Towards Warp-Scheduler Friendly STT-RAM/SRAM Hybrid GPGPU Register File Design

Quan Deng*, Youtao Zhang†, Minxuan Zhang*, Jun Yang‡
*College of Computer, National University of Defense Technolog, Changsha, Hunan, China
†Department of Computer Science, University of Pittsburgh, Pittsburgh, PA, USA
‡Department of Electrical and Computer Engineering, University of Pittsburgh, Pittsburgh, PA, USA
quandeng@pitt.edu, zhangyt@cs.pitt.edu, mxzhang@nudt.edu.cn, juy9@pitt.edu

***Abstract*—Modern Graphics Processing Units (GPUs) widely adopt large SRAM based register file (RF) to enable fast context-switch. A large SRAM RF may consume 20% to 40% GPU power, which has become one of the major design challenges for GPUs. Recent studies mitigate the issue through hybrid RF designs that architect a large STT-RAM (Spin Transfer Torque Magnetic memory) RF and a small SRAM buffer. However, the long STT-RAM write latency throttles the data exchange between STT-RAM and SRAM, which deprecates warp scheduler with frequent context switches, e.g., round robin scheduler.**

In this paper, we propose HC-RF, a warp-scheduler friendly hybrid RF design using novel SRAM/STT-RAM hybrid cell (HC) structure. HC-RF exploits cell level integration to improve the effective bandwidth between STT-RAM and SRAM. By enabling silent data transfer from SRAM to STT-RAM without blocking RF banks, HC-RF supports concurrent context-switching and decouples its dependency on warp scheduler. Our experimental results show that, on average, HC-RF achieves 50% performance improvement and 44% energy consumption reduction over the coarse-grained hybrid design when adopting LRR(Loose Round Robin) warp scheduler.

I. Introduction

Modern Graphics Processing Units (GPUs) widely adopt large SRAM based register file (RF) to enable fast context-switching. A large SRAM RF may occupy more die area than the sum of L1 and L2 cache [11], and consume 20% to 40% GPU power [5]. Due to deteriorating SRAM leakage in sub-micron regime, it becomes less appealing to architect large capacity SRAM based RF in future GPUs, making the design of scalable RF one of the major challenges for GPUs.

Recent studies proposed to construct GPU RFs using STT-RAM (Spin Transfer Torque Magnetic memory) [5], [9], [16], [21]. An STT-RAM cell stores binary information using the resistance of MTJ (Magnetic tunnel junction). Compared with SRAM, STT-RAM has many advantages: it is non-volatile, and of a smaller cell size with low leakage [19]. In addition, it is CMOS-compatible, and thus is ready to be integrated with logic circuits as well as SRAM cells. The recent advance in STT-RAM replaces in-plane MTJ with perpendicular MTJ that has shorter write latency and lower threshold current [2].

Due to long STT-RAM write latency, an STT-RAM based GPU RF design often includes an SRAM buffer to consolidate write operations. The integration is at coarse grained, i.e., a hybrid RF consists of a large STT-RAM RF and a small SRAM buffer. For example, Goswami *et al.* [5] proposed to place an SRAM write buffer before STT-RAM banks such that consecutive instructions can consolidate their register writes. Li *et al.* [9] proposed to use two SRAM buffers to holding the warp contexts. When one buffer is writing the data back to STT-RAM, the other can hold the writes from the new warp.

This work was supported by NSF 1535755, 1617071.

While existing coarse-grained hybrid RF designs achieve significant energy consumption reduction, there exist two design challenges. (1) Due to limited write bandwidth between STT-RAM and SRAM, hybrid RF designs are sensitive to the number of context switches and prefer the warp schedulers that conduct fewer context switches, e.g., the greedy GTO strategy [14]. (2) Hybrid RF designs often improve their write bandwidth using more RF banks. However, such an approach introduces significantly large area overhead [6].

To address these challenges, we propose HC-RF, a hybrid GPU RF design that integrates STT-RAM and SRAM at cell level. We make the following contributions.

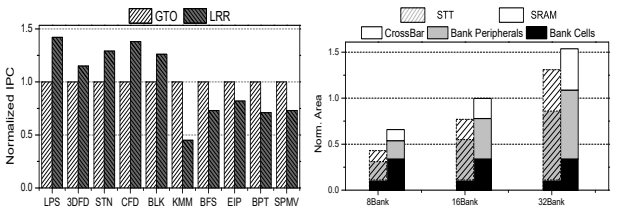
- We propose a novel hybrid cell (HC) structure. An HC cell consists of one area-optimized SRAM cell and one CP-MTJ STT-RAM cell [4]. The HC cell can read and write each component cell independently and, more importantly, support the silent data transfer from SRAM to STT-RAM, i.e., without occupying the bitlines. The HC cell achieves high reliability and low leakage.
- We propose an HC-cell based GPU RF design HC-RF that exploits the silent data transfer to effectively hide the long STT-RAM write latency. By integrating the transfer in the pipeline, HC-RF dynamically allocates write intensive registers in SRAM for performance improvements.
- We evaluate the proposed HC-RF design and compare it to the state-of-the-art. Our experimental results show that on average, HC-RF achieves 50% performance improvement and 44% energy consumption reduction over the coarse-grained hybrid design when adopting LRR warp scheduler.

II. Motivation

In this section, we motivate our HC-RF design by elaborating two design issues in existing hybrid GPU RF architectures.

