Adaptive Cache Power Management Strategies

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Vinoth Selvan

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

Professor John Sartori

June, 2016

Acknowledgements

Foremost, I would like to thank my family and friends who has been a tremendous source of love, encouragement, and inspiration. The support from my parents Selvan Mani and Buvaneswari Selvan and my wonderful sister Ramya Selvan kept me focused and motivated me to achieve great heights. I would like to thank them for financially backing my graduate studies. A special acknowledgment to my dearest friend, my role model Saranya Ganapathy for mentoring me personally and academically.

I am thankful for having Prof. John Sartori as my research adviser. He has certainly been a major source of motivation and support for the past two years. I am inspired by the quality of design ideas and technical insights he had brought to this thesis. I also thank my final defense committee members Prof. Antonia Zhai and Prof. Ulya Karpuzcu for their guidance and feedback. I also take this opportunity to thank Dr. Aamer Jaleel for his valuable contributions during research group meetings and thank Hari Cherupalli, PhD student and Sathishkumar Vijayakumar, graduate student for their constant help in improving the technical aspects of the project.

I am grateful to my undergraduate professors specially Dr. Subha Rani and Dr. Keshavamurthy, PSG College of Technology for being a source of inspiration and motivating me to take up challenging career roles. I am what I am because of my mentors who extended continuous support in all my ventures. I am thankful to Dr. APJ Alphonse, Dr. Venkatesh Madyastha, Sunjiv Sachan, Shivaprashant Bulusu, Barun Paul, Subhra Bandyopadhyay, Mark Kelley, John Thomson and Scot Hildebrandt. I have tried my best to acknowledge people who had contributed to the success I have been fortunate to experience but I apologize for any omissions and convey my respect and admiration.

Abstract

Power management in SRAM based caches was increasingly popular for the fact that leakage power consumption of cache hierarchies is comparable to the power consumed by logic cores. There are a number of power management techniques proposed using circuit design and architectural techniques to reduce leakage power in SRAM based memories. Existing techniques closely monitor the behavior of sub-array building blocks in caches and dictates when to enter and exit sleep mode. This global sleep control policy is conservative and inefficient as it requires a closed loop architectural support to make low power decisions. In this work, we introduce aggressive power management schemes which has local decision logic that maintains parts of the sub-array in sleep mode even during active phases of operation thus saving leakage and idle power. We propose and implement three micro-architectural techniques named DRAM-like refresh, column based sleep and bitline segmentation in the 128kb SRAM sub-array building block to adopt additional power management schemes that are enabled per idle cycle basis. In active mode, each of these methods achieve leakage and dynamic power reduction by activating only a portion of the sub-array while during sleep mode, DRAM-like refresh trades off sub-threshold and dynamic refresh power to save idle data retention power. Our evaluations show that these methods on an average can achieve 20% more leakage savings during sleep and up to 75% more average power savings during active and idle operation. The implementation comes with an area overhead of 4-5% without any impact on the memory occupancy ratio. Later, we perform architectural evaluations to exploit memory access pattern and reconfigure the power management control circuitry to dynamically operate based on the demand of the application during active and idle states thus achieving additional power savings compared to static schemes. We also recommend combinations of these adaptive power management schemes for different levels of memory hierarchy after profiling the memory access pattern of various workloads.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Today's focus is shifting from a pure performance based computing to power-efficient high performance computing. In these HPC systems, one of the key determinants of overall system performance has been the memory hierarchy. In the coming days, there will be a huge pressure on the memory systems to accommodate more data for increasing performance and energy efficiency. The efficiency of the memory hierarchy is a strong function of the power consumption of on-chip caches. Future last-level caches are expected to occupy half the processor chips die area and consume more than half of the total power budget [1]. Power dissipation is becoming an increasingly important constraint on high performance processor design. With the advent of deep sub-micron circuit technology, the ratio of static-to-dynamic power in on-chip memories proportionally increases and therefore managing leakage power in SRAM to meet the power budget is critical. Moores Law has driven CMOS technology well into the nanoscale regime and controlling SRAM leakage power becomes more and more challenging. At the circuit level, designers propose low leakage SRAM operating modes (i.e., sleep mode or standby mode) and at architecture level, there are increasing interests in how to efficiently integrate such features into the design.

SRAM-based memory structures in modern microprocessors are equipped with advanced leakage management designs that are enabled in stand-by state. Leakage control techniques are implemented in all the levels of cache hierarchy and are carefully designed considering the wakeup time and wakeup energy. The importance of memory energy

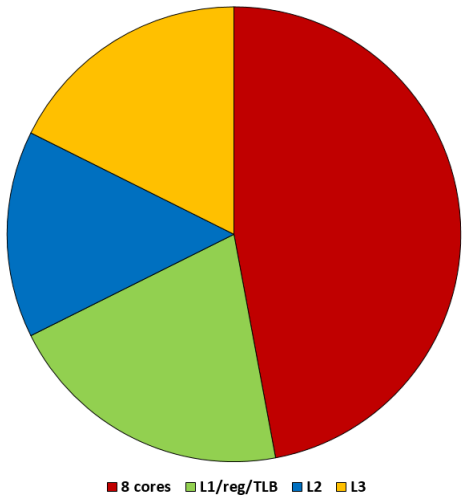


Figure 1.1: Power Breakdown of an 8 Core Server Chip cache[1]

is shown in Figure 1.1 which gives the power breakdown of a 40nm, 8-core superscalar processor with an 8MB last-level cache[1]. Over 50% of the processor die energy is dissipated in the caches and register files in this machine. While the cache hierarchy consumes a significant amount of dynamic power, majority of the total power consumption is consumed during idle and sleep state. Given that the average power consumption is now critical, it is worth revisiting many of the cache power management design strategies with a view to minimizing the average power.

On-die cache in modern microprocessors is organized in multiple levels of cache hierarchy but the basic building block is a 128kb sub-array which is tiled to form bigger cache units. So any optimization that can happen at sub-array level can potentially benefit the cache hierarchy on whole. Most of the power and energy spent is not to read or write contents in memory but to retain data when not being accessed. Architectural simulations performed at various levels of cache hierarchies indicate that most of the time cache is in data retention state and this idle state is the major source of leakage power. There are a number of prior works over the past two decades that proposed efficient solutions to address these problems at circuit level. An overview on some of these power management techniques are mentioned in chapter 3 section 3.3.1. In addition, architects used these circuit techniques to propose power management schemes at higher

levels of abstraction. Some of the widely adopted schemes in this segment are discussed in chapter 3 section 3.3.3. All of these techniques uses a global control mechanism in determining the sleep condition. This is inefficient as it becomes complicated to manage the assertion and desertion of sleep as the number of sub-arrays increases especially in last-level caches.

