.

As control signals for the VR-NVFF, there are eight NVFF management signals (SR1, SR2, SR3, SB_N, RB_N, LPGB_N, LPGS_N, and CTRL), one for PG (PS_EN), and one for clock gating. In NVCMA/MC, these signals are controlled by the dedicated instruction of a microcontroller, as described in IV-B2.

III. Related Work

The introduction of NVM into FPGAs offers the following three advantages: First, a rapid start without transferring configuration data from outside memory is possible. Second, the large leakage power, which is one of the major disadvantages of FPGAs over ASICs, can be suppressed. The relatively large power that is required to store the data into NVM is not a concern because the frequency of rewriting the configuration memory is much smaller than that of the data memory. MTJ-based NVM was introduced into FPGAs for sensor-node applications [2], and the authors of [3] attempted to incorporate LUTs into FPGAs.

However, as the total amount of configuration data is smaller in CGRAs than in FPGAs, NV technology has been used for interconnection networks and logic elements, rather than for configuration memory. R-Accelerator [8] used RRAM technology for the logic and interconnection network of the CGRA, and configuration memory-less CGRA was proposed. Instead of changing the configuration data, the logic and interconnections were changed directly by reconfigurable wires using RRAM technology. HyCube [9] also switched its interconnection network.

Unlike these methods, our approach introduces the VR-NVFF, which can save on the storing energy with sophisticated control. The CGRA microcontroller can be used for VR-NVFF management and is applied for all memory elements in the CGRA. In the following section, we discuss the design policy.

IV. NVCMA/MC

A. NVCMA

1) Cool Mega Array (CMA): NVCMA/MC is based on the CMA architecture [10], which is a type of straightforward CGRA for mapping a straightforward dataflow on the PE array (Fig. 3). Piperench [11], Kilo-core [12], S5 engine [13], and EGRA [14] also fall into this category. Unlike in other CGRAs, the PEs of the CMA do not have register files to hold intermediate results, and the PE array of the CMA forms large combinatorial circuits. The clock signal for the PE array is removed to reduce the dynamic power consumption. Although a long critical path delay is a concern, PE arrays are designed to be multi-cycle paths and the system clock frequency does not decrease. The number of execution cycles is programmable depending on the mapped application. The computation of the CMA is performed as follows: First, the data to be computed are read out from the interleaved dual-port data memory modules, passed through the permutation network, and stored in the fetch registers that are provided at the input of the large PE array. After a certain delay time, the results are stored in the gather register, and subsequently transferred into the interleaved dual-port memory through another permutation network. The dataflow in the system is controlled by a tiny microcontroller that provides a simple RISC-style ISA.

2) Non-volatile CMA (NVCMA): CMA was originally designed for low-power edge computing, which often requires sleep-down and wake-up to save the battery. The provision

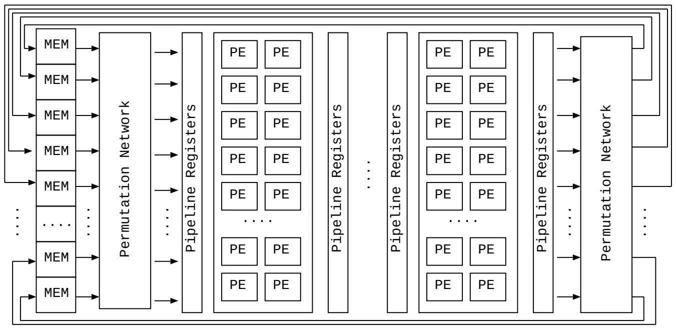


Fig. 3: Straightforward CGRAs

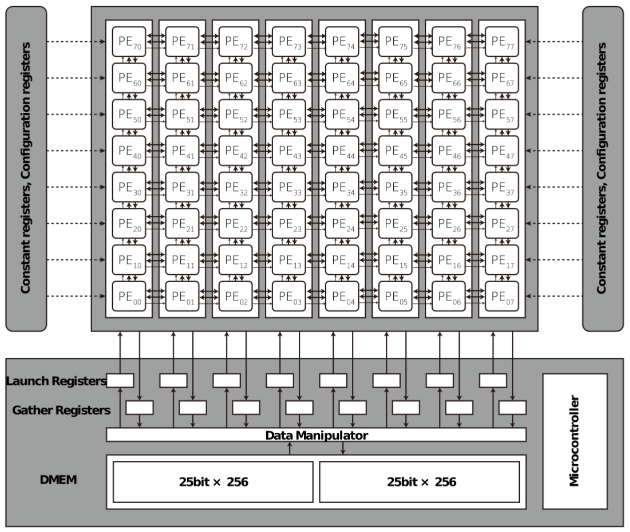


Fig. 4: Block diagram of NVCMA.