Warp scheduling dependency. The STT-RAM banks, due to their long write latency, suffer from significant write bandwidth degradation. For example, if the latency of STT-RAM write is 4x that of SRAM write, the write bandwidth of an STT-RAM RF is only 1/4 of that of an SRAM RF if the two RFs use the same number of banks. Existing hybrid GPU RFs alleviate the bandwidth demand with write consolidation — the write buffer in [5] consolidates the register writes if they fall in the same row; the active thread buffer [9] consolidates all register writes from the active thread; compressing contexts [21] also focus on reducing the bandwidth usage.

978-1-5386-3093-8/17/$31.00 ©2017 IEEE



(a)Performance Comparation (b)Area Comparation

Fig. 1. (a) Comparing the application performance using GTO or LRR [7]. (b) The peripheral circuit overhead increases with increasing bank counts [6].

In pratice, write consolidation opportunities are closely coupled with warp scheduler. While the greedy GTO scheduler [14] exhibits good access locality that facilitates write consolidation, the round robin warp scheduler keeps issuing instructions from different warps so that the target registers from consecutively issued instructions may scatter across different banks and rows in the RF. For maximized performance improvement, hybrid GPU RF designs deprecate warp schedulers that have many context switches, e.g., the loose round robin (LRR) scheduler.

However, both schedulers are useful in practice. Lee *et al.* studied GTO and LRR on a set of widely adopted benchmark programs [7]. As shown in Figure 1(a), they found that different programs exhibit different inter-/intra- warp parallelism and thus prefer different schedulers — GTO achieves better performance for only half of the tested programs. With GPUs expanding to more application domains and the recent GPU advances in supporting multiple kernels [18], future GPU workloads are likely to become more mixed and thus demand different schedulers. It becomes increasingly important to decouple the hardware RF design from the choice of warp scheduler.

The large peripheral circuit overhead. As an orthogonal approach to improve the write bandwidth of STT-RAM, a hybrid RF may increase the number of RF banks such that more registers may be accessed simultaneously. For example, most hybrid RF designs use 32 STT-RAM banks instead of 16 banks in the baseline.

Unfortunately, increasing the bank count leads to large area increase. Jing *et al.* revealed that a 32-bank RF is more than 30% bigger than a 16-bank RF [6], as shown in Figure 1(b). The bank area consists of cell array and peripheral circuits (including the crossbar between data banks and operand collectors). With increasing bank count, there is a large area increase from the peripheral circuits, which diminishes the area savings of a hybrid RF in adopting small sized STT-RAM cells in the cell array.

To summarize, our goal is to architect a hybrid GPU RF that has good area and power efficiency, and works well with different warp schedulers. Next we propose fine integration of STT-RAM and SRAM at cell level and design a hybrid GPU RF based on the novel cell structure.

III. The Hybrid Cell

In this section, we first present the novel hybrid cell (HC) design and its operations. We then study its benefits comparing to coarse level integration.

A. The Hybrid Cell (HC)

A hybrid cell (HC) consists of one SRAM cell and one STT-RAM cell, as shown in Figure 2. In the following discussion, the SRAM cell and the STT-RAM cell are referred to as *sub-cells*, or *s-cells* of the hybrid cell. The hybrid cell itself is referred to as *cell* when there is no confusion.

Given that the size of an HC cell is dominated by the SRAM s-cell, we adopt an area-optimized cell design [22] for the SRAM s-cell and the CP-MTJ (complementary-polarizers magnetic tunnel junction) based CP-STT [4], [13] design for the STT-RAM s-cell.

The area-optimized SRAM s-cell (the red box in Figure 2(a)) is one of 4T (four transistors) cell designs that reduce cell size by using fewer transistors. Comparing to the traditional 6T SRAM design, it has similar read and write performance but is much more leaky if replacing the 6T cell directly in a cell array. Our design addresses the issue by connecting it to an STT-RAM cell as we elaborate next.

TABLE I
Basic Operation of HC Hybrid Cell

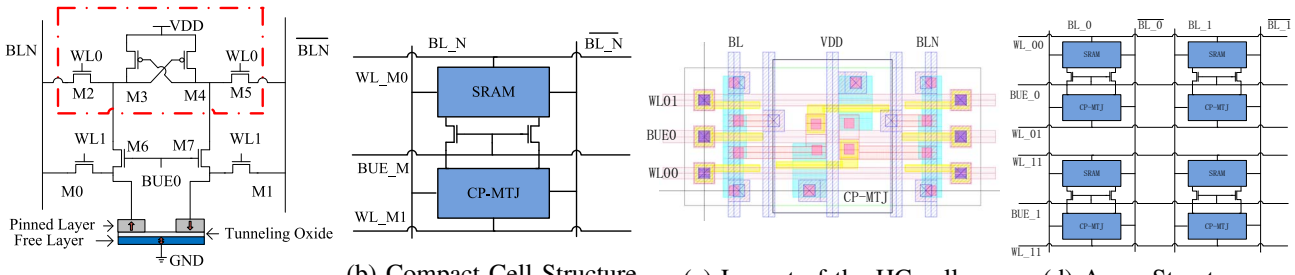
Operation	Signals BL, BLN, WL0, WL1, BUE	Operation Meaning
S-Write	DATA, !DATA, 1, 0, 0	W to SRAM s-cell
S-Read	Pre0, Pre0, 1, 0, 0	R to SRAM s-cell
T-Write	DATA, !DATA, 0, 1, 0	W to NV s-cell
T-Read	Pre0, Pre0, 0, 1, 0	R to NV s-cell
X-Transfer	X, X, X, 1, 0	W from SRAM to NV

The CP-MTJ STT-RAM cell consists of one free layer whose magnetic direction can be switched with injected current, and two pinned layers whose magnetic directions are fixed and set to opposite directions. One CP-MTJ cell stores one bit information using the direction of the free layer — it saves '0' if the free layer is same as the left part the cell; and saves '1' otherwise. CP-MTJ adopts the *self-reference* design whose read operation is to sense the resistance difference between the left and right parts of the cell, which has larger sense margin than that in the traditional cell design that compares to a reference cell [4], [13]. Similar designs have been prototyped using two separate cells [12].

