Brief Papers

Exploring the Use of Emerging Nonvolatile Memory Technologies in Future FPGAs

Yangyang Pan, Yiran Li, Hongbin Sun, Wei Xu, Nanning Zheng, and Tong Zhang

***Abstract*— As new nonvolatile memory technologies become increasingly mature, there has been a growing interest on investigating their use in future field-programmable gate arrays (FPGAs). Similar to existing FPGAs with embedded Flash memory, future FPGAs can embed these new nonvolatile memories to persistently store configuration data. By comparing with prior work, we first propose the more appropriate design style for new nonvolatile configuration data storage memory. Moreover, this brief studies a dynamic random-access memory (DRAM)-based FPGA design strategy enabled by high-density embedded nonvolatile memory. Existing FPGAs do not use on-chip DRAM cells for configuration data storage mainly because DRAM self-refresh involves destructive DRAM read. This problem can be solved, if we use embedded nonvolatile memory as primary FPGA configuration data storage and externally refresh on-chip DRAM cells. Analysis and simulations have been carried out to demonstrate the potential advantages of such a design strategy.**

***Index Terms*— Dynamic random-access memory (DRAM), field-programmable gate arrays (FPGA), magnetoresistive random-access memory (MRAM), nonvolatile memory.**

I. Introduction

Over the past decade, the semiconductor industry has experienced a resurgence of interest in the search for highly scalable memory technologies. Among various new memory technologies, phase-change random-access memory (PCRAM) and magnetoresistive random-access memory (MRAM) are the two most promising candidates and hence have received a lot of attention [1]. In this brief, we are interested in exploring the potential of realizing embedded nonvolatile configuration data storage for future field-programmable gate arrays (FPGAs). Most FPGAs use distributed on-chip static random-access meomory (SRAM) cells to store the data for configuring both logic components and interconnects, where configuration data are persistently stored off-chip and loaded to initialize on-chip SRAM cells during power-up. Flash memory has been used to realize nonvolatile FPGAs with two different design styles: 1) distributed-embedded Flash memory cells can replace SRAM cells to store configuration data and configure logic components and interconnect (e.g., FPGA devices from Actel [2]), which is referred to as *fully distributed design style* in this brief and 2) standard SRAM-based FPGAs are supplemented with one embedded Flash memory block that stores configuration data and initializes distributed SRAM cells during power-up (e.g., nonvolatile FPGA devices from Lattice [3]), which is referred to as *fully centralized design style* in this brief. Compared with embedded Flash memory, both PCRAM and MRAM can achieve higher storage density and require fewer mask layers [4].

With respect to using the emerging nonvolatile memory in FPGAs, prior work [5], [6] focused on the fully distributed design style with the use of MRAM. Since PCRAM and MRAM realize data storage by modulating the resistance of each memory cell and full-swing voltage signals are required in FPGA configurations, we have to convert the stored configuration data from resistance state to voltage level. Although this can be easily done in the context of a centralized memory block, such resistance-to-voltage conversion becomes nontrivial when the fully distributed style is being used. As a result, prior work mainly focused on how to address such fully distributed resistance-to-voltage conversion issue.

In [5], each configuration bit is persistently stored in a pair of magnetic tunneling junctions (MTJs) that are programmed differentially, which is used to bias one SRAM cell that converts the differential resistance into a full-swing voltage signal. After the conversion during power-up, each SRAM cell holds the configuration bit and configures the corresponding logic element or interconnect. Apparently, this design approach increases the area overhead for configuration data storage. Aiming at reducing the area overhead, the authors of [6] proposed to use a pair of differential MTJs to form a voltage divider that directly converts the configuration bit from resistance domain to voltage domain. To obtain a full-swing voltage signal at minimal area overhead, architecture of FPGA logic elements is modified so that several MTJ-based voltage dividers can share one SRAM cell. Although such a fully distributed design style enables true instant-on FPGAs, it is subject to two major issues, including: 1) due to the increasingly significant device variability and reduced allowable sensing current as these emerging memory technologies are being scaled down, there may not be sufficient operational margin to ensure reliable resistance-to-voltage conversion, especially when MTJs are directly used to build a voltage divider as in [6] and 2) more importantly, those emerging memory technologies may suffer from relatively high defect density and/or write

Manuscript received January 3, 2011; revised February 15, 2012; accepted April 12, 2012. Date of publication May 11, 2012; date of current version March 18, 2013. This work was supported in part by the National Natural Science Foundation of China, under Grant 61103048, and the National Science and Technology Major Project of China, under Grant 2010ZX01032-001-001-5.

