# Adaptive MRAM-Based CGRAs

Xiaobin Liu, Tedy Thomas, Alan Boguslawski, and Russell Tessier

Department of Electrical and Computer Engineering
University of Massachusetts, Amherst, MA, USA

***Abstract*—In this paper, we describe the use of magnetic RAM (MRAM) in coarse-grained reconfigurable arrays (CGRAs) as a configuration cache to allow for bulk low-energy storage and rapid device reconfiguration. If an energy-saving configuration update for an application is needed, a new configuration can be quickly swapped into compute blocks and interconnect switchboxes to minimize system down time. The determination of when to configure and an analysis of the CGRA architectural impact of MRAM is evaluated via system-level emulation. Our experiments show that the use of MRAM reduces overall application energy consumption by nearly 30% when dynamic reconfiguration is used.**[1]

## I. Introduction

In this paper, we address concerns about coarse-grained reconfigurable array (CGRA) reconfiguration time and energy in response to changing environmental parameters for signal processing applications. Our main contribution is the storage of multiple configurations for CGRA processing elements and associated interconnect in *magnetic RAM (MRAM)* close to their target resources. This emerging memory architecture is well suited to bulk configuration storage, given its non-volatile and low leakage characteristics. The reduced size of MRAM in contrast with SRAM (roughly a factor of four) allows for increased configuration storage with the ability to support numerous applications, both active and inactive. Since both the energy and access time characteristics of MRAMs differ from SRAM, architectural changes to the CGRA system are required. Our work includes a full evaluation of MRAM implementation tradeoffs for CGRAs and the energy benefits that can be achieved. We demonstrate that this CGRA design is rapidly responsive to changes in processing demands, providing *just enough* computation to the target application.

The specific contributions of this work include: (1) Examination of the architectural impact of MRAM on CGRAs with respect to memory architecture, (2) quantification of energy savings from using MRAM instead of on-chip SRAM to cache CGRA configuration information for time-varying signal processing applications and (3) quantification of the energy reduction available from fast CGRA reconfiguration based on DSP application parameters.

## II. Background

Energy-efficient implementation of baseband signal processing and image processing is challenging due to recent industrial trends. For example, link adaptation (or adaptive coding) is widely used to maximize overall data transmission throughput by adapting the coding scheme according to the quality of the radio channels [1]. Adaptive image processing is being performed on a large class of personal and automotive devices, often in energy-constrained environments. These trends create significant run-time computational workload variations. Specific signal processing applications which exhibit *time-varying* computational loads, communication decoding (e.g. Reed Solomon) and image processing (filtering, motion estimation), are targeted to our platform for experimentation.

Coarse-grained reconfigurable arrays generally contain tens or hundreds of ALU-based processing elements (PEs) which include small amounts of memory. CGRAs typically store configuration information on-chip in SRAM and the information is distributed to processing elements and switchboxes as application execution changes (e.g. in response to a context switch or application update). In this paper, we address SRAM power limitations through the use of MRAM to allow for the storage of multiple configurations for the processing elements and determine appropriate times to reconfigure the system.

MRAM has emerged as a promising candidate for on-chip non-volatile RAM due to programmability using standard supply voltages. Compared with SRAM, MRAM achieves a 4 to $7\times$ storage density improvement [2]. Each magnetic tunneling junction (MTJ), the basic storage element in MRAM, has two ferromagnetic layers separated by an oxide barrier layer. MTJ resistance is dependent on the directions of magnetization of the ferromagnetic layers (parallel or anti-parallel). In spin-torque transfer memory (STT-RAM)[2], magnetization occurs using write currents in the MTJ in opposite directions. MRAM cells typically exhibit substantially lower static power than SRAM ($10\times$) and similar dynamic read power and latency. However, high write energy and a long write latency are observed (3 to $10\times$ greater) [2].

## III. MRAM-enabled Coarse-Grained Reconfigurable Array

Our MRAM-enabled coarse-grained architecture is designed to support streaming applications with pre-scheduled communication paths. The architecture is based on a two-dimensional array of ALU-based processing elements. Each cell contains a 32-bit ALU, a 36K-word data memory, and a 24K-word configuration memory. These memories are directly connected to

[1]An extended version of this paper is available at http://www.ecs.umass.edu/ece/tessier/liu-fpl15-extended.pdf.

[2]We use the terms MRAM and STT-RAM interchangably.