B. The Basic Operations of HC Cell

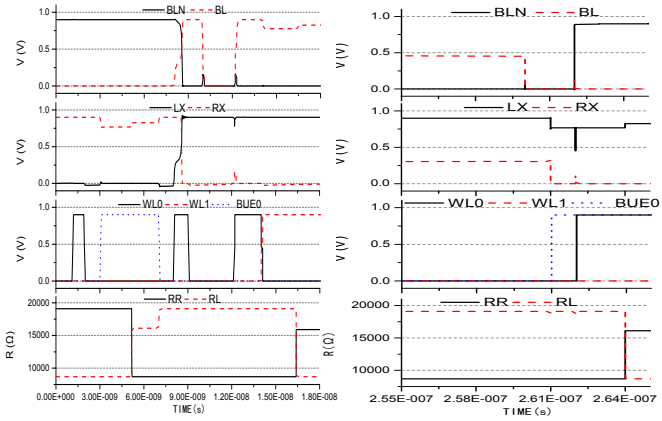
An HC cell array adopts traditional 2D array structure, as shown in Figure 2(d). Either of the two s-cells in each HC cell, since being connected directly to bitlines `BL` and `BLN`, can be accessed independently. For a $M \times N$ HC cell array where M and N are the numbers of rows and columns, respectively, when enabling a wordline $\texttt{WL}_{\texttt{2i}}$ ($0 \leq i < M/2$), we can read or write the corresponding even row that consists of SRAM s-cells. The operations are referred to as `S-Read` and `S-Write`, respectively. When enabling a wordline $\texttt{WL}_{\texttt{2i+1}}$, we can read or write the corresponding odd row that consists of STT-RAM s-cells. The operations are referred to as `T-Read` and `T-Write`, respectively. While `S-Read` and `T-Read` have comparable speeds, `T-Write` is much slower than `S-Write`.

More importantly, the two s-cells within each HC cell are connected locally, which enables moving the stored bit in SRAM s-cell to STT-RAM cell without occupying the bitlines. The enabling signal *BUE* is similar to a wordline. When $\texttt{BUE}_i$ ($0 \leq i < M/2$) is enabled, it couples two rows `row#2i`



(a) Circuit Schematic (b) Compact Cell Structure (c) Layout of the HC cell (d) Array Structure

Fig. 2. The hybrid cell (HC) structure



(a) Five operations (b) S-Read and X-Transfer

Fig. 3. The HC cell operations.

and `row#2i+1` and copies the saved bits from `row#2i` to `row#2i+1` .

Table I lists the five operations together with their control signals. Figure 3(a) presents the waveform of each operation. The results are simulated using HSPICE — the 4T SRAM model is built using 40nm CMOS library from a major chip manufacturer and the CP-MTJ HSPICE model is shared by courtesy of the authors of [13]. From the figure, `T-Write` is about 4X times slower than `S-Write`. Figure 3(b) presents the waveform when performing `S-Read` and `X-Transfer` for the same HC cell. The figure shows that both operations can be accomplished reliably.

C. Benefits from Cell Level Integration

The HC design integrates SRAM and STT-RAM at cell level, which exhibits many benefits that cannot be achieved if adopting cell arrays of either type.

1) Benefit 1: STT-RAM write performance becomes less critical: A major design challenge for choosing STT-RAM in performance critical RFs is to effectively mitigate its long write latency. Optimizations at cell structure level often adopt large access transistors to improve write performance, e.g., the recent 1T1J and 3T3J designs [20]. Unfortunately, this introduces large parasitic capacitance across WLs and BLs such that WL and BL operations, e.g., decode and precharge, not only become slower but also consume more energy.

HC-RF design addresses the challenge with cell level hybrid integration, which moves STT-RAM write related operations off the critical path. For the two related operations `T-Write` and `X-Transfer`, HC-RF rarely uses `T-Write` while `X-Transfer` can operate in parallel to other accesses to the same bank, in particular, Figure 3(b) has shown that `S-Read` and `X-Transfer` can be executed reliably in parallel.

In the design, we choose two small accesses transistors `M0` and `M1` for STT-RAM s-cell, which reduces peripheral power consumption for all STT-RAM related accesses. The slow `X-Transfer` operation is executed silently in the background, i.e., without blocking the RF banks. Given that most MTJ write errors are due to tight write time [15], relaxing STT-RAM write time helps to reduce write errors.

2) Benefit 2: The weak leakage path improves 4T SRAM reliability: While a pure STT-RAM cell array has no cell leakage, there exists a weak leakage path in the HC cell, i.e., when `M3` and `M4` save '1' and '0', respectively, a leakage path from VDD to `M4` and `M7` and to right side of CP-MTJ to GND. A similar leakage path exists in the left side if `M3` saves '0'. The leakage current of the transistor is defined as Equation 1, where I_t, V_t and n are constant values. V_{th} is the threshold voltage, W/L is the geometry size, V_{GS} is the diversity voltage of the gate and the source, V_{DS} is the diversity voltage of the drain and the source.

$$I_{DS} = I_t \frac{W}{L} exp(\frac{V_{GS} - V_{th}}{nV_t})[1 - exp(\frac{-V_{DS}}{V_t})] \quad (1)$$

In our design, we choose high threshold transistors for `M3`, `M4`, `M6`, and `M7` to increase threshold voltage and leveraged the resistance of CPMTJ to improve the source voltage of the closed transistors. The leakage current is effectively decreased in HC cell.