Y. Pan, Y. Li, and T. Zhang are with the Department of Electronics, Computers, and System Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180 USA (e-mail: pany2@rpi.edu; liy16@rpi.edu; tzhang@ecse.rpi.edu).

H. Sun and N. Zheng are with the Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, Xi'an 710049, China (e-mail: sunsir@mail.xjtu.edu.cn; nnzheng@mail.xjtu.edu.cn).

W. Xu is with Marvell Technology, Santa Clara, CA 95050 USA (e-mail: xuwei@marvell.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2012.2195786

failure rate. However, such a fully distributed design style may not be able to use conventional memory fault tolerance techniques to handle the defects and write failures.

In this brief, we focus on the use of MRAM, and the design/evaluation strategy can be readily extended to the case of PCRAM. First, we use VPR [7] to evaluate the area and speed performance of FPGAs with fully distributed MRAM memory cells, which are further compared with conventional SRAM-based FPGAs. Even if we very optimistically estimate the MRAM cell size and ignore any possible memory cell defects, compared with its SRAM-based counterpart, such a fully distributed design approach can result in an area overhead of 14.4%, which also leads to noticeable speed performance degradation. Next, we evaluate the fully centralized design approach, where a single embedded MRAM block is used to persistently store the configuration data and initialize the distributed SRAM cells during power-up. Even assuming the use of error correction codes for memory defect tolerance, this design approach only results in an area overhead of 7.9% and can maintain almost the same speed performance as conventional design practice. Hence, the results suggest that a fully centralized design style is more favorable, especially considering the inevitable significant process variability and nonnegligible defect densities of these emerging memory technologies. Moreover, we note that recent research [8] demonstrated that such centralized design style can also effectively enable run-time dynamic reconfigurability.

Beyond simply replacing embedded Flash memory block in the centralized design style, we propose a new design to replace the SRAM cells with dynamic random-access memory (DRAM) cells to reduce the area and use embedded MRAM to refresh DRAM cells. We elaborate on the analysis of cost induced by periodic on-chip DRAM refresh. We further use the versatile place and route (VPR) tool set to quantitatively evaluate the involved tradeoffs and potential speed performance benefits. Through detailed SPICE simulations and VPR modeling, we show that, for a 45-nm FPGA consisting of 80×30 tiles, such a DRAM-based FPGA design strategy can reduce the FPGA die footprint by up to 8.4%, while its DRAM refresh power consumption is only up to 54.7 mW.

II. Distributed Versus Centralized Design Style

The island-style FPGA logic fabric architecture, being used in latest Xilinx Virtex FPGAs, contains a mesh of reconfigurable logic blocks (LBs) connected through reconfigurable switch boxes (SBs) and connections boxes (CBs). FPGA fabric is conventionally partitioned into an array of tiles, and each tile contains one LB, one SB, two CBs, and associated routing channel [9]. In Virtex devices, each LB contains four slices, and each slice mainly contains two four-input look up tables and two flip-flops. In current design practice, on-chip configuration data storage is realized using either SRAM or embedded Flash memory. SRAM-based FPGAs need at least five or six transistors to store each configuration bit, hence SRAM-based on-chip configuration data storage tends to occupy a significant percentage of the total FPGA die area. Interconnects spanning one, two, four, and six tiles are referred to as single, double, HEX-4, and HEX-6 interconnects, respectively, and we assume the ratio among these four different types of interconnects is 14%:20%:18%:48%. We assume each LB has 16 input and eight output signals, and set the routing channel width as 72 and the connection box connectivity as 36. Using the VPR tool set and setting one SRAM cell size as 9.15 minimum-width transistors, we estimate that SRAM cells for configuration data storage occupy about 33% of the total area.

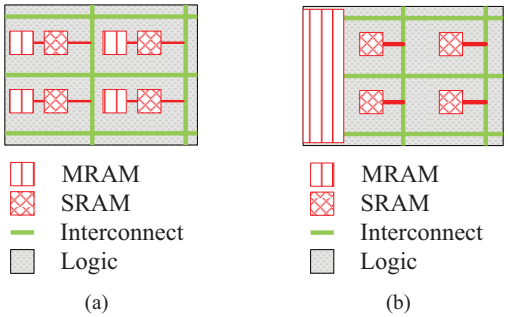

Fig. 1. Illustration of (a) SRAM-based FPGA with fully distributed MRAM and (b) SRAM-based FPGA with fully centralized MRAM.