TABLE I
MRAM AND SRAM PARAMETER COMPARISON FOR TWELVE 256K×32 (1MB), THREE 256K×128 (4MB), AND ONE 256K×384 (12 MB) MEMORY BANKS. ACCESS TIMES AND ENERGY VALUES ARE PER MEMORY ACCESS. ENERGY AND LEAKAGE VALUES ARE SUMS ACROSS ALL MEMORY BANKS.

| | MRAM | | | SRAM | | |
|---|---|---|---|---|---|---|
| Output bits/bank | 32 | 128 | 384 | 32 | 128 | 384 |
| Banks | 12 | 3 | 1 | 12 | 3 | 1 |
| PEs/ bank ($n$) | 1 | 4 | 12 | 1 | 4 | 12 |
| Read time (ns) | 1.30 | 1.67 | 2.74 | 0.60 | 1.23 | 3.39 |
| Write time (ns) | 5.36 | 5.88 | 6.73 | 0.49 | 0.82 | 3.09 |
| Read ene. (pJ) | 470.04 | 382.20 | 517.63 | 364.08 | 416.30 | 313.58 |
| Write ene. (pJ) | 356.88 | 317.13 | 592.75 | 273.72 | 349.35 | 256.46 |
| Area ($mm^2$) | 29.04 | 14.76 | 13.09 | 40.56 | 38.85 | 45.77 |
| Leakage (mW) | 42.72 | 11.61 | 5.27 | 102.24 | 95.94 | 75.04 |

much larger multi-MB non-volatile memory (MRAM) blocks which can be used to change their configuration memory contents. The array interfaces to a network of multiplexer-based switchboxes to provide inter-element communication. Data are switched in 32-bit increments to lower the switch configuration memory overhead. A dedicated microprocessor is located on the bottom of the array to provide control over the loading of configurations to MRAM (if needed) and coordinating signal processing application data transfer to and from DRAM.

The switchbox for each processing element operates on a pre-determined schedule that coordinates communication with neighboring elements. Schedule information for cycle-by-cycle configuration of the multiplexer-based crossbar is stored in the *schedule memory*. Buffers and flow control signals are provided on each interface port to prevent overflow. On each cycle for a port, data can be forwarded from the buffer or the neighbor. Each buffer contains storage for data from multiple independent streams (effectively, virtual channels).

A key architectural feature in this scalable multi-PE system is the use of MRAM to cache multiple PE and switchbox configurations. To allow for faster configuration download, many CGRAs [3] distribute the SRAM-based configuration cache so that one *bank* is located near each target PE. As a result, individual PEs can be updated separately, as needed. However, the characteristics of MRAM and the use of streaming DSP applications draw this approach into question. Specifically, (1) MRAM has substantially reduced leakage power versus SRAM. As the array size of the MRAM increases, the ratio of MRAM to SRAM leakage increases, advocating for the use of larger MRAM arrays. (2) The read time of MRAM array accesses grow at a much smaller rate as the array increases in size versus corresponding access time increases in SRAM memory arrays. (3) Most PEs of a signal processing application can be reconfigured at the same time, advocating for wide MRAM arrays which can transfer configurations to multiple PEs at the same time.

As a result, we can consider using a single memory bank as a configuration cache for multiple PE/switchbox pairs and configuring all pairs at the same time. To quantify the potential of this approach, parameters for similarly sized MRAM and SRAM caches were generated. NVSIM [4] is used to evaluate circuit-level area, performance, power, and energy for our STT-RAM (MRAM) implementations. The NVSIM LOP (low power) library is used to evaluate read/write access time and static and dynamic energy for varying MRAM array sizes. NVSIM was also used to model low-power SRAM banks in the same 40 nm size.

Table I provides an MRAM/SRAM comparison for 12 MB of configuration cache distributed across 12 PE/switchbox pairs. The number of banks of each memory and the respective output bits per bank are listed in the first two rows of the table. Energy and leakage values are sums across all memory banks. For 1MB (32 bit output), 4MB (128 bit output), and 12MB (384 bit output) memory configurations, the ratio of total leakage power for SRAM versus MRAM are 2.4, 8.3, and 14.2, respectively. In general, STT-RAM cells exhibit little leakage, so much of the array leakage is due to the decoders and output multiplexer.