The low leakage helps to stabilize the signals of the SRAM s-cell. A pure 4T SRAM cell holds a weak '0' using gate floating, e.g., when `M4` and `M7` are not connected, which has severe reliability concern as the current from `M4` and `M5` charges the capacitance and raises its voltage. When the voltage reaches VDD-VTH, both of `M3` and `M4` are closed, which destroys the data saved in the 4T SRAM cell. In our design, NMOS transistors are used as the access transistors to transfer '0', while PMOS transistors can only pass a threshold voltage. By choosing PMOS transistors with high threshold voltage, we reduce the leakage current and improve the SRAM reliability when it saves '0'. Comparing to a pure 4T SRAM cell, the HC cell has two bypasses which connect to the ground through CP-MTJ. It increases the capacitance of the storage point and helps to keep the weak '0' at the low voltage level.

3) Benefit 3: `T-Read` *becomes more reliable:* With fast technology scaling, STT-RAM read reliability has become as a major concern. There are two types of read errors – sensing errors and read disturbance errors. STT-RAM has sensing errors because the bits saved in some STT-RAM cells may not be reliably read within the preset sensing time. Sensing errors

closely correlates to the TMR (Tunnel MagnetoResistance) of MTJ.

$$TMR = \frac{R_{ap} - R_p}{R_p}$$

where R_{ap} and R_p are the high and low resistances of the cell, respectively. A typical TMR ranges from 100% to 150%. STT-RAM sensing compares the cell with a reference cell whose resistance is set to the middle of R_{ap} and R_p. Due to the small sensing margin, a small number of cells may not be reliably read within the sensing interval. Recent prototype chips adopted self-reference 2T2J cell designs — [12] compares two cells in opposite resistance states, which effectively double the sensing margin and improves sensing reliability. HC cell adopts self-reference to reduce sensing errors, which mitigates the potential ECC (error correction code) overhead in a pure STT-RAM RF design. We adopt CP-MTJ over 2T2J because the former achieves the same sensing speed with fewer terminals and has lower write power [4], [13].

STT-RAM has read disturbance errors due to shrinking difference between read and write powers under technology scaling. Injecting a small read current may correctly sense the memory cell but has a possibility of destroying the cell during read. HC cell achieves read disturbance free sensing by adopting self-reference cell structure [4], [13]. In our HC cell design, the BL or BLN of the high resistance side becomes '1' after the sense operation. The read current through the CP-MTJ is the same direction with the corresponding write current, while the current on the other side comes to be '0' with only very little current.

D. Cell Comparison

Table II compares the HC cell with 6T SRAM cell and 1T1J STT-RAM cell. For the HC cell, the number in the parentheses are the ones for SRAM s-cells. To save two bits, either SRAM or STT-RAM need two cells; HC structure just needs one cell. The size of the latter is about 33% smaller than that of SRAM, and 70% bigger than that of STT-RAM.

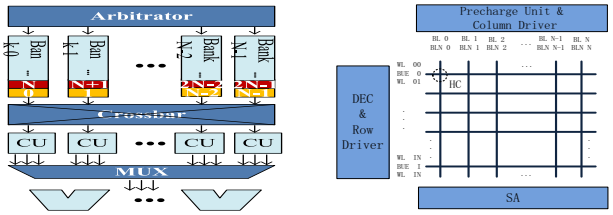
The STT-RAM s-cell in HC structure has lower write energy. This is because (i) CP-MTJ has an optimized write path; (ii) the STT-RAM s-cell chooses small access transistors, which leads to lower dynamic energy consumption in row decoder and row driver.

TABLE II
COMPARISON OF DIFFERENT CELLS

Parameter	SRAM (6T)	STT-RAM (1T1J)	HC cell STT(SRAM)
Cell Factor(F2)	146	57.5	196
Area(mm2)	0.299	0.152	0.216
Write Energy(pJ/bit)	0.186	0.293	0.267(0.186)
Read Energy(pJ/bit)	0.175	0.219	0.225(0.175)
Write Latency(ns)	0.77	3.37	2.8(0.77)
Read Latency(ns)	0.77	1.31	0.71(0.77)
Leakage Power(mW)	187.5	21.1	34.4

IV. THE HC-RF DETAILS

In this section, we construct HC-RF, an HC hybrid cell based GPU RF. The baseline GPU is similar to Nvidia Fermi architecture that consists of multiple streaming multiprocessors (SMs). As shown in Figure 4, the RF in each SM is split to multiple banks that connect to the operand collector units (CUs) through a crossbar. The RF in one SM saves the contexts of multiple warps each of which contains 32 threads. By splitting the RF to multiple banks and using operand collectors, the SM can support concurrent accesses of multiple registers from different banks.



(a) Operand Collector (b) Register File Bank

Fig. 4. The GPU RF structure.

A. Register Mapping

Similar as that in the baseline, HC-RF divides the 128KB RF into 16 banks, with each bank being 64x1024. A single row in each bank has 1024 bits to store the same registers, e.g., `R0`, from all threads of a warp (i.e., 32x32=1024 bits). Register names in warps are mapped to physical addresses in RF banks in sequentially increasing order.