Current sleep management policies are such that, a global power management unit observes the scope of low power modes in individual sub-arrays and initiates a transition from active to sleep mode and vice versa. These techniques are not intelligent enough to adapt its mechanism since the control is coarse-granular. Global control is particularly inefficient because there is an overhead involved in identifying which units can be put in low power mode and complexity in scheduling the timing of sleep and active transitions. Understanding the urge to save more power, we propose techniques that enhance the sub-array to locally make a decision to enter and exit low power mode. Having decision logic local to the sub-array is easier to maintain and more accurate. In addition, there will be no interference from neighboring sub-array units when deciding to operate in different power modes. In chapter 4, we presents in detail the implementation of our proposed adaptive power management techniques.

All the levels of cache hierarchy uses the same 128kb sub-array [3] as the basic building block. The sub-arrays in every of these levels exhibit a different behavior based on the demand of the application and the size of the working set. When architectural explorations were performed, there were different regions of operations were there was a huge scope for additional improvements where the proposed schemes would perform much better with minor modifications. Like the earlier architectural methods, we also see an opportunity to bridge the gap between memory access pattern and micro-architectural techniques and propose adaptive dynamic power management schemes that behaves differently for different workloads to achieve better performance in comparison to static adaptive power management schemes proposed earlier. In chapter 5 we discuss the architectural exploration and how the proposed techniques are enhanced based on our observations.

This work makes the following contributions:

- We propose three power management schemes - a. DRAM-like refresh b. fine-granular column sleep and c. bitline segmentation that are locally triggered in different power modes of sub-array
- We implement and evaluate our proposed adaptive power management circuitry with an 128kb sub-array and show that we save leakage and dynamic power during sleep and active mode respectively. DRAM-like refresh provides 12% and 19% average power savings; fine-granular column sleep achieves upto 75% average power reduction and 15% average access power savings with bitline segmentation
- Perform architectural simulations to study the memory access pattern for a variety of applications at different levels of cache hierarchy and show that our proposed adaptive power management schemes can achieve additional benefits with architectural support

Rest of the thesis is organized as follows:

- chapter 2 presents an overview of SRAM based cache architecture and explains the sub-array organization.
- chapter 3 describes the importance of power management and outlines existing power management techniques.
- chapter 4 hashes out the strategy and implementation of adaptive power management techniques and presents the simulated results and analyses.
- chapter 5 demonstrates the importance of architectural exploration to improve local sleep controllers followed by scope of future work.
- chapter 6 presents a final discussion of the analyses presented in the thesis.

Chapter 2

Cache Organization

2.1 Logical Organization

A simple cache consists of a set of Static Random Access Memory (SRAM) cells organized as multiple two-dimensional arrays to store data and tags. Figure 2.1 shows the organization of a cache with a single data and tag array[2]. Each memory cell is equipped with access transistors that facilitate read or write operations, and the access transistors are controlled through horizontal wiring called wordlines. A centralized decoder takes the address request and identifies the appropriate wordline in the data and the tag array. To reduce the area overhead of the decoder and wiring complexity, a single wordline is shared by an entire row of cells. For every access, the decoder decodes the input address and activates the appropriate wordline matching the address. The contents of an entire row of cells are placed on bitlines, which are written to the cells for a write operation or delivered to the output for a read operation. Although the data placed on the bitlines can be used to directly drive logic for small arrays, because of large bitline capacitance, the access latency of such a design can be very high even for moderately sized arrays. To reduce access time, sense amplifiers are employed to detect a small variation in bitline signal and amplify them to the logic swing. The sense amplifiers also limit the voltage swing in bitlines and hence their energy consumption.

The energy and delay to read or write an SRAM cell is inherently very small; it takes only a fraction of a CPU cycle to read an SRAM cell. Also, the dynamic energy

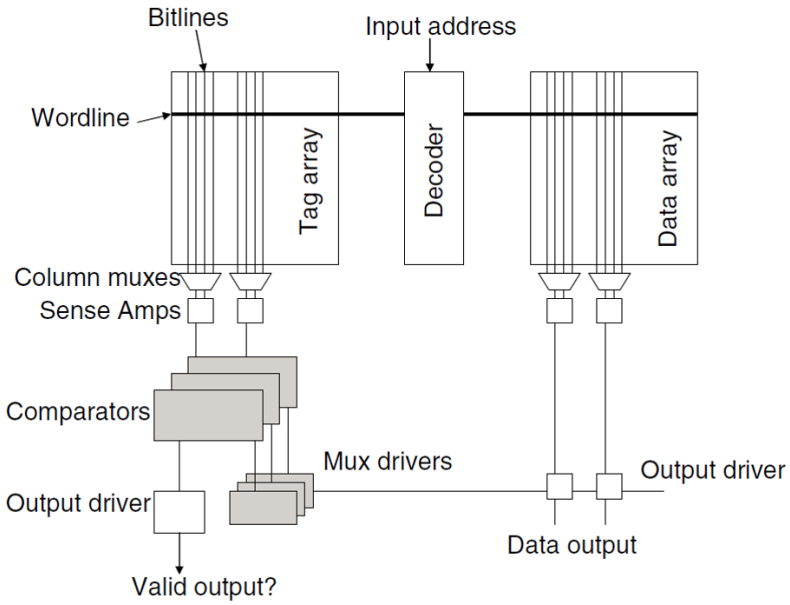


Figure 2.1: Logical Organization of a Cache [2]

consumption in SRAM cells is insignificant. The cache access latency is primarily dominated by the wire delay associated with decoder, wordline, bitline, and output bus. Similarly, the number of bitlines activated during a read or write and the length of the output bus determines the dynamic power of a cache.

The data and tag arrays are built from instantiating small units called sub-arrays. These sub-arrays are usually 128kb in size and each of them has it's own row decodes and column circuitry. Figure 2.2 shows the organization of a 2MB 64B block 16-way set associative cache [24]. The cache consist of 128 such sub-arrays and each way has 8 sub-arrays. One sub-array contains 8B data in a wordline making upto 64B block size per way. The following section presents the 128kb SRAM sub-array design.