We present a comprehensive comparison between the fully distributed design style and the fully centralized design style as shown in Fig. 1. Since the fully distributed design strategy presented in [6] is seriously subject to insufficient operational margin for resistance-to-voltage conversion as we push the technology scaling to the limit, we focus on the fully distributed design strategy presented in [5].

A. Distributed Design Style

It is well known that the switching time and write current threshold of a given MTJ are variable and correlated, i.e., as MTJ switching time reduces, the write current threshold will increase. Meantime, to sustain a larger write current threshold, the associated nMOS transistor must have a bigger size. In this brief, the MRAM write latency (i.e., MTJ switching time) is apparently not as important as the MRAM cell size. Therefore, we should use relatively small MRAM cell size at the cost of long MTJ switching time. Based on [10], we very optimistically assume that we can set the MTJ write current threshold as low as 80 μA at the 45-nm node without incurring write failures. Based on PTM transistor model [11] at the 45-nm node, we carried out SPICE simulations, and the results show that the width of the nMOS transistor should be 90 nm (i.e., $W/L = 2$) in order to deliver the 80-μA write current.

Therefore, we can optimistically assume that each configuration bit storage cell as in [5], including one SRAM cell and two differential MTJs with associated two nMOS transistors, has a size as small as $(9.15 + 4) = 13.15$ minimum transistor size. Hence, compared with conventional SRAM-based FPGAs, each configuration bit storage occupies at least 44% more silicon area. In conventional SRAM-based FPGAs, all the SRAM cells for configuration data storage occupy about 33% of the total FPGA die area. Hence, using the fully distributed design style as presented in [5], the FPGA die area will increase at least by 14%. Since all the configuration storage cells uniformly distribute over the entire FPGA die,

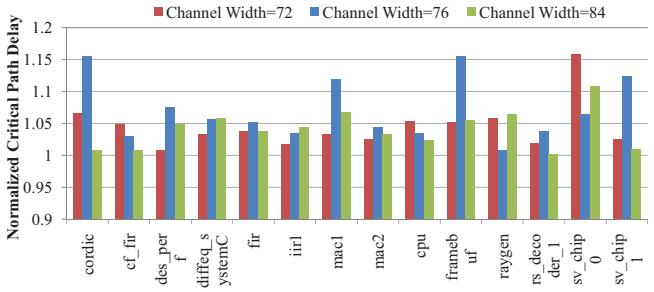

Fig. 2. Normalized critical path delay of FPGA with fully distributed MRAM cells over its counterpart with a fully centralized MRAM block.

such area overhead directly results in larger interconnect length, leading to longer critical path delay.

B. Centralized Design Style

When using the fully centralized design style, we add a single MRAM block on one side of the FPGA die to persistently store all the configuration data and initialize all the distributed SRAM cells during power-up. We modified CACTI [12] to estimate the area, access power, and read latency of embedded MRAM block. In particular, the modification is based on the CACTI DRAM modeling functions because of the cell and sub-array circuits similarity between MRAM and DRAM, i.e., both DRAM and MRAM cells simply contain one data storage element and one nMOS transistor for access control and hence have a three-terminal cell interface. Speed and area performance modeling in CACTI are decomposed into several parts. Clearly, we can keep the modeling of H-tree routing intact. Furthermore, we keep the same modeling of sub-array peripheral decoders and word-lines. In the context of bit-lines and sense amplifiers, we carried out SPICE simulations to estimate their latency using the STT RAM sensing scheme proposed in [13] and predictive technology model (PTM) transistor model [11] at 45-nm node. Since the MTJ tunnel magnetoresistance affects the memory sensing latency, we set the high and low resistance of MTJ as 2 and 1 KΩ, respectively, at 45-nm node according to [14].