of configuration, constant, and data registers with NVFFs enables rapid sleep-down and wake-up without obtaining data from external memory. Our key concept was the use of a microcontroller to manage the dataflow in the CMA to control the NVFFs. NVCMA [15][16][5] was implemented as the first prototype of CMA with the introduction of NVFFs into its configuration registers, constant registers, and data memory, as well as the instruction memory of the microcontroller. The NVFFs are divided into 10 domains and dedicated instructions to manage the control signals were provided for each domain. NVCMA has an 8×8 PE array and a tiny microcontroller to manage the data transfer between the data memory and PE array Fig. 4. The PE array is connected to the network using a two-channel island-style interconnect. The SEs transfer input data from the south, west, and east of the PE and forward the ALU output data to the PEs in the appropriate direction according to the configuration data. The real chip evaluation results demonstrated that the two-phase store operations could efficiently reduce the power for the store operation [5].

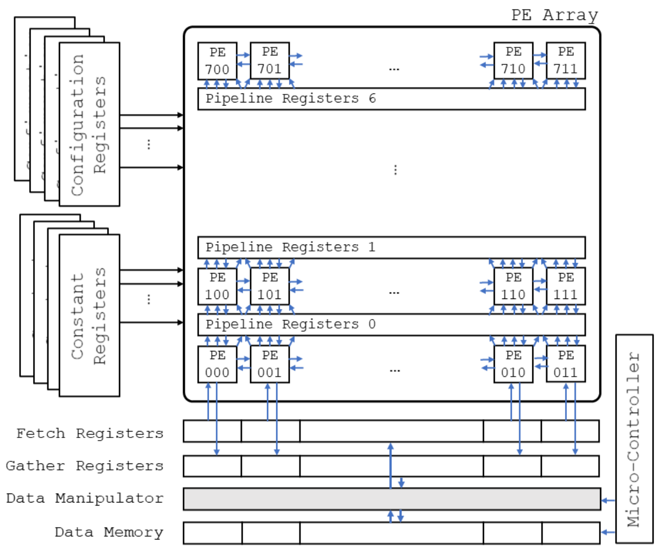


Fig. 5: Block diagram of NVCMA/MC

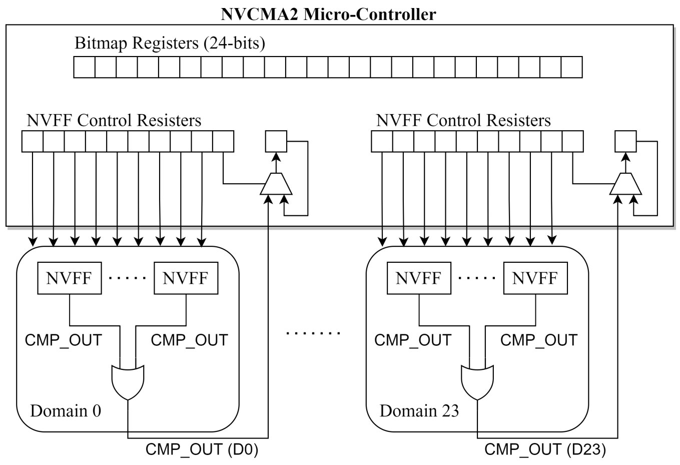


Fig. 6: NVCMA/MC microcontroller.

B. *NVCMA with multi-context (NVCMA/MC)*

1) Overview of NVCMA/MC: Based on our experience of NVCMA, we designed NVCMA/MC as the second version. The most significant difference is that NVCMA/MC provides four hardware contexts. In all past CMA implementations, the multi-context structure that provides multiple configuration memory modules was eliminated. Although the multi-context structure extends the flexibility and usability of CGRAs, the increasing energy required to switch the hardware context is not tolerable for CMA, the primary goal of which is to save energy. However, the energy required to maintain the configuration and constant data for each context can be ignored with the introduction of the NVFFs and PG mechanism. The four hardware contexts that are provided in NVCMA/MC are used to maintain four different sets of tasks inside the chip, rather than replacing the configuration data from outside memory when the task is switched. Unlike in other multi-context CGRAs with clock-by-clock context switching, task-by-task one is assumed. The context switching can be triggered by executing an instruction with the microcontroller or by selecting signals from outside the chip.

Moreover, the size of the PE array is expanded to 12×8 to execute larger tasks in parallel. Although increasing the size will increase the power consumption, the static power increases can be reduced by the introduction of NVFFs and the power-switching mechanism. Therefore, the size of the hardware can be extended significantly in NVCMA/MC. As opposed to the PE array with monolithic combinatorial circuits that is used in NVCMA, we insert pipeline registers for each row of PEs. The PE array structure is the same as that of VPCMA2, and the benefit of introducing pipeline registers is discussed in [17]. Fig. 5 presents a block diagram of NVCMA/MC.