2.2 Sub-array Design

E Karl et al. proposed a 128kb SRAM sub-array design [3] with integrated read and write assist circuitry. As shown in Figure 2.3, the macro floorplan for the 128 kb SRAM (512 x 256) consists of 6T SRAM cells arranged as four quadrants, separated by the row

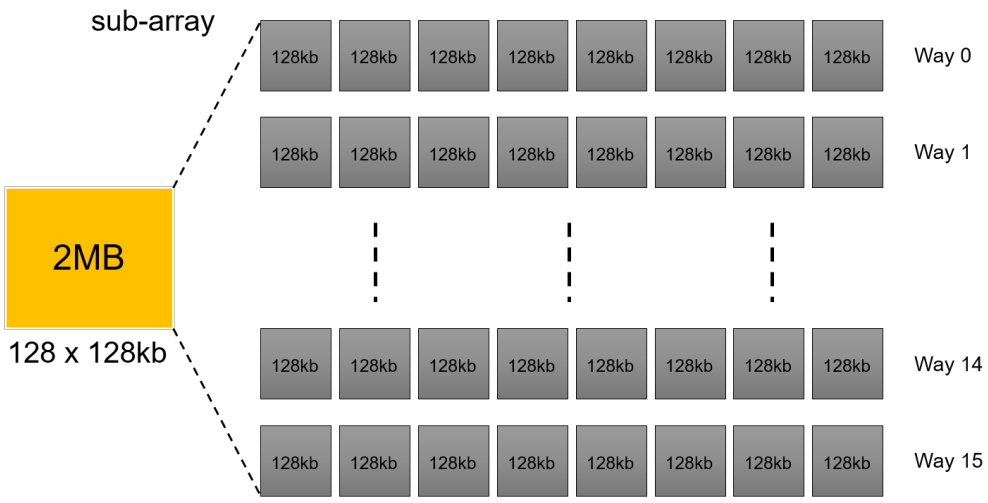


Figure 2.2: Organization of a 2MB 16-way Cache

decoder, which selects the row address vertically and the column decoder selecting the column to be addressed horizontally. The pre-decoder and sleep circuitry are laid-out at the center of the sub-array. The SRAM macro is designed for synchronous high-speed operation in a highly pipelined design environment, utilizing one clock cycle for each wordline assertion and supporting accesses with 2-cycle total latency. Sense amplifier control signals are derived from the clock cycle following wordline assertion, but the array timing circuitry contains delay tunability to trade sense amplifier margin with the interconnect-dominated output path delay to achieve maximum clock frequency.

We implemented the 128kb sub-array design as shown in Figure 2.3 using NCSU FreePDK [30] using 45nm technology node following the above recommendation. Table 2.1 summarizes the parameters used and we use this as our baseline design configuration to further build other power management techniques.

Memory Array: The 128kb (512 wordlines x 256 bitlines) sub-array is divided into four identical symmetric quadrants and the peripheral control circuits are placed in the center of the quadrants. The sub-array can be addressed with 11 bits, 9 MSB bits for wordline selection and interleaved bitlines are column multiplexed with 2 LSB bits.

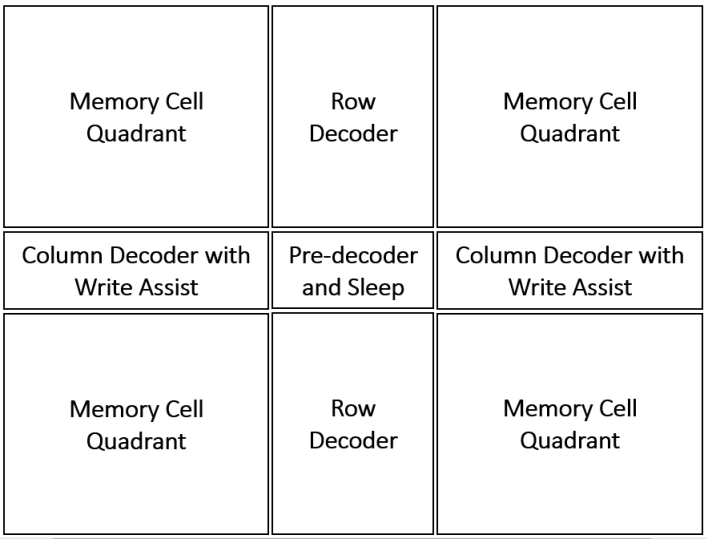


Figure 2.3: 128kb Sub-array Design

Pre-decoder and Row Decoder: The memory cells to be addressed are enabled by asserting the appropriate wordlines using a two-stage decoder. To reduce the overall delay the pre-decoder stage is clocked while the decoder uses address information to drive the horizontal wordlines.

Column Circuit: Column circuit consist of bitline precharge logic and address based column multiplexing to select bit-interleaved 64b words. Before any read or write cycle, this circuit performs non-anticipated precharge assisting the next access.

Write and Read Assist Circuit: This logic consist of write enabled based muxing circuit to choose voltage levels to assist write at a lower voltage to avoid pseudo-read or half-select issue. Based on the address being accessed, the column circuit and the write assist circuit works together to lower the voltage of the column being written while keeping the other three columns in Vdd.

Sleep Controller: Sleep in sub-array is accomplished in a staggered fashion on a quadrant by quadrant basis by raising the ground to a higher potential of 0.3V using a sleep activated current mirror circuit. The sleep latency is $\sim$ 7-10 cycles after the

assertion of global sleep and the wakeup latency is 4 cycles after the desertion of sleep.

Table 2.1: Baseline Sub-array Configuration

Parameter	Symbol	Value
Clock Frequency	Clk	1 GHz
Supply Voltage	Vdd	1.1V
Ground Voltage	Vgnd	0.0V
Virtual Ground Reference	Vref	0.3V
Write Voltage	Vwrite	0.8V
Read Voltage	Vread	1.1V

Chapter 3

Power Management in SRAM

Cache is maintained in low power mode when not active. Specifically, the global power management unit observes every individual sub-array and identifies those that can be held in data retention state. The sub-array presented in chapter 2 section 2.2 is equipped with sleep controllers that gets activated in low power mode. Before understanding the existing power management techniques, we use the following two sections to understand leakage power breakdown in sub-array and the importance of data retention voltage in SRAM.

