RESEARCH ARTICLE OPEN ACCESS

Various RAM Technologies in the Reduction of Static and Dynamic Power: A Survey

P. Lakshmi Priya[1], R. Raj Kumar[2]
ME Research Scholar[1], Assistant Professor[2]
ECE Department,
SVS College of Engineering, Coimbatore
Tamil Nadu - Chennai

ABSTRACT
Power consumption is becoming one of the most important constraints in the VLSI field in nano-meter scale technologies. Especially, as a transistor for supply voltage and threshold voltage are scaled down, leakage energy consumption is increased even when the transistor is not switching. This paper proposes to use various techniques in reducing the static power as well as dynamic power consumption by using different cache techniques and various types of RAM.
Keywords:- Caches, instruction caches, loop caches, low power, power gating, spin-transfer torque RAM (STT-RAM), power efficiency, static power and dynamic power.

I. INTRODUCTION

Magnetic Random Access Memory (MRAM) has been considered as one of the most promising universal memory technologies due to its non-volatility, fast speed, zero standby power, and high density. Over the past two decades, the CMOS microprocessor design process has been confronted by a number of seemingly insurmountable technological challenges (e.g., the memory wall and the wire delay problem). At each turn, new classes of systems have been architected to meet these challenges, and microprocessor performance has continued to scale with exponentially increasing transistor budgets. With more than two billion transistors integrated on a single die, power dissipation has become the current critical challenge facing modern chip design. On-chip power dissipation now exhausts the maximum capability of conventional cooling technologies; any further increases will require expensive and challenging solutions (e.g., liquid cooling), which would significantly increase overall system cost. Multicore architectures emerged in the early 2000s as means of avoiding the power wall, increasing parallelism under a constant clock frequency to avoid an increase in dynamic power consumption. Although multicore systems did manage to keep power dissipation at bay

for the past decade, with the impending transition to 32nm CMOS, they are starting to experience scalability problems of their own. To maintain constant dynamic power at a given clock rate, supply and threshold voltages must scale with feature size, but this approach induces an exponential rise in leakage power, which is fast approaching dynamic power in magnitude. Under this poor scaling behavior, the number of active cores on a chip will have to grow much more slowly than the total transistor budget allows; indeed, at 11nm, over 80% of all cores may have to be dormant at all times to fit within the chip's thermal envelope .

This paper presents resistive computation, an architectural technique that aims at developing a new class of power- efficient, scalable microprocessors based on emerging resistive memory technologies. Power- and performance-critical hardware resources such as caches, memory controllers, and floating-point units are implemented using spin-torque transfer magneto resistive RAM (STT-MRAM)—a CMOS- compatible, near-zero static-power, persistent memory that has been in development since the early 2000s and is expected to replace commercially available magnetic RAMs by 2013.

II. 3D STACKING MAGNETIC RAM (MRAM)

A. MRAM CIR4CUIT DESIGN

This section introduces the physical mechanism of MRAM, and discusses the circuit design for MRAM.

1) Fundamental of MRAM

MTJ (Magnetic Tunnel Junction) is the storage element of MRAM cells. Normally, there are two ways to form an MRAM cell – the cross-point ("XPT") structure and one transistor, one MTJ ("1T1J") structure [2, 3]. Because current XPT structure has very poor read performance[1].

For a 1T1J MRAM cell, as illustrated in Fig. 1, each MTJ is connected in series with a NMOS. The gate of the NMOS is connected to the word line (WL), and the NMOS is turned on if its connected MTJ needs read or write operations. As shown in Fig. 1, the source of the NMOS is connected to the source line (SL), and the free ferromagnetic layer is connected to the bit line (BL). The read and write operations are explained as follows:

• Read Operation

On a read operation, a negative voltage difference is applied on BL relative to SL . This negative voltage is usually very small, which is -0.1V in our design. This voltage difference will lead to a current passing through the MTJ, which is small enough and will not invoke a disturbed write operation? The value of the current is mainly dependent on the resistance of the MTJ. Finally, a sense amplifier compares this current with a reference current and then decides whether a "0" or a "1" is stored in the selected MRAM cell.