2) Controlling NVFFs: The control instructions for the NVFFs are designed based on those for NVCMA. A certain number of NVFFs in a store domain share control signals and are managed together. Unlike NVCMA, which provides 10 store domains, 24 store domains are included. Nine of the 24 store domains are used for the data memory, two are used for the instruction memory of the microcontroller, and the remaining 13 are divided among the four configuration memories, contexts #0-3. As the NVFFs for the different hardware contexts are mapped to different store domains, store/restore control can be applied to the configuration data for inactive contexts.

For each store domain, NVC instruction sets or resets 10 bits of NVFF control registers corresponding to control signals according to the 24 bits that are stored in the dedicated bit map registers, as indicated in Fig. 6. Using a sequence of NVC instruction, set, reset, store, restore, and verify are executed on all NVFFs in the store domain specified by the bit map. Thereafter, CBB instruction is applied to clear a bit of the bit map register and branch, unless all bits in the bit map register become zero. Six power domains for the PG are provided by integrating a certain number of store domains. As illustrated in Fig. 7, PGC instruction has a 6-bit operand corresponding to each power domain and controls the power of these domains independently. The wake-up is controlled by either the instructions or signals from outside the chip. To control long-time sleep, PSE instruction manages counters for long pauses.

3) Design CAD: The programming of NVCMA/MC must be performed separately for the PE array and microcontroller. The process that is executed on the PE array is described in C language, and the development environment MENTAI [18] translates it into a data-flow graph. It is fitted to the PE array with a genetic algorithm-based general mapping tool known as GenMap [19]. Programmers can select their favorite mapping from several candidates, each of which is optimal from different perspectives. The code of the microcontroller, including the NVFF control, is described in assembly code and translated directly into machine code.

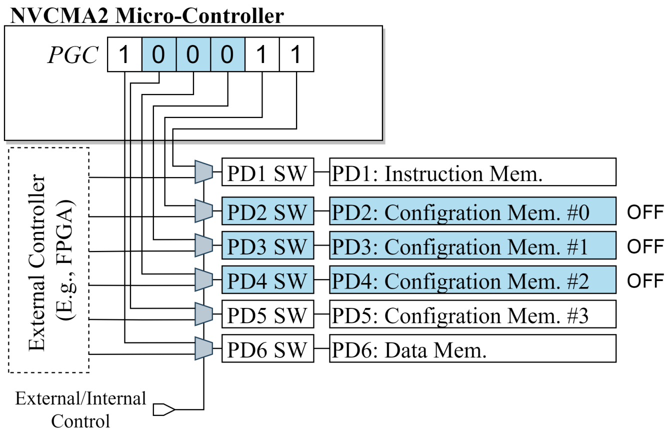

NVCMA2 Micro-Controller
PGC
1
0
0
0
1
1
External Controller (E.g., FPGA)
PD1 SW
PD1: Instruction Mem.
PD2 SW
PD2: Configration Mem. #0
OFF
PD3 SW
PD3: Configration Mem. #1
OFF
PD4 SW
PD4: Configration Mem. #2
OFF
PD5 SW
PD5: Configration Mem. #3
PD6 SW
PD6: Data Mem.
External/Internal Control

Fig. 7: NVCMA/MC PGC instruction.

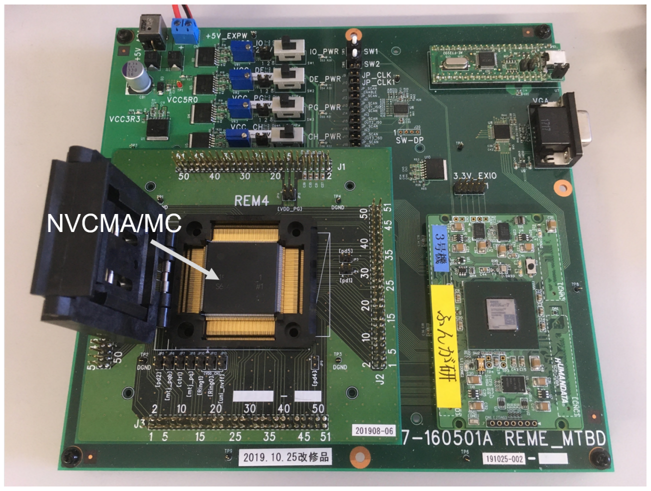


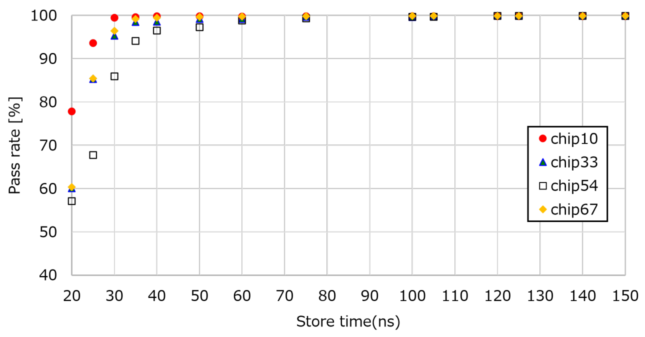
Fig. 8: Implemented chip and board.