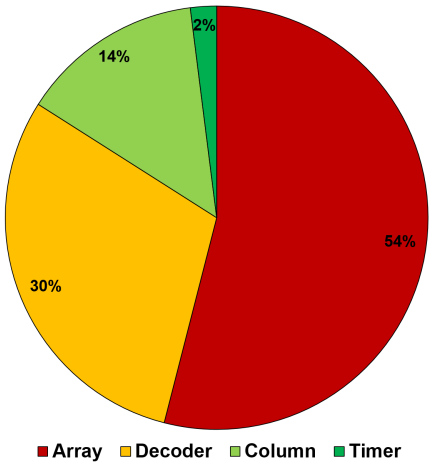


Figure 3.1: 128kb Sub-array Leakage Breakdown [4]

3.1 Leakage in SRAM

Figure 3.1 shows the breakdown of sub-array leakage components based on circuit simulation[5]. The data suggests SRAM memory array and column circuitry together contribute 68% of standby leakage. The majority of the leakage comes when the memory is in stand-by state. Figure 3.2 explains the cell leakage and bitline leakage paths of a typical 6T-SRAM cell [6]. When there is no activity, all the cells in the sub-array are maintained at Data Retention Voltage (VDRV) aka Vccmin. At this lower voltage level, leakage is less and hence overall power consumption is reduced.

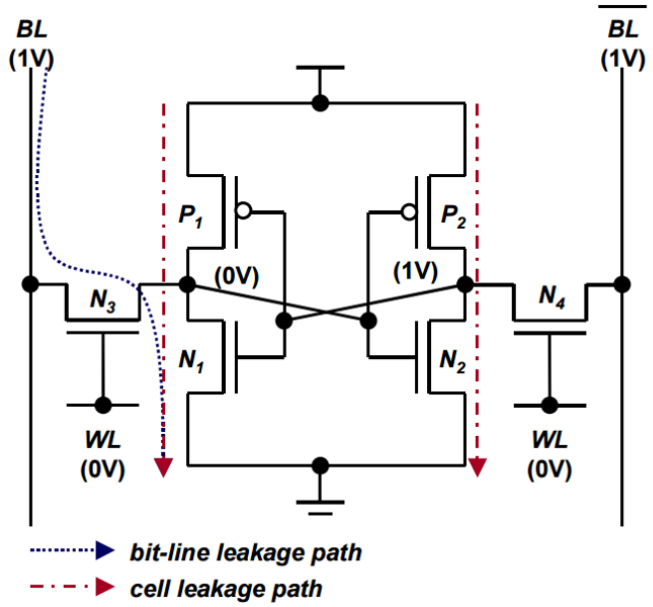


Figure 3.2: Leakage Paths in 6T SRAM Cell [5]

3.2 Data Retention in SRAM

Data Retention Voltage is the voltage below which a bitcell loses its data. As the minimum Vdd required for data preservation, DRV of an SRAM cell is a measure of its state-retention capability under very low voltage. In order to reliably preserve data in an SRAM cell, the cross-coupled inverters must have a loop gain greater than one. The stability of an SRAM cell is also indicated by the static-noise margin (SNM). When Vdd

scales down to DRV, the Voltage Transfer Characteristics of the cross-coupled inverters degrade to such a level that the loop gain reduces to one and SNM of the SRAM cell falls to zero in stand-by mode. The basic idea of lowering the Vdd is to reduce leakage. At DRV, sub-threshold leakage, Gate Induced Drain Leakage and gate tunneling are low. In the interest of saving these leakage powers, during stand-by mode, all the sub-arrays that are not being used are held at data retention mode. DRV for NCSU 45nm FreePDK was characterized and found to be 420mV. The following section outlines existing power management techniques in circuit design and architectural level to reduce leakage and dynamic power of sub-array.

3.3 Power Management Techniques

Many approaches to reduce leakage power have been investigated, at the technology, circuit and architecture levels.

3.3.1 Circuit-level Leakage Control

Four main circuit schemes have been proposed to reduce leakage power:

- ***Gated-Vdd*** turns off the power by using a high threshold transistor [9]. The advantage of this technique is in reducing the leakage power virtually completely. However, it does not retain the state of the memory cell. Similarly, Vss can also be Gated (Gated Vss).

- ***ABB-MTCMOS*** increases the threshold voltage of a transistor dynamically [10]. The overhead of applying this technique in terms of performance and power makes it inefficient. A variation of ABB-MTCMOS technique is proposed in [11], [12] in which the leakage power is suppressed in the unselected part of cache by utilizing super Vt devices (forward body biasing scheme).

- ***Voltage scaling*** reduces the source voltage. As explained in [13], due to short-channel effects in deep submicron processes, voltage scaling reduces the leakage current significantly. Voltage scaling can be done dynamically and combined with Frequency Scaling (DVFS) [14], [15].

- ***Cell bias voltage reduction*** in the standby state to reduce cell leakage [16] is a technique shown to reduce cell leakage more compared to source voltage scaling [17].

3.3.2 Intel's Power Management

Y. Wang et al. proposed several integrated sleep features [4] to enable leakage reduction in the 128kb SRAM macro when the arrays are not undergoing active operation. The sleep circuitry in the SRAM macro is designed to tradeoff the dynamic power associated with charging the region and sleep device and decoding area overhead associated with fine-grained power gating. The SRAM bitcell array power grid is segmented into four 32kb regions as shown in Figure 3.3 to enable fine-grained power gating. One cycle prior to a read or write operation, the cell supply voltage for one of the four 32kb SRAM regions is charged to the current active VCC level from a retentive sleep state. SRAM bitcell array regions in the retentive sleep condition are passively biased with weak, programmable devices to prevent retention failures at low voltages. A wide range of silicon skew and temperature conditions can be accommodated by the seven settings for the passive sleep bias circuitry included in the SRAM macro. Timer and I/O column peripheral circuits make extensive use of low-leakage and stacked devices to minimize overall leakage on devices on non-critical speed paths. The limitation of this implementation is the overhead in the implementation of voltage drivers for every cache line. For a 2048 cache line sub-array with 4 bits interleaved, this proposal requires 512 drivers. Our proposal addresses to implement a much efficient power management scheme with minimal modification to the peripheral circuitry.