In this brief, we set the MTJ write current threshold as 80μA and the nMOS transistor width as 90 nm (i.e., $W/L = 2$), which is the same as the study on fully distributed design style discussed above. In addition, we assume that the embedded MRAM block employs a (71, 64) Hamming code to realize tolerance to possible defective memory cells and random write failures. We assume the FPGA die at the 45-nm node contains an array of 80×30 tiles, which occupies a total 10.45 mm^2 and requires 386-kB configuration data. Using the developed CACTI-based MRAM modeling tool, we estimate that the centralized embedded MRAM block occupies 0.83 mm^2, which increases the FPGA die area by 7.9%. In comparison, as discussed above, the fully distributed design style results in 14% of area increase. More importantly, since the centralized MRAM block locates at the edge of FPGA die, the centralized design style will not increase the interconnect length. In comparison, when using the fully distributed design style, the interconnect length will accordingly increase. As a result, this leads to different speed performance between the fully distributed design style and fully centralized design style. Using the VPR tool set with the benchmarks from Open Cores [15] and internal University of Toronto projects [16], we estimate and compare the critical path delay under different FPGA channel width and the results are shown in Fig. 2. The above results clearly suggest that the fully centralized design style may be a more viable option for using future nonvolatile memory technologies in FPGAs.

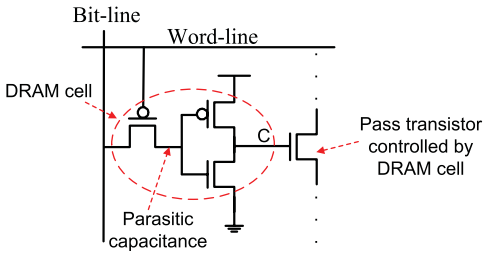

Fig. 3. Illustration of on-chip planar DRAM cell structure based on parasitic capacitance.

III. DRAM-Based FPGA Enabled by Embedded Nonvolatile Memory

SRAM-based FPGA configuration data storage tends to incur a large silicon area overhead. Although, a DRAM cell can be much smaller than an SRAM cell, DRAM cells are not used to realize on-chip FPGA configuration data storage in current design practice due to its destructive self-refresh operation. When an MRAM memory block is embedded to persistently store the configuration data on-chip, intuitively we can use it to periodically refresh distributed DRAM cells in DRAM-based FPGAs. Since we no longer need to explicitly read the on-chip DRAM cells, such DRAM external-refresh will not interrupt the normal FPGA operations. Compared with SRAM-based FPGAs, this DRAM-based FPGA design strategy can reduce the logic element and interconnect silicon area, leading to speed improvement. On the other hand, this design strategy is subject to certain implementation overheads. In particular, the periodic on-chip DRAM cells refresh may incur nonnegligible extra energy consumption. Hence, the refresh circuits and energy cost must be carefully designed and analyzed. These overheads heavily depend on the retention time of on-chip DRAM cells, which further depends on the implementation of DRAM cells. In this brief, we consider the planar DRAM cell implementation strategy as illustrated in Fig. 3.

Because the nMOS pass transistor controlled by each DRAM cell may induce a nonnegligible amount of gate tunneling leakage, we should use an inverter between the storage node and the nMOS pass transistor. The DRAM refresh power consumption is mainly dominated by the power consumed by bit-lines/word-lines, and the planar DRAM structure tends to have low retention time and hence high refresh frequency. Therefore, we may need to use a partially centralized style, in which the FPGA array is partitioned into several regions and each region has its own centralized MRAM block. This can readily accommodate the low retention time and meanwhile reduce refresh power consumption because of relatively shorter bit-lines/word-lines. In this specific case study, we set the nMOS transistor with minimum size, the pMOS transistor with 1.3 $\times$ minimum size, and the pMOS

pass transistor with 4 × minimum size. Hence, according to the area estimation method used in VPR, each DRAM cell has 4.65 minimum-width transistor size. By carrying out further SPICE simulations based on the above configurations, we set the DRAM cell retention time as 100 μs in this case study.

A. On-Chip DRAM Refresh Cost Analysis Methodology

In the following, we first discuss how to estimate the period of each DRAM refresh operation. In this brief, we assume the DRAM cells within each FPGA tile are distributed along the two perpendicular middle lines and four sides of the FPGA tile. Each DRAM cell controls one pass transistor, and all the DRAM cells are connected by bit-lines and word-lines through which the configuration data are written into the DRAM cells.