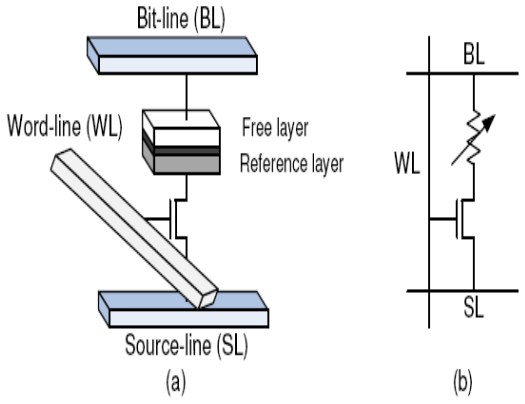

Fig 1.Demonstration of a MRAM cell (a) structural view (b) schematic view

• Write Operation

When writing "0" state into MRAM cells, positive voltage difference is established between SL and BL; when writing "1" state, vice versa. The current amplitude required to reverse the direction of the free ferromagnetic

III. AVOIDING THE POWER WALL WITH LOW-LEAKAGE, STT-MRAM BASED COMPUTING

A. Memory Cells and Array Architecture

STT-MRAM relies on magneto resistance to encode information. Figure 1 depicts the fundamental building block of an MRAM cell [1], the magnetic tunnel junction (MTJ). An MTJ consists of two ferromagnetic layers and a tunnel barrier layer, often implemented using a magnetic thin-film stack comprising Co40Fe40B20 for the ferromagnetic layers, and MgO for the tunnel barrier.

One of the ferromagnetic layers, the pinned layer, has a fixed magnetic spin, whereas the spin of the electrons in the free layer can be influenced by first applying a high-amplitude current pulse through the pinned layer to polarize the current[9], and then passing this spin-polarized current through the free layer.The spin polarity of the free layer on the direction of the current, can be made either parallel or anti-parallel to that of the pinned layer.

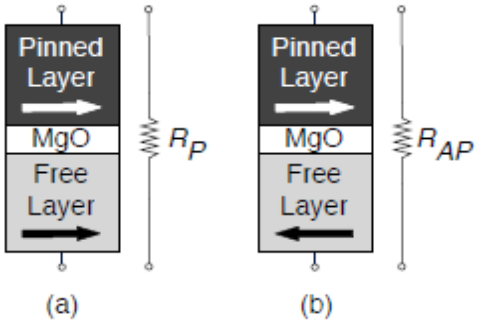

Fig 2.Illustrative example of a MagnetoTunnel Junction(MTJ) in (a) Low-resistance parallel and (b) High-resistance antiparallel states

Applying a small bias voltage (typically 0.1V) across the MTJ causes a tunneling current to flow through the MgO tunnel barrier without perturbing the magnetic polarity of the free layer and thus, the resistance of the MTJ—is determined by the polarity of the two ferromagnetic layers: a lower, parallel resistance (RP in Figure 2) state is experienced when the spin polarities agree[8], and a higher, anti parallel resistance state is observed when the polarities disagree (RAP in Figure 2). When the polarities of the two layers are aligned, electrons with

polarity anti-parallel to the two layers can travel through the MTJ easily, while electrons with the same spin as the two layers are scattered; in contrast, when the two layers have anti-parallel polarities, electrons of either polarity are largely scattered by one of the two layers, leading to much lower conductivity, and thus, higher resistance .These low and high resistances are used to represent different logic values.

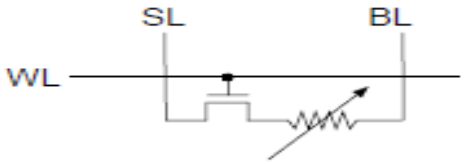

Fig 3. Illustrated example of a 1T-1MTJ cell

An MTJ [5] can be written in a thermal activation mode through the application of a long, low-amplitude current pulse (>10ns), under a dynamic reversal regime with inter-mediate current pulses (3-10ns), or in a processional switching regime with a short (<3ns), high-amplitude current pulse .In a 1T-1MTJ cell with a fixed-size MTJ, a tradeoff exists between switching time (i.e., current pulse width) and cell area. In processional mode, the required current density Jc(t) to switch the state of the MTJ is inversely proportional to switching time _

$$J_c(t) \propto J_{c0} + C/t$$

Where Jc0 is a process-dependent intrinsic current density parameter and C is a constant that depends on the angle of the magnetization vector of the free layer. Hence, operating at a faster switching time increases energy-efficiency: a 2× shorter write pulse requires a less than 2× increase in write current, and thus, lower write energy.