3.3.3 Architectural Power Management Techniques

A number of architecturally driven cache leakage reduction techniques have been proposed:

- ***Drowsy Cache*** reduces the supply voltage of the L1 cache line (instead of gating it off completely) [18]. The advantage of this technique is that it preserves the cache line information but introduces a delay in accessing drowsy lines. However, the leakage power saving is slightly lower than the Gated-Vdd and Vss techniques.

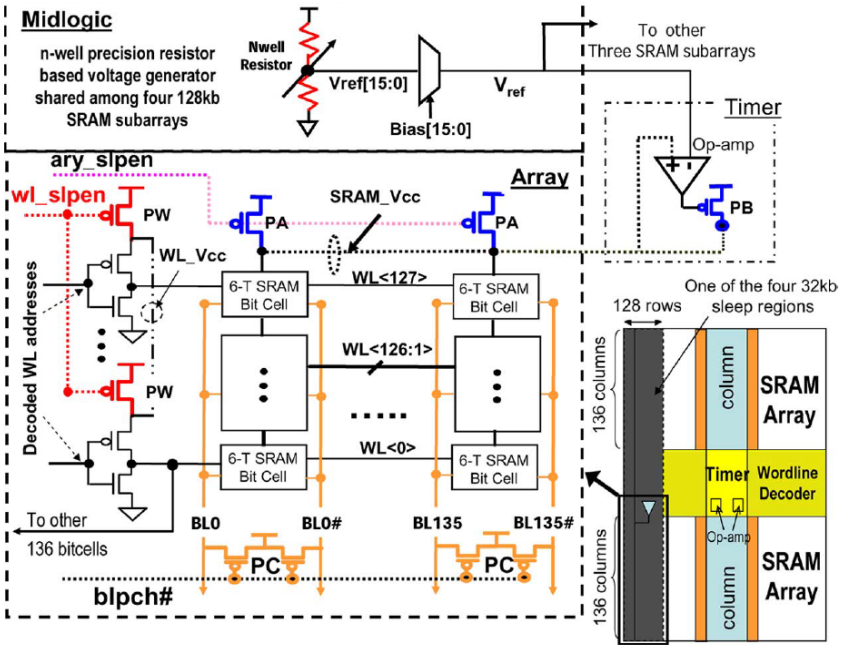


Figure 3.3: Schematic of Intel's Power Management Circuit [4]

- ***Cache Decay*** technique reduces cache leakage by turning off cache lines not likely to be reused [19]. In this method, a cache line is kept on as long as it is re-accessed within a time window called decay interval.

- ***Gated-Vdd*** approach can be applied to gate the power supply for cache lines that are not likely to be accessed [20]. This technique results in the data loss in the gated cache line. This leads to an increase in the cache miss rate and loss of performance.

- Fast Speculative Address Genera-tion and Way Caching using address generation to apply the drowsy cache line technique to reduce both the L1 cache dynamic and leakage power [21].

- Leakage Optimazation techniques for several components of on-chip caches to reduce gate leakage power [22].

3.3.4 Adaptive Pheripheral Power Management Techniques

The technique proposed by H. Homayoun et al. [23] aims at putting the cache peripheral circuits into a stand-by, low power mode. This is accomplished by inserting appropriately-sized sleep transistors for both Vdd and Vss in the peripheral circuits. The SRAM cell leakage is assumed to be controlled by other techniques such as Gated-Vdd, in both the baseline and their enhanced architectures. The design technique discussed consists in a set of modifications to the standard L2 cache architecture including a global input sleep signal SLP to place its peripheral circuits into stand-by mode. The other modification deals with cache access in the lowpower mode- the processor is not always disabled during lowpower mode and can generate L2 accesses. The misses from L1 cache within the processor are stored in a delayed-access buffer, which is smaller in size and consumes less power. The buffer makes it appear as if the processor has issued a request to the disabled cache. The accesses stored in this buffer are sent to the L2 cache after it returns from the low-power mode.

These techniques achieve significant power savings when the power management policy identifies when sub-arrays can be put into sleep. Cache units receives the global sleep input from a centralized controller and accuracy of the sleep assertion depends on the complexity of the controller. Specifically, for last level caches, there will be lot many sub-array units and issuing customized sleep by the global controller is practically not feasible. Also, these techniques will not be activated during the active or idle states of the sub-array when there are limited accesses. We address these issues by minimize the architectural dependencies by having controllers locally and make the sub-array responsive to dynamic sleep transitions.

Chapter 4

Adaptive Power Management Strategies

Power management schemes are implemented at all levels of cache hierarchy to reduce leakage when they are in stand-by state. As detailed in chapter 3, conservative power management strategies has global sleep controllers that dictates which sub-arrays can enter and exit low power modes. This mechanism is inefficient as it requires significant architectural support in determining which units can potentially be put in data retention state. Considering the overhead involved in deciding the timing of sleep assertion and desertion, this global sleep controller observes idleness on these sub-arrays for a considerable duration and hence not every idle cycle is capitalized. In addition, the current sleep policy is at a sub-array granularity meaning either the sub-array is completely active or maintained in data retention mode. We address the above issues and propose three different adaptive power management techniques that perform fine-granular power management in sub-array level. These techniques implements the decision logic inherent to the sub-array rather than relying on global controller. Local control is efficient as it can be immediate and more accurate. There will also be less interference from neighboring sub-arrays and hence every individual sub-array can mutually independently transition to and out of low power modes based on the decision made by their controllers. The following sections will discuss in detail on adaptive power management schemes with results and analysis.

4.1 DRAM-like Refresh in SRAM - Sleep and Active Mode

Conventionally, the memory elements that are not being accessed are put in Data Retention Voltage. The main objective of retaining memory elements in DRV is to reduce leakage power during idle condition. Existing power management policies observe idle cycles in cache and determine an appropriate state at which the sub-arrays can be put into sleep. Considering the overhead involved in entering into or exiting out of data retention mode, the policy waits for enough time before it determines the sleep criteria. DRV is the limiting factor and maintaining memory at voltage levels lower than DRV will result in readability, writability and retention. If DRV is not satisfied, we will lose all the stored data. But our experiments show that we can operate SRAM based cache below DRV still maintaining data for a momentary duration. Fundamentally, by changing the way retention is performed in an SRAM, traditional data retention way sleep can be replaced with our novel DRAM-like refresh mechanism.