## A. Time-Varying Signal Processing Applications

Localized embedded MRAM configuration caches can dynamically reconfigure PE and switchboxes, enabling changes in signal processing algorithm implementations in response to the physical environment and/or signal characteristic changes (e.g. noise). Thus, *multiple, independent* CGRA configurations can be used for an application and swapped into the PE configuration and switchbox schedule memories as needed. Periodic changes between applications may also be needed. To illustrate the benefit of MRAM in supporting rapid, low-energy reconfiguration, three signal processing algorithms are mapped to our parallel, MRAM-enabled CGRA: a Reed Solomon communications decoder, a motion estimation algorithm, and a large multi-tap FIR filter.

**Reed Solomon (RS)** coding is widely used for communications in environments with noise. Message data is digitized and broken into multi-bit *symbols*. A group of $k$ data symbols is augmented with $n-k$ parity symbols used for error correction to form an $n$ symbol *codeword*. A Reed Solomon decoder provides error recovery by detecting and correcting codeword symbols [5]. The number of recovery operations can be tuned based on the amount of noise in the received channel data. While data communication throughput rates can vary, many applications require a constant codeword error rate (CER), the rate at which decoder symbols are in error. In response to changes in channel noise, system reconfiguration to maintain constant CER is required. An increase (decrease) in channel noise requires a more (less) powerful decoder implemented in each processing element. The processing elements are reconfigured to reflect the change.

**Motion estimation** is used in image processing to identify portions of an image that move from frame to frame. A new image is broken into windows which are repetitively compared against portions of a previous image to determine a motion vector. For improved accuracy, a smaller search window within a frame can be used. For reduced accuracy, a larger window is deployed, saving computation and energy. An **FIR filter** is used as the third signal processing application which exhibits time-varying computational needs. The number of taps are varied over time to trade off energy and filter performance.

## IV. Experimental Approach

To validate the energy and performance aspects of our approach, we use a combination of simulation and FPGA-based architectural emulation to accurately measure system energy and performance. The processing element in our system, a modified version of the ALU-based SPREE functional unit [6], and the switchbox were written in Verilog. The performance of the architecture for 12 PEs and switchboxes was validated using both RTL simulation and in-circuit emulation after design synthesis using an Altera DE4 FPGA board. Cycle-accurate performance and node toggle rate calculations for the applications were determined using the Altera Performance Counter Unit, instrumented in the FPGA hardware. Detailed power consumption was determined using a current measurement sensor included on the DE4 board. Power and performance were scaled from measured 40 nm FPGA values found in emulation to estimated 40 nm ASIC values. The switchbox and SPREE designs were analyzed by both Altera PowerPlay and Synopsys Design Compiler using the NanGate open core 40 nm library[3] to determine the scaling factors of 11.6 for performance and 5.0 for power.

The implementation specifics for each benchmark are provided below. In the **RS decoding** application, each PE is assigned one RS decoder. The error-correcting capability of the decoder (e.g. number of $k$ message symbols) is determined by the PE configuration. Our experimentation considers $k$ values ranging from 217 to 239 out of $n$ = 255 total symbols (e.g. RS(255,217) to RS(255,239)). A constant codeword error rate (CER) of $10^{-4}$ is used. The specific decoder configuration is selected based on channel noise. To simulate the behavior of a communication system, channel noise (SNR) was considered to vary at an accepted rate [5] of 1.5 seconds and a new noise value was randomly selected at this rate. If necessary, the CGRA configuration was updated in response to the change in noise, necessitating a PE configuration load from the attached configuration cache.

For **motion estimation**, a series of 1024×1024 pixel greyscale images are distributed to the PE array. Each of these images is part of a sequence in which motion can be detected. Subsequent images in the sequence are split into windows and each window is tested by each PE for motion. In this implementation, the accuracy of the motion estimation algorithm is dependent on the window size. For higher accuracy, a smaller window size is used, leading to more windows and increased computation. For reduced accuracy, the opposite effect is observed. In our experimentation, we consider square window sizes of 14, 16, 18, 20, 24 and 26 pixels on a side. Reconfiguration is considered every 0.5s based on changes in expected motion.