TABLE I. STT-MRAM parameters at 32nm based on ITRS'09 projections

Parameter	value
Cell size	$10F^2$
Switching current	$50\mu A$
Switching time	6.7ns
Write energy	0.3pJ/bit
MTJ,Resistance(R_{Low} /R_{High})	2.5kΩ / 6.25kΩ

IV. INSTRUCTION FETCH ENERGY REDUCTION USING LOOP CACHES

A. Loop Cache Organization:

Figure 4 shows the organization of a 2W-entry loop cache and how it is being accessed with an instruction address A [31:0]. A loop cache does not have an address tag store. The loop cache array can be implemented as a direct mapped Array. It is indexed using the index (A) field of the instruction address. The index (A) field is w-bit wide. The array can store 2w instructions. By definition of a sbb, the maximum program loop size that can be recognized and captured by the loop cache is ***2w*** instructions. Thus accessing the loop cache during program loop execution **is** guaranteed to be unique and non-compacting. That is: (i) an instruction in the program loop will always be mapped to a unique location in the loop cache array; and (ii) there could never be more than one instruction from a given program loop to compete for a particular cache location. When the program loop being captured **is** smaller than the loop cache size, only part of the loop cache array is utilized.

Unlike many other loop caching schemes, our loop caching scheme does not require the program loop to be aligned to any particular address boundary. The software can place a loop at any arbitrary starting address.

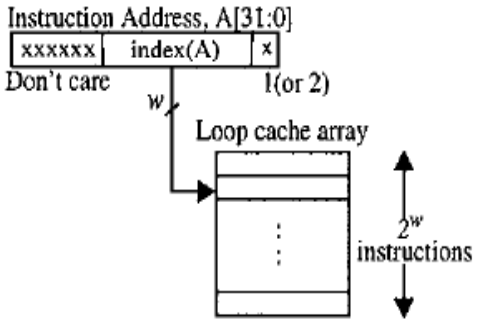

Fig 4. Loop cache organisation

V. DETERMINING LOOP CACHE HIT/MISS EARLY

In order to determine in advance, whether the next instruction fetch will hit in the loop cache, the controller needs the following information on a cycle-to-cycle basis: (a) **is** the next instruction fetch a sequential fetch or is there a change of control flow (cot)? (b) if there is a cof, is it caused by the triggering sbb? (c) is the loop cache completely warmed up with the program loop so we could access

the loop cache instead of the main cache? Information pertaining to (a) can be easily obtained from the instruction decode unit as well as the fetch and branch unit in the pipeline.

A. Loop cache controller

The state transition diagram for the loop cache controller is show in Figure5. The controller ensures that the loop caches warmed up before it is being utilized. It begins with an IDLE state. When a sbb is decoded, its *Id* field is loaded into the Count-Register. If the sbb is taken, the controller enters a FILL state. The sbb becomes the triggering sbb. When inthe ELL state, the controller **fills** the loop cache with the instructions being fetched from the main cache. As the value in the Count-Register increases and becomes zero, the controller knows that the instruction currently being executed is the triggering sbb. If the triggering sbb is not taken, the controller returns to the IDLE state. Otherwise, it enters an ACTIVE state. While in the FILL state, if there is a cof that is not caused by the triggering sbb, the controller also returns to the IDLE state

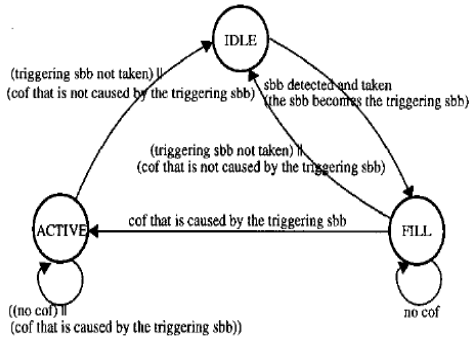

Fig 5. Loop cache controller

When in the ACTIVE state, the controller directs all the instruction requests to the loop cache. When in the ACTIVE state, [12] the controller will returns to the IDLE state if one of the following two events occurs: (i) the triggering sbb is not taken (the loop sequentially exits through the sbb); or (ii) there is a cof that is not caused by the triggering sbb.