V. Energy reduction by Two-Step Store

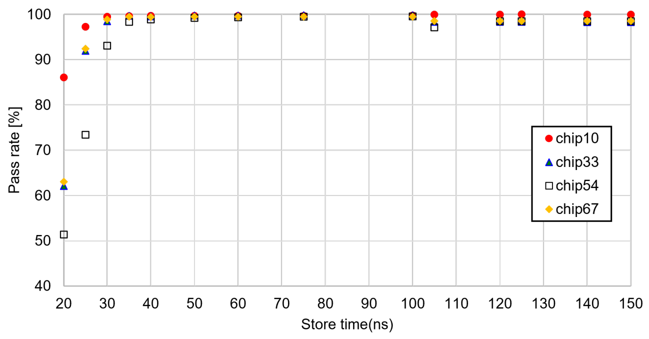
We implemented the proposed architecture with 40 nm CMOS/MTJ hybrid technology and conducted real chip measurements. Fig. 8 presents a photograph of the chip and the board for the testing and measurements. The common conditions for all measurements were an operating frequency of 28 MHz and a supply voltage V_{DD} of 1.15 V.

A. Optimal store duration

The optimal store durations of T_{short} and T_{long} for the short store and long store, respectively, were determined from the perspective of minimizing the store energy. T_{long} was regarded as the minimum required duration of time for completely storing all NVFFs except those stacked at 0/1 with a hard error. T_{short} was determined to minimize the total energy of the TSS control. A store domain with 2,400 NVFFs was used for the measurements, which is one of the store domains for the data memory. First, we measured the supplied current $I_{\mathrm{store\_all}}$ to store all 2,400 NVFFs simultaneously. Thereafter, the pass rates PR calculated by (1) for various store times were measured.



(a) Store all NVFFs to “0”



(b) Store all NVFFs to “1”

Fig. 9: Measured pass rate with various store times.

$$PR\,[\%] = \frac{\#\,of\;successfully\;stored\;NVFFs}{\#\,of\;all\;NVFFs\;(=2,400)} \times 100 \qquad (1)$$

T_{long} was assumed to be equal to the store time at which PR was saturated. T_{short} was determined to minimize the total energy of the short store and long store E_{total}, which was calculated by (2).

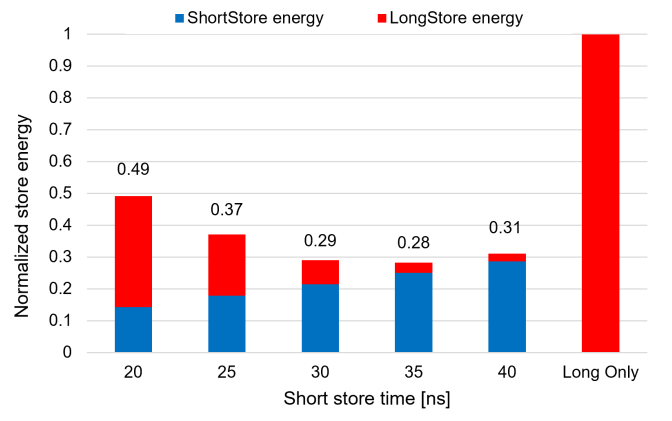
$$\begin{aligned} E_{\mathrm{total}} &= E_{\mathrm{short}} + E_{\mathrm{long}} \\ E_{\mathrm{short}} &= I_{\mathrm{store\_all}} \times V_{\mathrm{DD}} \times T_{\mathrm{short}} \\ E_{\mathrm{long}} &= I_{\mathrm{store\_all}} \times (1 - \frac{PR_{\mathrm{short}}}{100}) \times V_{\mathrm{DD}} \times T_{\mathrm{long}} \end{aligned} \qquad (2)$$

where PR_{short} is the pass rate after the short store.

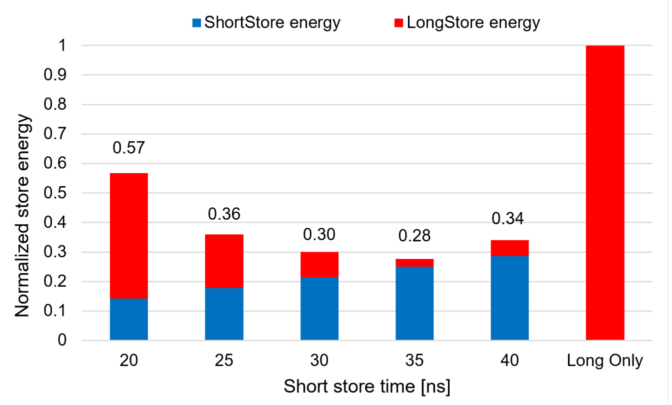
Fig. 9 presents the measured PR when storing all NVFFs to ”0” and ”1,” respectively. In each case, PR was saturated when the store time exceeded approximately 120 ns. Thus, in order to ensure the success of the store of more bits, T_{long} was determined as 140 ns. Note that PR does not reach 100% owing to hard errors that occurr in MTJs at a certain probability.