We model DRAM word-lines and bit-lines as distributed resistor-capacitor networks following the approach used in CACTI modeling tool [12] and estimate τ_{wl} and τ_{bl}, which denote the worst-case delay of word-lines and bit-lines in the absence of buffers.

Let N^w_{buff} and N^b_{buff} denote the number of buffers inserted along each word-line and bit-line, respectively. Let τ_{dec} denote the extra delay induced by word-line decoders, and τ_{buff} denote the delay of each buffer. Let τ_{read} denote the access time to read the data from the centralized MRAM. We can estimate the total time required for one refresh cycle as

$$T_{\text{cycle}} = \frac{\tau_{wl}}{(N^w_{\text{buff}}+1)} + \frac{\tau_{bl}}{(N^b_{\text{buff}}+1)} + (N^w_{\text{buff}} + N^b_{\text{buff}}) \cdot \tau_{\text{buff}} + \tau_{\text{dec}} + \tau_{\text{read}}. \quad (1)$$

As pointed out earlier, in order to meet the DRAM cell retention time constraint and meanwhile reduce the power consumed on bit-lines/word-lines, we could further partition all the DRAM cells on the FPGA die into a certain number of banks, and all the banks are refreshed in parallel. Let N_{bank} denote the number of DRAM banks. Recall that each word-line and bit-line connect with N_w and N_b DRAM cells, hence each bank has N_b word-lines and N_w bit-lines. During each cycle, within each bank all the DRAM cells on one word-line are refreshed and all the banks are refreshed simultaneously, hence the time for refreshing all the DRAM cells once can be estimated as $T = N_b \cdot T_{\text{cycle}}$.

Using the power consumption estimation method in CACTI modeling tool [12], we estimate the word-line energy consumption within each DRAM bank E_w and the bit-line energy consumption during each refresh cycle in each DRAM bank E_b. The inverters within DRAM cells also consume a nonnegligible amount of leakage energy, denoted as E_{leak}, because the inverter input voltage may degrade due to the charge leakage at the storage node. Finally, we should also take into account of the energy consumed when we read the configuration data from the embedded MRAM blocks, which is denoted as E_{read}. Therefore, the overall energy cost can be estimated as

$$E_{\text{cost}} = E_w + E_b + E_{\text{leak}} + E_{\text{read}}. \quad (2)$$

We use a 45-nm FPGA consisting of 80 × 30 tiles as a test vehicle for the purpose of demonstration. The FPGA works at a supply voltage of 1.0 V, and the DRAM word-line/bit-line supply voltage is also 1.0 V. To reduce the DRAM cell leakage current, we use low-power transistors with a high threshold voltage to implement the DRAM cells. The pass nMOS transistor controlled by each DRAM cell (as shown in Fig. 3) is a high-performance transistor with a low threshold voltage in order to minimize the FPGA routing latency overhead. In this brief, we use the low power and BISM4 45-nm PTM transistor models [11], i.e., low-power nMOS and pMOS transistors have threshold voltages of 0.62 V and −0.58 V, respectively, and high-performance nMOS transistors use metal-gate/high-K and have threshold voltage of 0.34 V.

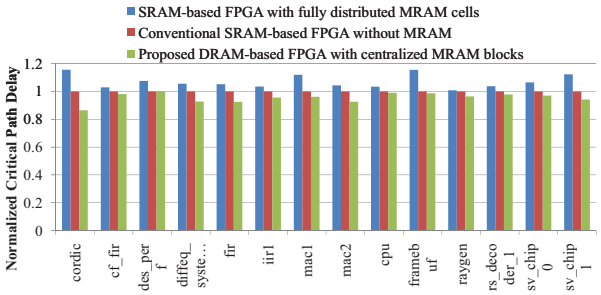

Fig. 4. Normalized critical path delay comparison among different FPGA design strategies.