4.1.1 DRAM-like Refresh Mechanism

The micro-architecture of the sub-array read and write assist circuit is enhanced with additional intelligence to have a better control on the following parameters:

Cell Supply Voltage, Vcell: The memory cells in each columns of the sub-array can be maintained at cell supply voltage Vcell, which can be dynamically controlled based on the type of access. During any read or write, Vcell will assume Vread and Vwrite respectively and when not being accessed Vcell is maintained in sub-threshold region (0.0V to 0.3V). Vcell momentarily assumes Vrefresh when refresh operation is performed.

Read Voltage, Vread: The read assist circuit is modified to use Vread as the voltage with which any read access is performed. Vread is currently maintained at 1.1V (baseline read voltage) to compare the benefits of our proposal with existing power management techniques. During any read, Vcell = Vread.

Write Voltage, Vwrite: The write assist circuit is modified to use Vwrite as the voltage with which any write access is performed. Vwrite is currently maintained at 1.1V (baseline write voltage) to compare the benefits of our proposal with existing power management techniques. During any write, Vcell = Vwrite. Write assist in [8] proposes techniques to use Vwrite as 0V, but we will not be using this scheme as we cannot capitalize a write as a refresh.

Refresh Voltage, Vrefresh: The read and write assist circuit is modified to have an additional control voltage called the refresh voltage Vrefresh. Vrefresh is used to refresh the contents of the SRAM cell before the contents get corrupted. During a refresh, Vcell = Vrefresh.

During sleep mode, instead of maintaining supply voltage of 6T SRAM cells at VDRV, they can be held at sub-threshold voltage levels. When $V_{DRV} < V_{th}$, the data stored in the cell is leaky and starts decaying following the sub-threshold equation. In order to preserve original data, all such capacitor nodes in the sub-array needs to be refreshed. This refresh mechanism is something similar to what is being performed in DRAM.

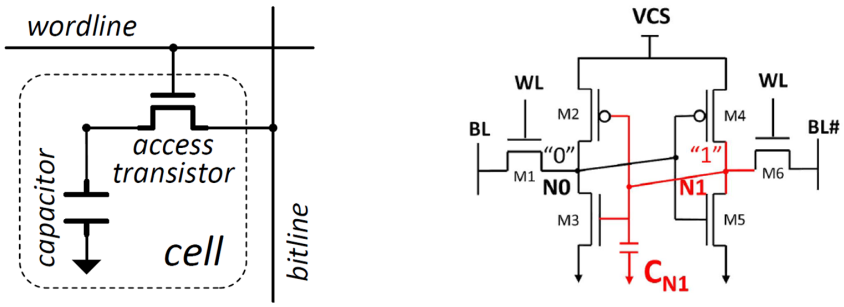
Summary of the DRAM-like mechanism will be helpful in understanding the rest of the sections.

- Maintain cell supply voltage at Vcell = $V_{DRV} < V_{th}$
- The data stored in all the nodes starts leaking
- Life of the data depends on the value of Vcell
- The capacitor nodes needs to be refreshed beyond repair
- Periodicity of the refresh depends on the duration of idleness and frequency of read and write accesses

Sections 4.1.3 and 4.1.4 explain in detail about the refresh mechanism in sleep and active mode.

4.1.2 DRAM-like Refresh in SRAM vs. DRAM refresh

Before understanding the DRAM-like refresh mechanism in SRAM, it is worth comparing and contrasting the refresh mechanism in DRAM [27]. Figure 4.1 shows 1 bit storage nodes of a DRAM and SRAM memory.



(a) DRAM Storage Node [26] (b) SRAM Storage Node [3]

Figure 4.1: DRAM and SRAM Storage Capacitor Nodes

DRAM Refresh Mechanism:

a. The capacitor decays and lose data eventually
b. Refresh is performed by charging wordlines to restores capacitor
c. Refresh is done row by row after retention time
d. Address information is required to perform a refresh

SRAM Refresh Mechanism:

a. The capacitor (CN1) decays and lose data eventually
b. Refresh is performed by boosting Vcell above VDRV
c. Refresh is done column by column after Nlife
d. Refreshes are periodic and address independent

4.1.3 Pulsed Periodic DRAM-like Refresh - Sleep Mode

The operation of the SRAM is changed fundamentally by making the Vcell of the SRAM to be in sub-threshold level after any read or write access. As a result of this technique, after the execution of any read/write cycle, the data stored in the memory

starts decaying following the sub-threshold leakage equation. The amount of time for which the stored charge lasts before it completely corrupted is Nlife. Any read access after this Nlife will result in a read failure and the periodic pulsed refresh should happen before this time. The periodicity of the pulse depends on the Vcell level and other factors like technology parameters and temperature.

Figure 4.2 explains the modifications made to improve the existing assist circuit. In addition to the existing circuit, new parameters like Vcell, CE (Cell Enable), Vrefresh and SE (Shot Enable) are added. Following section will explain in detail the mechanism of DRAM-like refresh.

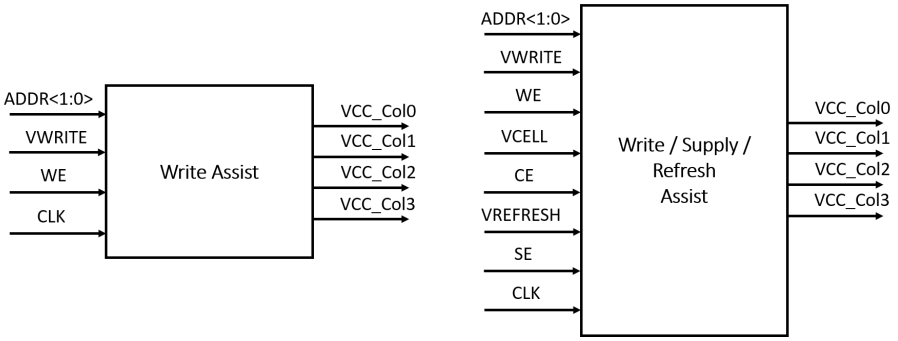


Figure 4.2: Baseline Assist Circuit (Left) and Adaptive Refresh Assist Circuit (Right)

Simulations were performed at varied ranges of Vcell from 0.0V to 0.3V and it was observed that the life cycles of the data stored in the cells are Nlife and any read performed after the Nlife period will result in a read failure. This failure can be avoided by continuously refreshing the contents of the cells at least one cycle before Nlife to keep the data alive. We need to keep the refresh activity happening at regular intervals of Nlife to preserve the data. This periodic pulsed refresh is done with Vcell = Vrefresh which restores the contents of SRAM cells. One inherent advantage of this micro-architecture is that, there is no additional control required from outside of sub-array to initiate this sleep mechanism. Hence this DRAM-like refresh is quick and potentially any idle cycle can be utilized to save leakage power. This techniques achieves average power reduction by trading off refresh dynamic power and sub-threshold leakage power to save idle data retention power.