VI. COMPARISON OF VARIOUS CACHES

A. Drowsy caches

The key difference between drowsy caches and caches that use gated-VDD is that in drowsy caches the cost of being wrong—putting a line into drowsy mode that is accessed soon thereafter—is relatively small. The only penalty one must contend with is an additional delay and energy cost for having to wake up a drowsy line. One of the simplest policies that one might consider is one where, periodically, all lines in the cache—regardless of access patterns—are put into drowsy mode and a line is woken up only when it is accessed again.

B. Reconfigurable hybrid cache:

We evaluate the proposed RHC design on a simulation platform built upon Simics with GEMS. Table III shows the parameters used in our model. The value K represents the number of cache ways that are powered on in a specific L2 cache configuration [4], which also equals the amount of "active" cache associativity. Notice that the configuration of the processor core, L1 caches, and main memory remains the same through all simulations.

TABLE II. Comparison of various memory technologies

Features	SRAM	MRAM	PRAM
Non-volatility	No	Yes	Yes
Leakage Power`	High	Low	Low
Dynamic Power	Low	Low for read Very high for write	Medium for read High for write
Density	Low	High	Very high
Speed	Very fast	Fast for read Slow for write	Slow for read very slow for write
Scalability	Yes	Yes	Yes

TABLE III. Comparison of STT-RAM with SRAM, PRAM

	SRAM	STT-RAM	PRAM
Density	1X	4X	16X
Read time	Very fast	Fast	Slow
			Very

Write time	Very fast	Slow	slow
Read power	Low	Low	Medium
Write power	Low	High	High
Leakage power	High	Low	Low
Endurance	10^{16}	4×10^{12}	10^{9}

C. Loop Cache:

A loop cache is a tiny buffer that is used for storing instructions of a small loop body .When a small loop that can fit into the loop cache is detected by monitoring short backward branches1 (called trigger branches), instructions are loaded into the loop cache and then the instructions for the next iterations are fetched from the loop cache until the processor exits the loop. Trigger branch detection [12] is performed by a simple additional hardware unit; neither the processor nor the target applications need to be modified at all. In addition, a loop cache itself typically does not have a tag array and valid bits because it is used only for loops whose body does not exceed its capacity (i.e., no conflict miss). Instead, a counter is added to determine whether the loop is completely loaded into the loop cache or not (once a backward branch is detected, the counter is used to count up the fetched instructions to see if all the instructions in the loop are cached). It is known that a loop cache has the ability to reduce dynamic energy of an L1 instruction cache.

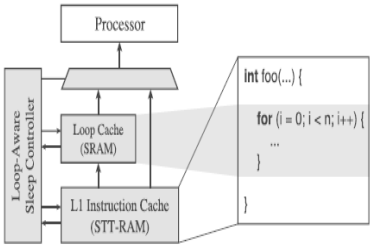

Fig 6. Proposed STT-RAM instruction cache architecture

1) *Static energy reduction via Loop cache:*

In order to reduce leakage energy[5] consumed in I-cache we proposed to combine the loop cache with CMOS circuits having sleep mode such as ABB-MTCMOS and drowsy caches.

TABLE IV. Simulator configuration

Processor	1GHZ,in-order,two-issue superscalar
Loop Cache(SRAM)	2KB,direct mapped,64b blocks
L1 I-cache (SRAM or STT-RAM)	16KB,4way,64 B blocks,1/1-cycle(SRAM read/write) or 1/11-cycle(STT-RAM read/write) latency
L1 D-Cache(SRAM)	16KB,4-way,64B blocks,1-cycle latency
L2 Cache	512KB,8-way,64Bblocks,9-cycle latency
Main memory	200-cycle latency

Even when instructions are supplied from loop cache, static power is consumed by I-cache. Therefore we propose to move I-cache into sleep mode, where threshold voltage of all transistors is increased and thus leakage current is reduced, during the loop cache is active. Because I-cache is not accessed during the active mode, almost no latency penalty is expected .only when the loop cache controller returns to the idle mode from active mode, a penalty is required to activate the sleeping I-cache. Because a tight loop stored in the loop cache [10] has a lot of iteration, this penalty is expected to be negligible.