The normalized store energy calculated by (2) with various short store times is depicted in Fig. 10. The long store time was always set to 140 ns. A time of 35 ns was considered as the optimal selection for T_{short}, at which E_{total} became the smallest.

Based on the above results, when T_{short} was set to 35 ns as the first store time and T_{long} was set to 140 ns as the second



(a) Store all NVFFs to "0"



(b) Store all NVFFs to "1"

Fig. 10: Estimated store energy with various short store times and 140 ns of long store.

store time, the pure store energy was estimated to decrease by 72% compared to the long store of 140 ns.

B. Store energy reduction by TSS control

The energy expended on the TSS control E_{TSS} with the optimal T_{short} and T_{long} was measured, as well as that of the long store only store control $E_{\text{Conv.}}$, which was regarded as the conventional method. E_{TSS} and $E_{\text{Conv.}}$ included the energy spent on the verify operations that were executed before each store operation, as expressed in (3) and (4), respectively. The store energies of both methods were compared for various numbers of NVFFs and the performance of the TSS control was evaluated.

$$E_{\text{TSS}} = E_{\text{verify}} + E_{\text{short}} + E_{\text{verify}} + E_{\text{long}} \tag{3}$$
$$E_{\text{Conv.}} = E_{\text{verify}} + E_{\text{long}} \tag{4}$$

C. Evaluation results of store energy

Fig. 11 presents the measured $E_{\text{Conv.}}$ and E_{TSS} with T_{short} = 35 ns and T_{long} = 140 ns. When the number of storing NVFFs was 2,400, a store energy reduction of 65% was achieved by the TSS control compared to the conventional method. However, it was also demonstrated that the effect of the TSS control declined with a decrease in the number of NVFFs. This was considered to be caused by the verify operations that were conducted before each store operation. A verify operation requires a certain current to be applied to all MTJs for all NVFFs in the store domain to verify whether or not an additional store is necessary. That is, E_{verify} is dependent on how many NVFFs the store domain has and is independent of the number of storing NVFFs. Therefore, with fewer NVFFs, E_{verify} becomes relatively higher and the energy saving effect of the TSS is more strongly offset by E_{verify}. In the case of Fig. 11, if the number of storing NVFFs is no more than approximately 100, the long store only control may be preferable. However, E_{verify} is expected to be reduced by optimizing the circuit design, which will be the focus of our future work.

VI. Multi-context with Power Gating

A. Expected energy reduction

As described in IV-B, NVCMA/MC can hold four different hardware contexts belonging to each independent power domain, and it is assumed to provide task-by-task context switching. Therefore, while one context is active, the power domains corresponding to the other three contexts can be powered off by the PG to cut off the leakage power. Fig. 12 presents a diagram of the power consumption and mode transition with and without PG in inactive contexts. P_{Run}, $P_{\text{Run\_PG}}$, P_{Idle}, $P_{\text{Sleep\_PG}}$, and P_{Recover} are the power consumptions during the "Run," "Run_PG," "Idle," "Sleep_PG," and "Recover" modes, respectively. Each power value was obtained in this measurement. The same correspondence applied for T as the time durations.

PG in inactive contexts can be performed during both the "Run" and "Standby" periods. During the "Run" periods, the leakage power for three unactive configuration memories can be cut off by the "Run_PG" mode, and all four can even be eliminated by the "Sleep" mode during "Standby" periods.

However, PG is accompanied by the overhead of recovering the configuration data as "Recover" mode before restarting the program execution. Thus, we analyzed the break-even time BET to balance the energy reduction and energy overhead compared to the case without PG. The energy overhead for the PG (E_{Overhead}) is mainly caused by the restore operation and power switching, which is a fixed value. Moreover, a longer run and/or sleep period results in more energy being saved. When the energy balance is break-even, (5) is established.