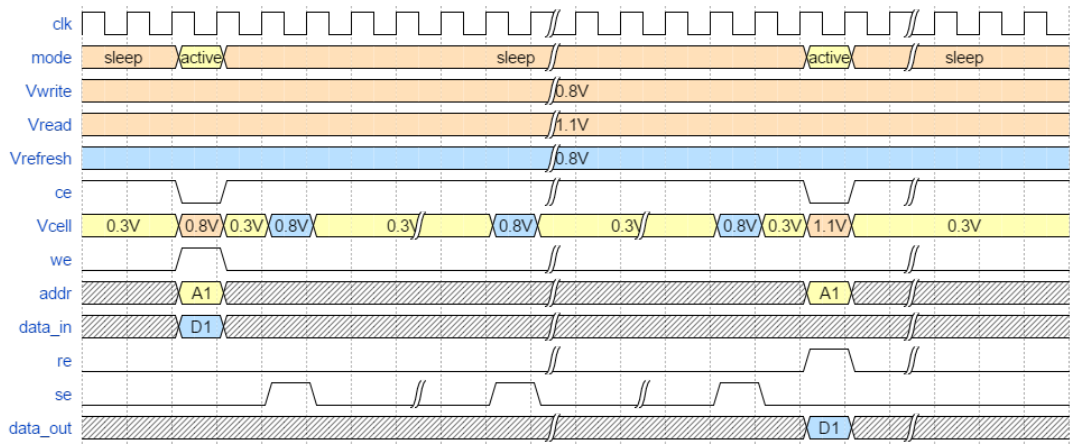


Figure 4.3: DRAM-like Refresh Waveform

Waveform in Figure 4.3 explains the pulsed periodic refresh mechanism for Vcell = 0.3V. Writes to memories are done at Vwrite = 0.8V, reads at Vread = 1.1V and refresh at Vrefresh = 0.8V. Upton write request, write enable (we) is asserted and the sub-array transitions from sleep mode (ce = 1) to active ode (ce = 0). After write operation, all the cells in the sub-array are maintained at Vcell = 0.3V by asserting the ce = 1. The data stored in the nodes stars leaking and as explained earlier, periodic pulsed refresh has to be performed. Local sleep controller generates periodic shot enable signal (se) when ce is asserted. The pulsed periodic refreshes keeps the data alive by boosting the Vcell = Vrefresh every time se is asserted. This periodic se pulse is spaced by Nlife cycles. When a read request arrives (re = 1), the sub-array transitions to active mode and legal data is properly latched by the sense amplifier.

4.1.4 Adaptive DRAM-like Refresh - Active Mode

Applications have varied access pattern and variable idle cycles between accesses. Each of the sub-array in the entire cache behaves different based on the demand of the application and conservative sleep management policy identifies sleep regions and initiates sub-array sleep. Alternatively, our proposed periodic pulsed DRAM-like refresh mechanism can be used that can potentially use idle cycles between accesses to save power. But for applications that has more timely accesses and shorter duration idles cycles, pre-times pulses may not be required. We need a mechanism to adaptively trigger

refresh whenever needed to keep the contents of the memory intact. Any read or write performed during the idle cycles is a self-refresh with Vrefresh = Vread / Vwrite. When there are no accesses that keeps the memory alive, refresh can be done on a column by column basis with Vrefresh. The following explains the how this adaptive application based refresh mechanism works.

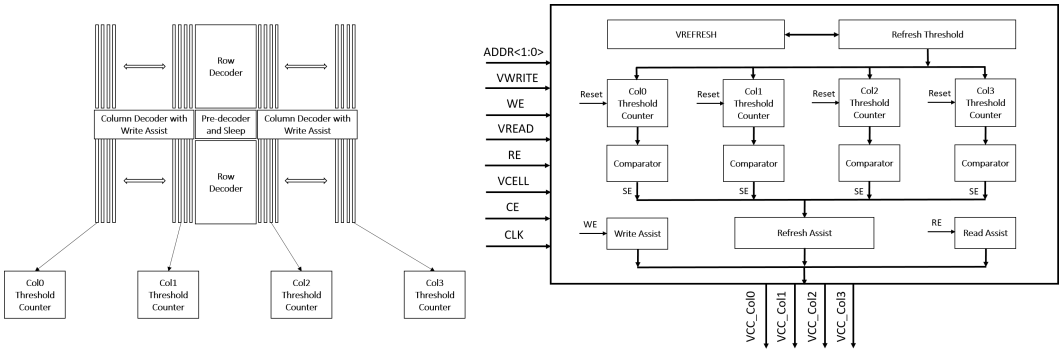
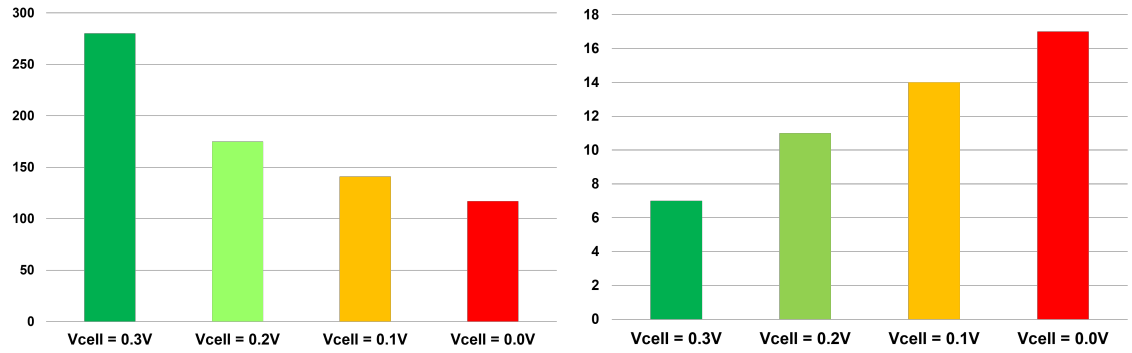