Assume each warp uses 20 registers, `R0...R19` from `warp#0` are mapped to the first row of `bank#0`, ..., `bank#15`, and then the second row of `bank#0`, ..., `bank#3`. Next, `R0...R19` from `warp#1` is mapped to the second row of `bank#4`, ...,`bank#15`, and then the third row of `bank#0`, ...,`bank#7`, and so on. Conceptually, one HC cell spans across two rows so that two consecutive rows, e.g., the first two rows in `bank#0` (for `R0` and `R16` in `warp#0`), are considered as one HC-cell row.

At the device level, all rows with even addresses use SRAM s-cells while all rows with odd addresses use STT-RAM s-cells. By default, HC-RF determines if a register uses SRAM or STT-RAM s-cells based on the last bit of the row index. That is, the write latency of each register is fixed after register mapping.

With this simple register mapping, we have to perform slow STT-RAM writes when accessing half of registers. We expect significant performance degradation from SRAM RF.

B. On-demand Register Remapping

We next discuss the on-demand mapping strategy to exploit the silent transfer capability in HC-cells. Intuitively, we dynamically alter the locations of two registers that map to one HC row such that more writes fall in SRAM s-cells. For the above example that `R1` and `R17` share one HC row. If `R17` is written more frequently than `R1`, we exchange their locations so that `R17` uses SRAM and `R1` uses STT-RAM s-cells. We used one bit tag to indicate if the location has been changed.

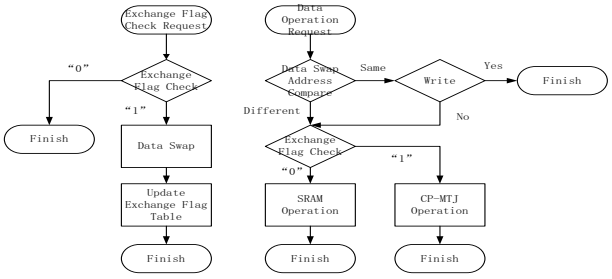


Fig. 5. The on-demand register remapping.

On-demand register remapping. A naive approach to exchange the locations of two registers is to read both registers from their old locations and then write them to new locations. This introduces four extra RF accesses, which block the RF banks for significant amount of time and tend to have large performance degradation. In HC-RF, we propose on-demand register remapping with minimal impact to normal RF operations.

An on-demand register remapping is always triggered by a `T-Write`, i.e., a write operation that needs to save data to the STT-RAM s-cells. The work flow is shown in Figure 5. For the above example, writing `R17` triggers the remapping. Given that `T-Write` operation has new data, i.e., new `R17`, the contents in STT-RAM s-cells are obsolete and can be overwritten. Scheduling a `X-Transfer` helps to write the contents of `R0` from SRAM to STT-RAM s-cells. Then we write the new `R17` to SRAM s-cells, which completes the location exchange. The data transfer from SRAM to STT-RAM uses the internal data path instead of the bitlines, which does not block the RF bank.

We attach a one-bit flag to each HC-row (i.e., two device rows). It records if the two registers mapped to this row has exchanged their locations. After remapping, future writes to `R17` are redirected to write SRAM rather than STT-RAM s-cells. By adopting on-demand remapping, frequently accessed registers are remapped to SRAM s-cells such that the average register access latency can be greatly reduced.

While speeding up frequently accessed registers, on-demand register remapping slows down the current access. As shown in Figure 6, for the current STT-RAM access, instead of one long latency `T-Write`, we need to check the exchange flag, finish `X-Transfer`, and then write new data to SRAM s-cells using `S-Write`. This is much slower and may degrade performance if the two registers are used alternatively. As an extreme, a round robin warp scheduler may trigger the remapping every time it schedules an instruction from a warp.

Pipeline integration. To decouple GPU RF design from warp scheduling, we propose to integrate the transfer within GPU pipeline by taking advantage of its non-blocking transfer capability.

The GPU SM core adopts a 6-stage pipeline, i.e., there are fetch, decode, issue, read operands, execute, and write back stages (Figure 6). An instruction that has two read operands needs two cycles to finish the operate read stage.

We start remapping as early as in the third stage rather than in the write back stage in the baseline. This is because, in the third stage, the decoded instruction stored in the I-Buffer has the target register address. HC-RF initiates on-demand register remapping if the following two conditions hold. (1) If the target register uses STT-RAM cells. By checking the exchange flag and the last bit of the physical row address, we determine if the target register use STT-RAM s-cells. (2) If it is safe to remap. If there are no conflicting accesses to the involved registers, that is, the involved two rows are not being read or written or swapped. To actually swap the register locations, we initiate `X-Transfer` to transfer SRAM contents to STT-RAM; and then write new register contents to SRAM s-cells.

C. Hardware Support

To enable on-demand register remapping, HC-RF enhances the instruction issue logic with exchange flag check(EFC).

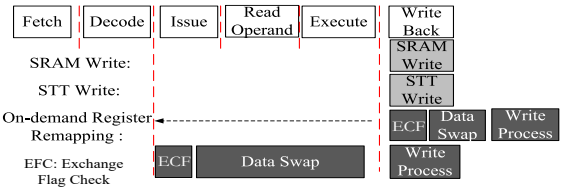


Fig. 6. Integrating silent transfer in the pipeline.