2) Loop aware sleep controller

We propose a loop-aware sleep controller to coordinate the loop cache and the power-gated L1 instruction cache. Fig.7 shows a state diagram of the proposed controller.2 our policy is composed of three states related to the
loop cache, which are standby (S), fill (F), and active (A1 and A2). The active state is divided into two substates according to whether the L1 instruction cache is in sleep mode (A2) or not (A1).

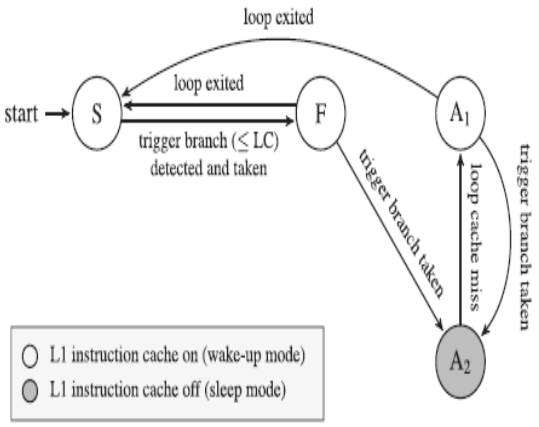

Fig 7. State diagram of the loop aware sleep controller

In the standby state, the loop cache is not used at all and instructions are fetched from the L1 instruction cache. During this state, the controller monitors trigger branches and changes its state to fill when it detects a loop smaller than the loop cache with the mechanism explained in Section III-A. In the fill state, the L1 instruction cache still serves instructions to the processor, but every fetched instruction is also written to the loop cache. The controller still monitors[12] trigger branches in this state and changes its state to active when the trigger branch is taken again, showing that the loop cache is ready to be used. In the active state, the loop cache serves instructions to the processor instead of the L1 instruction cache, and thus, the L1 instruction cache can be turned off completely (A2). On loop cache misses, the controller powers up the L1 instruction cache to load the corresponding block from it (A1). In that case, the controller powers up the L1 instruction cache to load the corresponding block from it. After the iteration, it is turned off again to reduce static energy consumption.

TABLE V. Characteristics of the instruction caches

	Read Energy	Write Energy	Static Power
Loop Cache	7.49pJ	7.49pJ	2.43 mW
SRAM L1 I-cache	20.1 pJ	20.1pJ	21.6 mW
STT-RAM L1 I-cache	33.0 pJ	647pJ	10.4 mW

Among the three loop cache states, the active state consumes the least power. In this state, not only the static power consumption is greatly reduced through powered-off instruction caches, but the dynamic power consumption is also reduced by small access energy of the loop cache compared with the L1 instruction cache. On the contrary, the fill state consumes slightly more dynamic power than the standby state because of the write energy of the loop cache. Its effect on the total power consumption is, however, small in general because it lasts for only one iteration per loop.

VII. EVALUATION OF THE PROPOSED TECHNIQUE

Note that STT-RAM is an emerging technology and thus the modeling results could be different from those of other researches. Although our STT-RAM caches are a bit ahead of the current technology, we believe that the technology will become available in the near future. STT-RAM was already used as L1 caches with read latency less than 1ns in several previous researches.

Power gating of the STT-RAM caches is modeled by referring to the results from the previous research that applied power gating to peripheral circuits of SRAM caches. This is because the only source of static power in STT-RAM technology is peripheral circuits, [12] which have a similar structure to those of SRAM. Among the three sleep modes defined in their research, the deepest sleep mode (the biggest leakage power reduction and the longest wake-up latency) is selected for our experiments. This is because our technique does not incur much performance degradation even with long wake-up latency.