$$E_{\text{Save\_Run}} + E_{\text{Save\_Sleep}} = E_{\text{Overhead}} \tag{5}$$

where $E_{\text{Save\_Run}}$ is the energy saving during run mode and

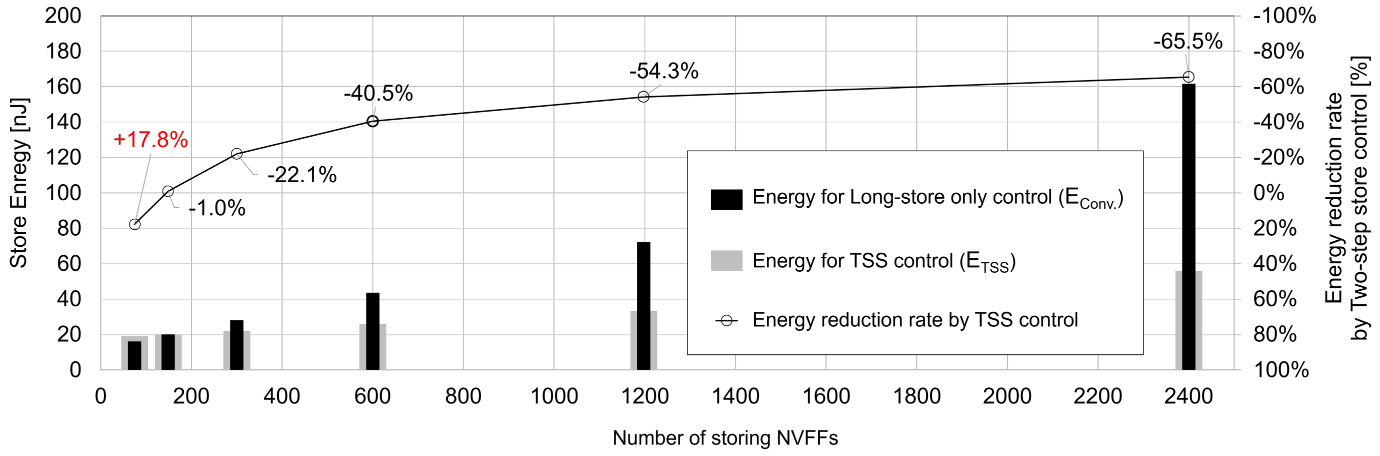


Fig. 11: Store energy comparison between TSS control and long store only control.

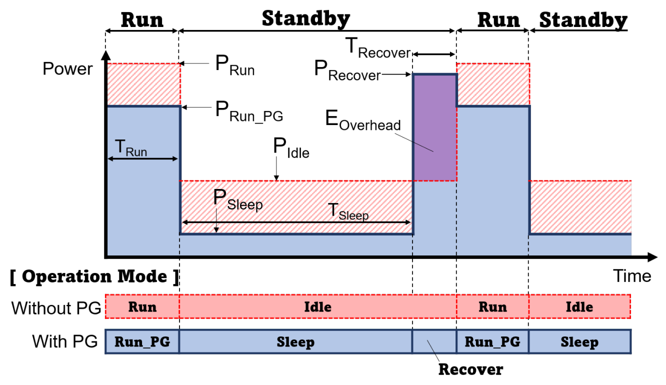


Fig. 12: Diagram of power consumption and mode transition with and without PG in inactive contexts

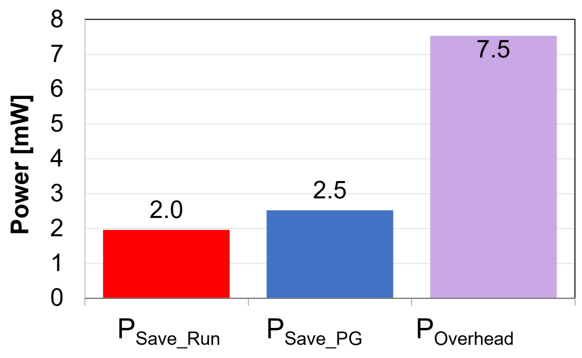


Fig. 13: Power savings and overheads with PG.

$E_{\text{Save\_Sleep}}$ is that during sleep mode, as expressed below.

$$\begin{aligned} E_{\text{Save\_Run}} &= P_{\text{Save\_Run}} \times T_{\text{Run}} \\ &= (P_{\text{Run}} - P_{\text{Run\_PG}}) \times T_{\text{Run}} \\ E_{\text{Save\_Sleep}} &= P_{\text{Save\_Sleep}} \times T_{\text{Sleep}} \\ &= (P_{\text{Idle}} - P_{\text{Sleep\_PG}}) \times T_{\text{Sleep}} \\ E_{\text{Overhead}} &= P_{\text{Overhead}} \times T_{\text{Recover}} \\ &= (P_{\text{Recover}} - P_{\text{Idle}}) \times T_{\text{Recover}} \end{aligned} \tag{6}$$

BET is given by (7) when (5) is satisfied.

$$\begin{aligned} BET &= T_{\text{Run}} + T_{\text{Sleep}} + T_{\text{Recover}} \\ &= T_{\text{Run}} + T_{\text{Standby}} \end{aligned} \tag{7}$$

B. Evaluated energy reduction by multi-context power gating

The measured $P_{\text{Save\_Run}}$, $P_{\text{Save\_PG}}$, and P_{Overhead} values are presented in Fig. 13. P_{Overhead} was approximately 3-4 times larger than $P_{\text{Save\_Run}}$ and $P_{\text{Save\_PG}}$. As the time required to restore one context data and restart the power (T_{Recover}) was 0.68 μs, E_{Overhead} was calculated as 5.1 nJ.