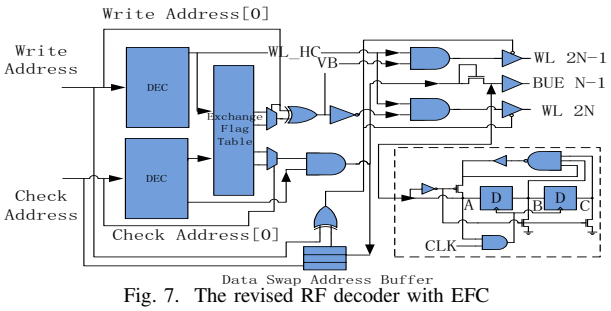


Fig. 7. The revised RF decoder with EFC

As shown in Figure 7, we organize the exchange flags for each RF bank as a small table, and add two ports (one read port and one read/write port) to support the following two operations at the same time: (1) `R-check`: we use the read port to check the flag before operand read so that the correct register location can be found; (2) `X-check`: we use the read/write port to check if the target register of the issued instruction is saved in the STT-RAM s-cells and, if there is no conflict ongoing operation, triggers early silent data transfer `X-Transfer`. In Figure 7, we use two decoders to decode `R-check` and `X-check` addresses independently. We XOR the last bit of `R-check` address and its exchange flag to determine its location, which then drives the corresponding wordline to read the register. For `X-check`, if the XOR result is '1', we active `BUE` line to trigger silent data transfer `X-Transfer`. After the transfer, we flip the bit in the exchange table.

V. The Evaluation

To study the effectiveness of our proposed HC-RF design, we compared different hybrid GPU RF designs on a GPU that is similar to Nvidia Fermi GTX480. The simulation of HC-RF is accomplished at two levels. At the cell level, we evaluated the read and write performance, the power consumption, and the reliability of HC-cells using HSPICE models — the SRAM was built using the CMOS HSPICE model from a major chip manufacturer while the CP-MTJ STT-RAM HSPICE model is shared by courtesy of the authors of [13]. At the architecture and system level, we used GPGPU-sim [1], GPUWattch [8] and NV-sim [3]. Table 2 lists the configuration details. Table III lists the setting details.

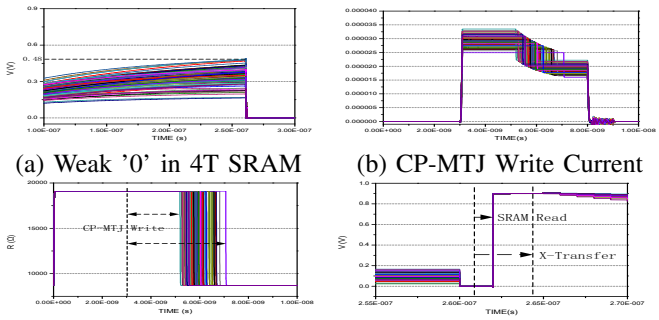
We compiled a set of widely used GPU benchmarks, including BFS (Breadth-First Search), CP (Coulombic Potential), MUM (MUMmerGPU), NN (Neural Network), NQU (N-Queens Solver), RAY (Ray Tracing), STO (StoreGPU), WP (Weather Prediction), LIB (LIBOR Monte Carlo), LPS (3D Laplace Solver). The details can be found in [1].

In the experiments, we evaluated the following schemes.

- **SRAM.** It denotes the SRAM based RF design. By default, it has 16 banks.
- **STT.** It denotes the pure STT-RAM based RF design.

TABLE III
THE GPU AND HC CONFIGURATION

GPU Architecture	Fermi	CMOS Tech.	40nm Bulk
Core Freq. (MHz)	70	CP-MTJ Tech.	Perpendicular
#. of SM	15	Free Layer Size	$45\times20\times1$ nm^3
Cores per SM	32	Critical Current	15.1 μA
RF Size(KB)	128	Parallel Resistance	11.9 KΩ
Max Warp per SM	48	Bank Size	16
Warp Scheduling	LRR	VDD(V)	0.9



(a) Weak '0' in 4T SRAM (b) CP-MTJ Write Current

(c) CP-MTJ Resistance Switch (d) X-Transfer

Fig. 8. The HC cell reliability.

- **HRF.** It denotes a coarse grained STT-RAM/SRAM hybrid design.
- **HC-RF.** It denotes the cell level hybrid integration in the paper. We adopt on-demand register remapping such that long STT-RAM write operation starts early and does not block RF banks.

A. The Hybrid Cell Reliability

To study the HC cell reliability under process variation (PV), we conducted Monte Carlo simulation covering all corner cases. Figure 8 summarizes the results which has no error in 1000 local Monte Carlo simulations, the similar approach as that in [20].

We first checked if the weak '0' in SRAM s-cell may be destroyed by leakage current under PV. Figure 8(a) shows the peak voltage of the weak '0' after enough time charging. From the figure, the peak voltage is about 0.48V, which is lower than the threshold voltage (VDD-VTH) of PMOS. Therefore, the value is reliable and there is no need to refresh the SRAM s-cell.

We then studied the relationship of write current and switch time for CP-MTJ and summarized the results in Figure 8(b)(c). The write current varies from 25uA to 35uA while the write delay varies from 2ns to 4ns. The range of CP-MTJ switch time is proportional to the write current at the TT corner.

Given silent `X-Transfer` move STT-RAM write off the critical path, it is safe to choose HC-RF with low write performance and large write margin, which reduces write error probability. We observed no write error in the simulation.

The last study that we performed was to check simultaneous `S-Read` and `X-Transfer`. Figure 8(d) shows `X-Transfer` reliability. The narrow depression in the middle is caused by the precharge operation. The data in 4T SRAM is preserved during BL read operation.