[3]www.nangate.com

TABLE II
PE and switchbox (SB) configuration bit sizes for one configuration of each application

| | one PE | | | whole CGRA | | |
|---|---|---|---|---|---|---|
| | PE | SB | Total | PE | SB | Total |
| RS | 81,920 | 16,384 | 98,304 | 983,040 | 196,608 | 1,179,648 |
| ME | 49,152 | 16,384 | 65,536 | 589,824 | 196,608 | 786,432 |
| FIR | 40,960 | 16,384 | 57,344 | 491,520 | 196,608 | 688,128 |

**FIR filtering** involves the implementation of a filter with tap counts ranging from 120 to 1920 taps. A stream of input data is sourced from the DRAM and the filtered data is streamed back to DRAM. For our experimentation, eight-bit sampled radar data is used. Computation for the taps is distributed evenly across the PEs. An increased number of taps leads to more accurate filtered data. Reconfiguration is considered every 0.5s based on changes in expected data quality.

## V. Results

Table II illustrates the configuration bit sizes required for the individual processing elements and switchboxes. A CGRA with 12 PEs is used for experimentation. The total bit count for 12 PE/switchbox pairs is shown in the table for each application. PE *configuration memory* bits set connections between the ALU and data memory. Switchbox configuration information, stored in the *schedule memory*, configures the routing crossbar on a per-cycle basis. As will be described later in this section, seven distinct Reed Solomon configurations, six distinct motion estimation configurations and five distinct FIR configurations are used. As a result, a configuration cache must hold multiple copies of configuration information for each PE/switchbox, even though only one is loaded into the PE config. memory and switchbox schedule memory at a given time. A total of 12 MB of configuration cache in the system is sufficient to hold these configurations.

An example of throughput and energy tradeoffs can be observed for the motion estimation application (Fig. 1) for search window sizes between 14 and 26. The figure indicates that the application throughput improves with increasing window size since fewer windows must be searched. A 4 MB cache size (256K×128, $n$ = 4) is assessed for this application. For comparable cache sizes, MRAM requires about 3 to 8× less energy than its SRAM counterparts.

Similar throughput and energy results are seen for the RS and floating point FIR applications. For example, the average energy per million decoded bits for the MRAM (1.60 mJ) is much less than the energy of a (255,217) RS decoder (11.76 mJ). However, an SRAM-based cache consumes a similar value (13.91 mJ). The throughput difference between a 120

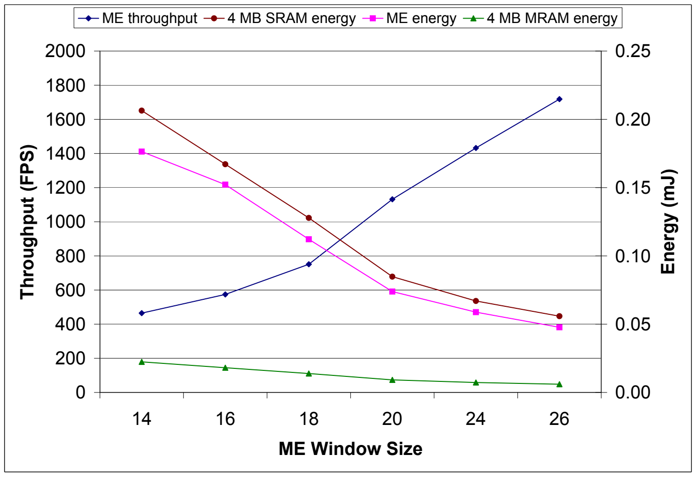

Fig. 1. Energy consumption and throughput of motion estimation mapped to 12 PEs for different ME window sizes. The energy consumption of the application and the caches was calculated over one processed frame. All configuration caches are implemented as 3 individual banks of 4 MB (256K$\times$128) MRAM or SRAM.

TABLE III
RESULTS OF DYNAMIC RECONFIGURATION FOR MAPPED APPLICATIONS USING THREE 4MB (256K$\times$128) MEMORY BANKS PER CGRA.

| | MRAM | | | | |
|---|---|---|---|---|---|
| | Ave. Power (mW) | Energy MRAM (mJ) | Av. Energy App (mJ) | Energy Total (mJ) | % Energy improvement |
| RS | 90.91 | 0.99 | 7.66 | 8.65 | 26 |
| ME | 94.31 | 0.013 | 0.106 | 0.11 | 38 |
| FIR | 97.91 | 7.11 | 59.31 | 66.42 | 33 |
| | SRAM | | | | |
| | Ave. Power (mW) | Energy SRAM (mJ) | Av. Energy App (mJ) | Energy Total (mJ) | % Energy improvement |
| RS | 176.46 | 9.12 | 7.66 | 16.78 | -43 |
| ME | 179.86 | 0.12 | 0.10 | 0.22 | -22 |
| FIR | 183.46 | 69.84 | 59.31 | 129.15 | 11 |

tap FIR implementation (5.2 MB/s) and the 1920 tap version (0.56 MB/s) is almost an order of magnitude. For three 4MB configuration caches, the average energy per million decoded bits of the MRAM (29.20 mJ) is less than half the energy of the 960 tap FIR filter (74.47 mJ). However, an SRAM-based cache is nearly the same value (69.84 mJ).