B. DRAM Refresh Power Consumption

Simulations using the VPR tool set show that each FPGA tile contains 1176 DRAM cells, and each FPGA tile has the size of 66 μm × 66 μm. We put two buffers on each word-line and bit-line. We partition the entire FPGA die into two parts, and each part has its own embedded MRAM. Using the modified CACTI tool, we estimate that the MRAM read access latency and read energy consumption are 1.86 ns and 0.016 nJ per bit and refreshing each word-line takes $T_{\text{cycle}} =$ 16.5 ns, leading to a total 45.4 μs for refreshing all the DRAM cells once. This can well satisfy the 100-μs DRAM cell retention time as estimated above. Assuming P_0 and P_1 are both 0.5, we estimate that a total 4.37 μJ (including energy consumed by word-lines/bit-lines and MRAM memory read) will be dissipated for refreshing all the DRAM cells once. Hence, assuming a DRAM refresh period of 80 μs, the overall DRAM refresh power consumption is 0.0547 W. Regarding the leakage energy consumed by the inverters within DRAM cells, as pointed out above, the static leakage current reaches 36 nA (denoted as I_{leak1}) and 450 pA (denoted as I_{leak0}) if its input voltage increases from 0 to 0.4 V or reduces from 1 to 0.81 V. From our SPICE simulations, the inverter static leakage current increases approximately linearly as its input voltage changes. Hence, the static power consumption for each inverter is *upper bounded* by

$$P_{\text{inv}} = N_{\text{DRAM}}(P_0 I_{\text{leak0}} + P_1 I_{\text{leak1}}) \frac{V_{\text{dd}}}{2}$$

where N_{DRAM} denotes the total number of inverters in the FPGA. Hence, based upon the above parameters, we have that the static power consumed by all the inverters within DRAM cells is upper bounded by 0.0514 W. This leads to a total 0.106-W power consumption cost induced by the use of DRAM-based FPGA.

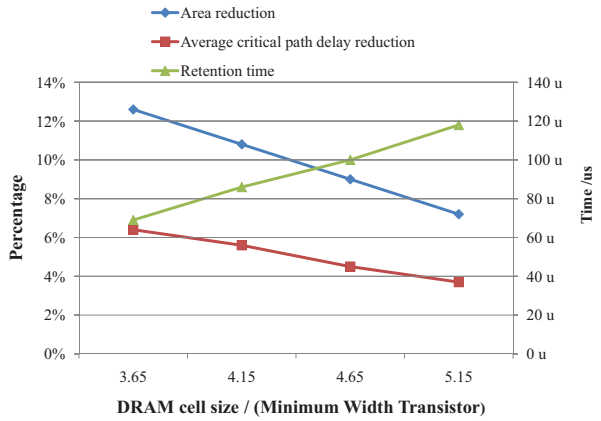

Fig. 5. Estimated FPGA die area and critical path delay reduction versus different DRAM cell sizes.

C. Delay Reduction

Our estimation results show that DRAM-based FPGA enabled by embedded MRAM can reduce the FPGA die area by 23.4, 16.33, and 8.4% compared with the FPGA using the fully distributed design style, SRAM-based FPGA with fully centralized MRAM, and conventional SRAM-based FPGA without MRAM, respectively. Such FPGA footprint reduction can directly translate to interconnect length reduction. Because interconnects play a very important role in determining FPGA speed and power consumption performance, it is reasonable to expect that DRAM-based FPGAs could achieve noticeable gains in terms of FPGA speed and dynamic power consumption compared with their SRAM-based counterparts.

We use the VPR tool set to estimate the critical path delay of DRAM-based FPGA using the benchmarks from Open Cores [15] and internal University of Toronto projects [16]. We use the BISM4 45-nm model and SPICE simulations to estimate the delay characteristics of all these types of devices. Fig. 4 shows the normalized critical path delay compared to the conventional SRAM-based FPGA without MRAM and FPGA, using the fully distributed design style with channel width equal to 76. The results show that the conventional SRAM-based FPGA without MRAM on average has 6% less critical path delay compared with the FPGA using the fully distributed design style. The DRAM-based FPGA can achieve the best critical path delay, which is on average 10 and 4% less than the FPGA, using the fully distributed design style and conventional SRAM-based FPGA, respectively.

The size of transistors within DRAM cells directly determines the tradeoff between FPGA die area reduction and DRAM refresh cost. In the above case study, we choose the DRAM cell size of 4.65 minimum-width transistor size. We further quantitatively investigate how different DRAM cell size may impact this tradeoff. In particular, we adjust the transistor size to obtain three other DRAM cell sizes, including 3.65, 4.15, and 5.15 minimum-width transistor size. We estimate the overall energy cost is 0.132, 0.106, 0.085, and 0.07 W when DRAM cell has a 3.65, 4.15, 4.65, and 5.15 minimum-width transistor size, respectively. Fig. 5 shows the estimated critical path delay reduction and FPGA die area reduction when using different DRAM cell sizes.