To summarize, our HC cell structure exhibits good reliability on both write and read operations under PV.

B. Area Reduction

Figure 9 compares the RF area using different schemes. The RF area consists of cell arrays, crossbar, and peripheral

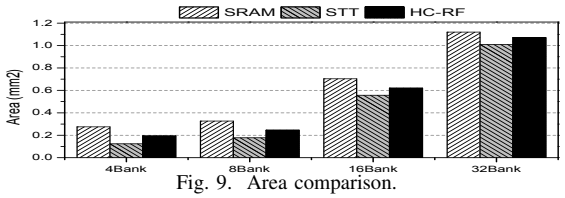


Fig. 9. Area comparison.

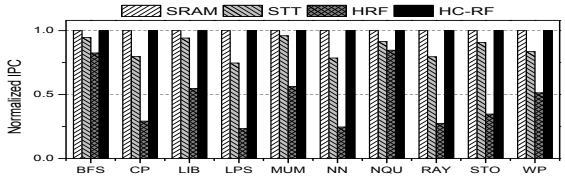


Fig. 10. Performance comparison.

circuits. From the figure, STT-RAM based GPU RF is smaller than SRAM based RF. This is because STT-RAM RF has a smaller cell array — an STT-RAM cell is about 33% of an SRAM cell. When using 4 banks, the size of STT-RAM RF is smaller than half of that of SRAM RF. The difference diminishes as the bank count increases. When using 32 banks, due to the large area overhead of crossbar and pheripheral circuits, the difference is less than 10%. Another observation is that a 32-bank STT-RAM RF is larger than a 16-bank SRAM RF.

By default, HC-RF uses 16 banks, the same as the baseline. The total area is about 88% of SRAM based RF. HC-RF has smaller cell array area due to the adoption of 4T-SRAM s-cell and CP-MTJ STT-RAM s-cell. However, it has large peripheral circuit for STT-RAM and the additional circuits to enable on-demand register remapping. For the latter, the logic is simple, which demands 32 decode units and an 8B exchange flag table for each bank. The total overhead area of the additional peripheral circuits is less than 2% of that of RF cell arrays.

Depending on the size of SRAM buffer integrated for performance optimization, the area of coarse-grained GPU RF designs is between that of SRAM RF and that of STT-RAM RF. For the RFs with the same number of banks, HC-RF is of the similar RF area.

C. Performance Reduction

We next compared the performance when adopting different RF designs. Figure 10 summarizes the normalized IPC when adopting LRR warp scheduler. In addition to SRAM RF and STT-RAM RF, we compared HC-RF with a coarse grained hybrid GPU RF deisgn in [9]. The latter is referred to as `HRF` in the figure.

From the figure, `STT-RAM` exhibits an average of 15% performance degradation over `SRAM`. The long write operation of STT-RAM not only degrades bank bandwidth, but also introduces more bank conflict. `HRF`, while showing a modest of 2% performance degradation when adopting GTO warp scheduler, exhibits 50% performance degradation when adopting LRR. This is because frequent context switch quickly saturates the SRAM buffer, after which it creates additional bottleneck on SRAM buffer. While waiting for STT-RAM operations to complete, `STT-RAM` may issue warps that use different banks. `HRF` has to wait for the SRAM buffer before issuing more warps. That is, due to the limited bandwidth between SRAM and STT-RAM, coarse-grained hybrid GPU RF designs, such as `HRF`, are closely coupled with the choice of warp schedulers. Given both GTO and LRR are valuable in

practice [7], future hybrid GPU RF designs prefer to decoupled designs.

HC-RF shows negligible performance degradation from SRAM. This is because while STT-RAM write operations are still slow, they are moved from the critical execution path. By enabling silent register remapping, the STT-RAM write operations do not block RF banks such that additional warp instructions can be scheduled to any bank.

When adopting GTO warp scheduler (not shown due to space limit), HRF exhibits 2% to 4% degradation from SRAM [9] while HC−RF still has negligible performance degradation.

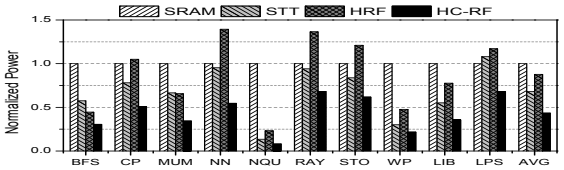


Fig. 11. Power comparison.

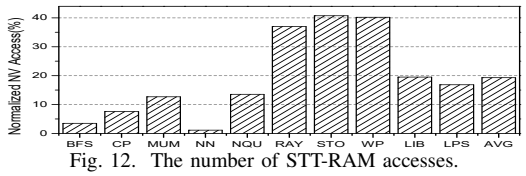


Fig. 12. The number of STT-RAM accesses.

D. Power Reduction

Figure 11 reports the power reduction using different GPU RFs. Figure 12 reports the normalized number of accesses to STT-RAM. From the figures, we observed that the dynamic energy consumption is closely coupled with the number of accesses to STT-RAM. This is because writing STT-RAM consumes 1.6x SRAM power. The larger the number of STT-RAM writes is, the larger the dynamic power consumption is. For example, for benchmark LPS, the dynamic power is 59.9% of the total power, the power consumption of STT-MRAM is larger than that of SRAM. On average, the power consumption of HC-RF is 56% of SRAM RF.