Based on the above measurements, BET were determined as illustrated in Fig. 14. The bar graph indicates that the energy balance was at break-even when the sum of T_{Run}, T_{Sleep}, and T_{Recover} was approximately 3 μs. For example, a 2.03 μs sleep mode or a 2.61 μs run mode would completely offset the recovery overhead. Similarly, a combination of a 1.00 μs run mode and 1.25 μs sleep mode or a 2.50 μs run mode and 0.09 μs sleep mode would balance out the recovery overhead. In summary, if one runs or sleeps for a longer duration than the break-even times, which is approximately 3.0 μs, the energy savings will be greater. For example, for an application that runs intermittently once every 1 ms for 500 μs and remains in standby for the remaining 500 μs, the energy savings would be 2.2 μJ per hour, including the overhead.

VII. Conclusions and Future Work

We proposed NVCMA/MC, which employs the STT-MRAM-based VR-NVFFs for all context memory elements. Similar to the previous NVCMA, it also manages the store/verify/restore operations with additional instructions provided to the microcontroller; however, the hardware contexts were extended to four and the size of the PE array was increased.

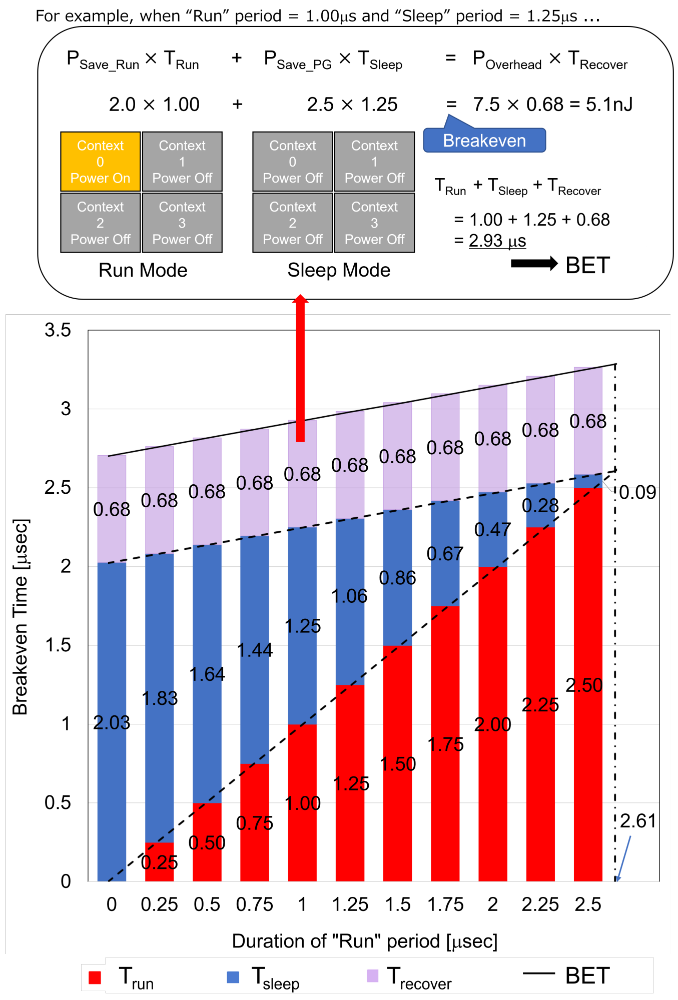


Fig. 14: Break-even time for multi-context PG.

The evaluation results using real chips with a 40nm MTJ/MOS hybrid process revealed that the two-phase storage mechanism with the first 35ns store and the second 140ns store could reduce the storing energy by 65% when writing 2,400 NVFFs.

Moreover, NVCMA/MC microcontroller has a dedicated instruction for the PG of the four context memories independently, and the PG can be constantly applied to inactive contexts. The PG power evaluation demonstrated that applications that run intermittently for short periods, with sleep and run cycles of approximately 3μs, can benefit from the energy savings of PG.

The operations on the PE array need to wait while the configuration data are restored, which is a time overhead. This is because NVCMA/MC microcontroller is responsible for both the operations on the PE array and the control of the NVFF, which cannot be performed simultaneously in parallel. Our future work will focus on refining the control method for NVFFs and hide the time overhead.

References

[1] S. G. Narendra and A. P. Chandrakasan, *Leakage in nanometer CMOS technologies*. Springer Science & Business Media, 2006.

[2] M. Natsui, D. Suzuki, A. Tamakoshi, T. Watanabe, H. Honjo, H. Koike, T. Nasuno, Y. Ma, T. Tanigawa, Y. Noguchi, M. Yasuhira, H. Sato, S. Ikeda, H. Ohno, T. Endoh, and T. Hanyu, "12.1 An FPGA-Accelerated Fully Nonvolatile Microcontroller Unit for Sensor-Node Applications in 40nm CMOS/MTJ-Hybrid Technology Achieving 47.14 μ W Operation at 200MHz," in *2019 IEEE International Solid- State Circuits Conference - (ISSCC)*, Feb 2019, pp. 202–204.