IV. Conclusion

Using MRAM as a test vehicle, this brief aimed to carry out a more comprehensive study on exploring the use of emerging resistance-based nonvolatile memory technologies in future FPGAs. First, we evaluated the area and speed performance of FPGAs with either fully distributed MRAM memory cells or a fully centralized MRAM block. The results suggest that the fully centralized design style is a more viable option. Beyond simply replacing embedded Flash memory in the centralized design style, embedded nonvolatile memory could potentially make it more feasible to replace distributed SRAM cells with distributed DRAM cells. We discussed such a DRAM-based FPGA design strategy in detail and analyzed the involved design issues and tradeoffs. Simulation results show that such DRAM-based FPGAs can achieve noticeable silicon area reduction and speed performance improvement at relatively small DRAM refresh cost.

References

[1] K. Kim and G. Jeong, "Memory technologies for sub-40nm node," in *Proc. IEEE Int. Electron Devices Meet.*, Dec. 2007, pp. 27–30.

[2] *Actel FPGA Devices* [Online]. Available: http://www.actel.com/products/devices.aspx

[3] *Lattice Nonvolatile FPGA Devices* [Online]. Available: http://www.latticesemi.com/products/fpga/index.cfm

[4] S. Natarajan, S. Chung, L. Paris, and A. Keshavarzi, "Searching for the dream embedded memory," *IEEE Solid-State Circuits Mag.*, vol. 1, no. 3, pp. 34–44, Aug. 2009.

[5] Y. Guillemenet, L. Torres, and G. Sassatelli, "Non-volatile run-time field-programmable gate arrays structures using thermally assisted switching magnetic random access memories," *IET Comput. Digital Technol.*, vol. 4, no. 3, pp. 211–226, 2010.

[6] S. Paul, S. Mukhopadhyay, and S. Bhunia, "Circuit and architecture co-design approach for hybrid CMOS-STTRAM non-volatile FPGA," *IEEE Trans. Nanotechnol.*, vol. 10, no. 3, pp. 385–394, May 2011.

[7] J. Luu, I. Kuon, P. Jamieson, T. Campbell, A. Ye, W. Fang, and J. Rose, "VPR 5.0: FPGA CAD and architecture exploration tools with single-driver routing, heterogeneity and process scaling," in *Proc. ACM/SIGDA Int. Symp. Field Program. Gate Arrays*, 2009, pp. 133–142.

[8] W. Zhang, N. K. Jha, and L. Shang, "A hybrid Nano/CMOS dynamically reconfigurable system I, design optimization flow," *J. Emerg. Technol. Comput. Syst.*, vol. 5, no. 3, pp. 13:1–13:31, Aug. 2009.

[9] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*. Norwell, MA: Kluwer, 1999.

[10] T. Min, Q. Chen, R. Beach, G. Jan, C. Horng, W. Kula, T. Torng, R. Tong, T. Zhong, D. Tang, P. Wang, M.-M. Chen, J. Z. Sun, J. K. Debrosse, D. C. Worledge, T. M. Maffitt, and W. J. Gallagher, "A study of write margin of spin torque transfer magnetic random access memory technology," *IEEE Trans. Mag.*, vol. 46, no. 6, pp. 2322–2327, Jun. 2010.

[11] *Predictive Technology Model* [Online]. Available: http://www.eas.asu.edu/ptm

[12] *CACTI: An Integrated Cache and Memory Access Time, Cycle Time, Area, Leakage, and Dynamic Power Model*. (2009) [Online]. Available: http://www.hpl.hp.com/research/cacti/

[13] W. Xu, T. Zhang, and Y. Chen, "Design of spin-torque transfer magnetoresistive ram and cam/tcam with high sensing and search speed," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 18, no. 1, pp. 66–74, Jan. 2010.

[14] X. Wang, Y. Chen, H. Li, D. Dimitrov, and H. Liu, "Spin torque random access memory down to 22nm technology," *IEEE Trans. Mag.*, vol. 44, no. 11, pp. 2479–2482, Nov. 2008.

[15] *Open Cores* [Online]. Available: http://www.opencores.org

[16] *Internal University Toronto Projects*. (2002) [Online]. Available: http://www.eecg.utoronto.ca/~jayar/benchmarks/bench.html