E. Design Efficiency

TABLE IV
EDA

	SRAM	STT	HC
Normalized Power Efficiency	1	1.47	2.27
Normalized Performance	1	0.75	1
Normalized Area Efficiency	1	1.26	1.13
EDA(Efficiency)	1	1.39	2.56

To summarize, Table IV compares the design efficiency of different schemes. The design efficiency is defined as

$$EDA = E \times D \times A \tag{2}$$

where E, D, A are normalized energy consumption, delay, and RF area. From the table, we found that HC-RF achieves 256% and 184% improvements over SRAM and STT−RAM, respectively.

F. Related Work

There are many hybrid memory designs in GPUs. Li *et al.* [9], builds a distribute register file system based on STT-RAM and SRAM. Goswami *et al.* [5], changes all the memory of GPU into STT-RAM. Zhang *et al.* [21], introduces a centralized SRAM based buffer and a light-weight compression framework. However, they are coarse-grained hybrid design, which relies on specific warp scheduling and ignores the overhead of peripheral circuits like crossbar.

For fine-grained hybrid memory, Wang *et al.* [17], and Liao *et al.* [10], combine the advantages of SRAM and MTJ for low leakage power and performance improvement. Fong *et al.* [4], use the CP-MTJ instead of MTJ as on-chip cache, which shows good write and read performance. Qu *et al.* [13], make further optimization on reliability and power. In consideration of that, we build a warp-scheduler friendly HC-RF based on SRAM and CP-MTJ with a silent data transfer.

VI. Conclusion

In this paper, we proposed HC-RF, an efficient GPU RF design based on the novel STT-RAM/SRAM hybrid cell. A cell integration effectively enlarges the bandwidth between STT-RAM and SRAM. By enabling silent data transfer from SRAM to STT-RAM, HC-RF decouples the RF design from the choice of warp schedulers. Comparing to coarse-grained STT-RAM designs, HC-RF exhibits negligible performance degradation when choosing either GTO or LRR.

References

[1] A. Bakhoda, *et al.*, "Analyzing CUDA Workloads Using A Detailed GPU Simulator," in *ISPASS*, 2009.

[2] S.-W. Chung, *et al.*, "4Gbit Density STT-MRAM Using Perpendicular MTJ Realized with Compact Cell Structure," in *IEDM*, 2016.

[3] X. Dong, *et al.*,, "NVSim: A Circuit-Level Performance, Energy, And Area Model For Emerging Nonvolatile Memory," *TCAD*, 31:994–1007, 2012.

[4] X. Fong, *et al.*,, "Complementary Polarizers STT-MRAM (CPSTT) for On-Chip Caches," *EDL*, 34, 2013.

[5] N. Goswami, *et al.*, "Power-performance Co-optimization Of Throughput Core Architecture Using Resistive Memory," in *HPCA*, 2013.

[6] N. Jing, *et al.*,, "Bank Stealing For Conflict Mitigation In GPGPU Register File," in *ISLPED*, 2015.

[7] M. Lee, *et al.*,, "iPAWS: Instruction-issue Pattern-based Adaptive Warp Scheduling For GPGPUs," in *HPCA*, 2016.

[8] J. Leng, *et al.*, "GPUWattch: Enabling Energy Optimizations In GPGPUs," in *ISCA*, 2013.

[9] G. Li, *et al.*, "A STT-RAM-based Low-power Hybrid Register File For GPGPUs," in *DAC*, 2015.

[10] C.-F. Liao, *et al.*,, "Zero Static-power 4T SRAM with Self-inhibit Resistive Switching Load by Pure CMOS Logic Process," in *IEDM*, 2016.

[11] S. Mittal, "A Survey Of Techniques For Architecting And Managing GPU Register File," *TPDS*, 28:16–28, 2017.

[12] H. Noguchi, *et al.*, "A 3.3 ns-access-time 71.2μW/MHz 1Mb Embedded STT-MRAM using Physically Eliminated Read-disturb Scheme and Normally-off Memory Architecture," in *ISSCC*, 2015.

[13] L. Qu, *et al.*, "A Disturbance-Free Energy-Efficient STT-MRAM Based on Complementary Polarizers," *EDL*, 37, 2016.

[14] T. G. Rogers, *et al.*, "Cache-Conscious Wavefront Scheduling," in *MICRO*, 2012.

[15] U. Roy, *et al.*, "Write Error Rate of Spin-Transfer-Torque Random Access Memory Including Micromagnetic Effects Using Rare Event Enhancement," *TM*, 52, 2016.

[16] M. H. Samavatian, *et al.*, "An Efficient STT-RAM Last Level Cache Architecture For GPUs," in *DAC*, 2014.

[17] J. Wang, *et al.*, "cNV SRAM: CMOS Technology Compatible Non-Volatile SRAM Based Ultra-Low Leakage Energy Hybrid Memory System," *TC*, 65:1055–1067, 2016.

[18] Z. Wang, *et al.*, "Simultaneous Multikernel GPU: Multi-tasking Throughput Processors Via Fine-grained Sharing," in *HPCA*, 2016.

[19] X. Wu, *et al.*, "Hybrid Cache Architecture With Disparate Memory Technologies," in *ISCA*, 2009.

[20] L. Xue, *et al.*,,"ODESY: A Novel 3T-3MTJ Cell Design With Optimized Area DEnsity, Scalability And LatencY," in *ICCAD*, 2016.

[21] H. Zhang, *et al.*, "Architecting Energy-efficient STT-RAM Based Register File On GPGPUs Via Delta Compression," in *DAC*, 2016.

[22] L. Wei and K. Zhang, "Static random access memory," in *US Patent*, 2005.