As mentioned in Section IV, applications with differing error-correcting capability and power consumption can be used at different times based on environmental factors. For example, for RS as $k$ changes from 239 to 217, decode rate is reduced from 16.39 to 6.34 Mb/s and the energy to decode one million bits increased from 4.44 mJ to 11.76 mJ. *One approach to providing sufficient error correcting capability would be to use an RS(255,217) decoder at all times and avoid decoder reconfiguration and the need to cache numerous configurations.* However, this approach limits opportunities for faster decode rates and reduced energy consumption.

For all three applications (Reed Solomon decoding, motion estimation, and FIR), energy consumption can be improved by adapting the application configuration to the measured environmental condition (e.g. noise in the communications channel for RS). In a final set of experiments, we examine the benefit of using dynamic reconfiguration of the CGRA with either an MRAM- or SRAM-based cache versus simply using the most powerful configuration of the application all the time (e.g. using the RS(255, 217) decoder for all RS decoding). A series of 10,000 random selections of decoder noise, required ME motion detection, and FIR accuracy were generated to represent changing environmental conditions. The application configuration which best met the requirements for these conditions were then chosen for each selection.

The average power and energy of the applications considering dynamic reconfiguration from an MRAM-based or SRAM-based cache is shown in Table III. Energy is determined per one million processed bits for RS and FIR and per frame for ME. The percentage improvement is the energy improvement for the average case using reconfiguration versus the most computationally powerful configuration of the application (e.g. RS(255, 217), window size 14 for ME, 1920 taps for FIR). For example, the RS(255, 217) decoder has a 11.76 mJ dissipation while the average energy with reconfiguration with MRAM is 8.65 mJ, a 26% savings. Although not shown in the table, the throughput performance improvement of the applications using the reconfigured average case versus the most computationally powerful configuration is 70%, 118%, and 94% for RS, ME, and FIR, respectively.

From Table III, it is apparent that the use of SRAM is limiting. The average energy consumption of the reconfigured RS and ME applications are *increased* versus the most computationally-powerful versions by 43% and 22%, respectively due primarily to leakage.

## VI. CONCLUSIONS

For CGRAs, leakage power can dominate configuration cache energy consumption. We show that the use of MRAM as a configuration cache is preferable to SRAM for a collection of three signal processing applications, due to reduced leakage. By using dynamic CGRA reconfiguration in response to environmental factors (e.g. noise), application energy consumption is reduced by about 30%.[4]

## REFERENCES

[1] T. Keller and L. Hanzo, “Adaptive multicarrier modulation: a convenient framework for time-frequency processing in wireless communications,” *Proceedings of the IEEE*, vol. 88, pp. 611–640, 2000.
[2] G. Sun, X. Dong, Y. Xie, J. Li, and Y. Chen, “A novel architecture of the 3D stacked MRAM L2 cache for CMPs,” in *Proc.: I'ntl Symp High Performance Comp. Arch.*, Feb. 2009, pp. 239–249.
[3] K. Eguro and S. Hauck, “Issues and approaches to coarse-grain reconfigurable architecture development,” in *Proc. FCCM*, Apr. 2003, pp. 111–120.
[4] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, “NVSim: A circuit-level performance, energy, and area model for emerging nonvolatile memory,” in *Proc.: ICCAD*, Nov. 2012, pp. 994–1007.
[5] L. Atieno, J. Allen, D. Goeckel, and R. Tessier, “An adaptive Reed-Solomon errors-and-erasures decoder,” in *Proc. FPGA*, Feb. 2006.
[6] P. Yiannacouras, J. G. Steffan, and J. Rose, “Application-specific customization of soft processor microarchitecture,” in *Proc. Int'l Symp. on FPGAs*, Feb. 2006, pp. 201–210.

[4]This work was supported by NSF grant EECS-1201834.