TABLE VI. Summary of the experimental results

	Static Energy	Dynamic Energy	Total Enegy
BASE-STT	-52%	+73%	+0.5%
LOOP-SRAM	-11%	-48%	-14%
LOOP-STT	-41%	-29%	-34%

LASIC	-62%	-30%	-49%

VIII. CONCLUSION

This paper has described various methods to reduce cache leakage power by exploiting generational characteristics of cache-line usage. Compared with the SRAM baseline, the STT-RAM baseline shows negligible performance degradation. Considering that, using STTRAM for the L1 data cache incurs severe performance overhead. This implies that instruction caches are a promising target for STT-RAM. As mentioned before, the reason why long write latency does not take effect is that write operations are performed very rarely in the instruction cache. Our architecture also shows almost identical performance to that of the STT-RAM baseline. Theoretically, it is possible that our architecture might incur performance overhead compared with the LOOP-STT because, each time the L1 instruction cache is accessed during its sleep state (e.g., after loop cache deactivation or on misses), the processor needs to wait for it to be powered up. It is, however, limited to only 0.24% on average according to the experimental results. This is because our policy has a tendency to turn off the instruction cache only when long sleep duration is expected. Once the instruction cache is turned off, its sleep lasts for approximately 200 cycles on average, which is far longer than the break-even time. We also measure the performance overhead under various wakeup latencies from 4 to 30 cycles. According to our results, the performance overhead is limited to only 3% on average even with the longest wake-up latency.

REFERENCES

[1]. X. Dong, X. Wu, G. Sun, Y. Xie, H. H. Li, and Y. Chen, "Circuit and microarchitecture evaluation of 3D stacking magnetic RAM (MRAM) as a universal memory replacement," inProc. 45th Design Autom. Conf., Jun. 2008, pp. 554–559.

[2]. X. Guo, E. Ipek, and T. Soyata, "Resistive computation: Avoiding the power wall with low-leakage, STT-MRAM based computing," inProc. 37th Int. Symp. Comput. Archit., Jun. 2010, pp. 371–382.

[3]. W. Xu, H. Sun, X. Wang, Y. Chen, and T. Zhang, "Design of lastlevel on-chip Cache using spin-torque transfer RAM (STT RAM),"IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 19, no. 3, pp. 483–493, Mar. 2011.

[4]. K. Flautner, N. S. Kim, S. Martin, D. Blaauw, and T. Mudge, "Drowsy caches:Simple techniques for reducing leakage power," in Proc. 29th Ann. Int. Symp.Comput. Archit., May 2002, pp. 148–157.

[5]. S. Kaxiras, Z. Hu, and M. Martonosi, "Cache decay: Exploiting generational behavior to reduce cache leakage power," in Proc. 28th Ann. Int.Symp. Comput. Archit., May 2001, pp. 240–251.

[6]. A. Jog, A. K. Mishra, C. Xu, Y. Xie, V. Narayanan, R. Iyer, and C. R. Das,"Cache revive: Architecting volatile STT-RAM caches for enhanced performance in CMPs," inProc. 49th Design Autom. Conf. Jun. 2012, pp. 243–252.

[7]. X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, "NVSim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," IEEE Trans. Comput. Aided Design Integr. Circuits Syst., vol. 31, no. 7,pp. 994–1007, Jul. 2012.

[8]. H. Sun, C. Liu, W. Xu, J. Zhao, N. Zheng, and T. Zhang, "Using magnetic RAM to build low-power and soft error-resilient L1 cache," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 20, no. 1, pp. 19–28, Jan. 2012.

[9]. C. W. Smullen, V. Mohan, A. Nigam, S. Gurumurthi, and M. R. Stan, "Relaxing non-volatility for fast and energy-efficient STT-RAM caches," in Proc. Int. Symp. High Perform. Comput. Archit., 2011, pp. 50–61.

[10]. H. Homayoun, A. Sasan, A. V. Veidenbaum, H.-C. Yao, S. Golshan, and P. Heydari, "MZZ-HVS: Multiple sleep modes zig-zag horizontal and vertical sleep transistor sharing to reduce leakage power in on-chip SRAM peripheral circuits,"IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 19, no. 12, pp. 2303–2316, Dec. 2011.

[11]. H. Sato and T. Sato, "A static and dynamic energy reduction technique For I-cache and BTB in embedded processors," in Proc. Asia South Pacific Des. Autom. Conf., Jan. 2004, pp. 831–834.

[12]. Junwhan Ahn and Kiyoung Choi,"LASIC:Loop-Aware Sleepy Instruction Caches Based on STT-RAM Technology" IEEE Trans.Very Large Scale Integr.(VLSI) syst,vol 22,No.5., may 2014.