[3] L. Xie, H. A. Du Nguyen, M. Taouil, S. Hamdioui, K. Bertels, and M. Alfailakawi, "Non-volatile look-up table based FPGA implementations," in *2016 11th International Design Test Symposium (IDT)*, Dec 2016, pp. 165–170.

[4] K. Usami, J. Akaike, S. Akiba, M. Kudo, H. Amano, T. Ikezoe, K. Hiraga, Y. Shuto, and K. Yagami, "Energy efficient write verify and retry scheme for mtj based flip-flop and application," in *2018 IEEE 7th Non-Volatile Memory Systems and Applications Symposium (NVMSA)*. IEEE, 2018, pp. 91–98.

[5] K. Usami, S. Akiba, H. Amano, T. Ikezoe, K. Hiraga, K. Suzuki, and Y. Kanda, "Non-volatile coarse grained reconfigurable array enabling two-step store control for energy minimization," in *2020 IEEE Symposium in Low-Power and High-Speed Chips (COOL CHIPS)*. IEEE, 2020, pp. 1–3.

[6] S. Yamamoto, Y. Shuto, and S. Sugahara, "Nonvolatile delay flip-flop using spin-transistor architecture with spin transfer torque mtjs for power-gating systems," *Electronics letters*, vol. 47, no. 18, pp. 1027–1029, 2011.

[7] M. Kudo and K. Usami, "Nonvolatile power gating with mtj based nonvolatile flip-flops for a microprocessor," in *2017 IEEE 6th Non-Volatile Memory Systems and Applications Symposium (NVMSA)*. IEEE, 2017, pp. 1–6.

[8] Z. Chen, H. Zhou, and J. Gu, "R-Accelerator: An RRAM-Based CGRA Accelerator with Logic Contraction ," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 27, no. 11, pp. 2655–2667, Nov. 2019.

[9] M. Karunaratne, A. K. Mohite, T. Mitra, and L.-S. Peh, "Hycube: A cgra with reconfigurable single-cycle multi-hop interconnect," in *Proceedings of the 54th Annual Design Automation Conference 2017*, 2017, pp. 1–6.

[10] N. Ozaki, Y. Yasuda, M. Izawa, Y. Saito, D. Ikebuchi, H. Amano, H. Nakamura, K. Usami, M. Namiki, and M. Kondo, "Cool mega-arrays: Ultralow-power reconfigurable accelerator chips," *IEEE Micro*, vol. 31, no. 6, pp. 6–18, Nov 2011.

[11] H. Schmit, D. Whelihan, A. Tsai, M. Moe, B. Levine, and R. R. Taylor, "PipeRench: A virtualized programmable datapath in 0.18 micron technology," in *Custom Integrated Circuits Conference, 2002. Proceedings of the IEEE 2002*. IEEE, 2002, pp. 63–66.

[12] B. Levine, "Kilocore: Scalable, High Performance and Power Efficient Coarse Grained Reconfigurable Fabrics," in Proc. of International Symposium on Advanced Reconfigurable Systems, 2005, pp. 129–158.

[13] J. M. Arnold, "S5: The Architecture and Development Flow of a Software Configurable Processor," in *Proc. of the 4th IEEE Int'l Conf. on Field Programmable Technology (ICFPT2005)*, December 2005, pp. 120–128.

[14] G. Ansaloni, P. Bonzini, and L. Pozzi, "EGRA: A coarse grained reconfigurable architectural template," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 6, pp. 1062–1074, 2011.

[15] T. Ikezoe, H. Amano, J. Akaike, K. Usami, M. Kudo, K. Hiraga, Y. Shuto, and K. Yagami, "A coarse grained-reconfigurable accelerator with energy efficient mtj-based non-volatile flip-flops," in *2018 International Conference on ReConFigurable Computing and FPGAs, ReConFig 2018, Cancun, Mexico, December 3-5, 2018*, 2018, pp. 1–6.

[16] T. IKEZOE, T. KOJIMA, and H. AMANO, "Reconvering faulty non-volatile flip flops for coarse-grained reconfigurable architectures," *IEICE Transactions on Electronics*, 2020.

[17] T. Kojima and H. Amano, "Refinements in data manipulation method for coarse grained reconfigurable architectures," in *2019 14th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*. IEEE, 2019, pp. 113–120.

[18] A. Ohwada, T. Kojima, and H. Amano, "Mentai: A fully automated cgra application development environment that supports hardware/software co-design," in *Proceedings of SASIMI 2021*, March 2021.

[19] T. Kojima, N. A. V. Doan, and H. Amano, "Genmap: A genetic algorithmic approach for optimizing spatial mapping of coarse-grained reconfigurable architectures," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 11, pp. 2383–2396, 2020.