Figure 4.4: Adaptive Refresh Assist Circuit Implementation

The life cycle of the memory elements in the sub-array is monitored by counting the idle cycles in all four columns mutually independent with threshold counters as shown in Figure 4.4. When the idle cycles count reaches the threshold Nthreshold, the column is refreshed with the assist circuit. This is feasible because all the cells in the entire column is supplied with Vcell = Vrefresh implying any access in a column will refresh all the memory elements present in the column. This is certainly advantageous because with four consecutive cycles of refresh, the entire sub-array can be refreshed. The refreshing introduces dynamic power over head for a single clock cycle but there will be huge power saving for Nthreshold cycles and hence the average power consumed over the time of cache access decreases. The dynamic power of the refresh can be reduced by minimizing the number of refreshes which depended on the Vcell voltage. Our evaluation shows that Vcell of 0.3V provided maximum benefits since the Nthreshold is higher and hence required fewer refreshes. The value of Nthreshold depends on technology parameters and the temperature of operation. We also recommend different Vcell and Vrefresh voltage for different application based on the memory access pattern in chapter 5.

As shown in Figure 4.5 (a), life of the data in storage nodes depended on the Vcell



(a) Time to data corruption in ns at 80C (b) Number of Refresh at 80C for 2000ns Sleep

Figure 4.5: Pulsed Periodic Refresh in SRAM

value. For a Vcell = 0.3V, Nlife is about 280ns and for Vcell = 0V, Nlife is 117ns. This implies that for the same sleep duration, when using different Vcell values, the frequency of the periodic pulsed refresh will vary. Figure 4.5 (b) shows the number of refreshes performed over a 2000ns sleep duration and can be seen that at Vcell = 0.3V we have to perform 7 refreshes and for Vcell = 0V we require 17 refreshes to keep the data legal.

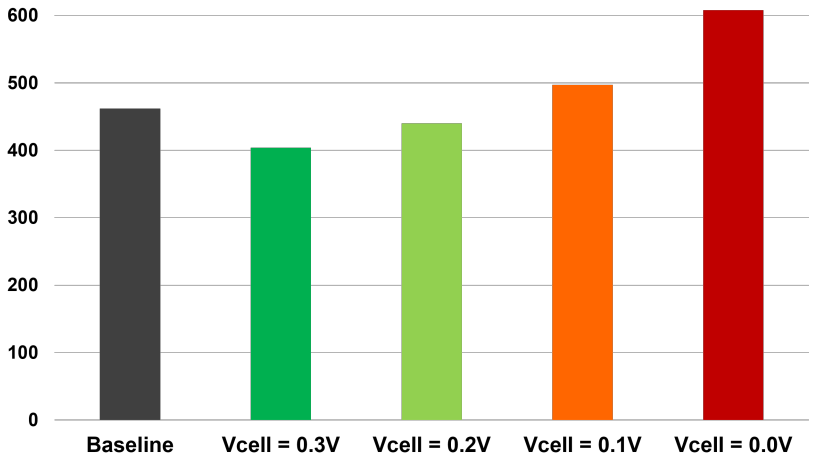


Figure 4.6: Average Power in microwatts at 80C during Sleep Mode

From the previous analysis, it is clear that for lower Vcell we need to perform frequent periodic refreshes. The dynamic power associated with a refresh is much higher and the overall average power consumed over a given sleep period increases if the number of refreshes are higher. Figure 4.6 explains the intuition that at higher Vcell the average power consumed will be lower than at lower Vcell. We could see that the periodic pulsed refresh performs better than baseline saving 15% and 5% respectively at Vcell = 0.3V and Vcell = 0.2V and this method under performs at lower Vcell values as the dynamic refresh overhead is more than the idle data retention to retain cell values.

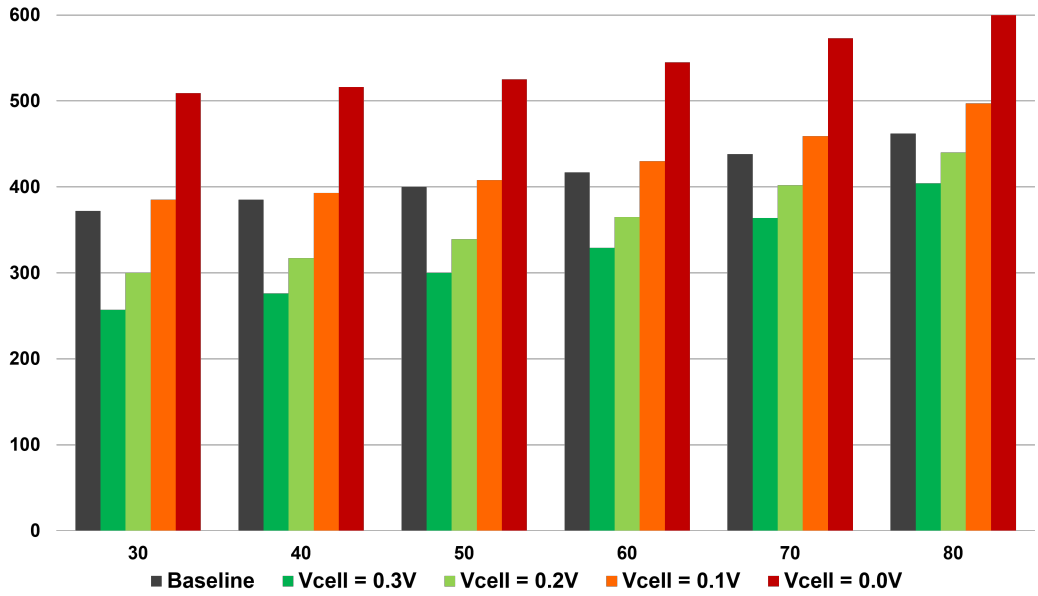


Figure 4.7: Average Power in microwatts during Sleep Mode vs. Temperature

Temperature analysis is critical especially in memory systems. Since our proposed methodology is based on the sub-threshold leakage, it is important to perform temperature dependence test. Figure 4.7 explains that the average power consumption gradually decreases with lower operating temperature for both baseline and DRAM-like refresh system. Modern microprocessor are equipped with on-chip temperature monitors, CMOS sensors, performance counters, etc with which real-time temperature measurement is feasible. Our analyses shows that the value of the Nlife is higher at higher temperatures. If there is no temperature feedback, considering the worst case

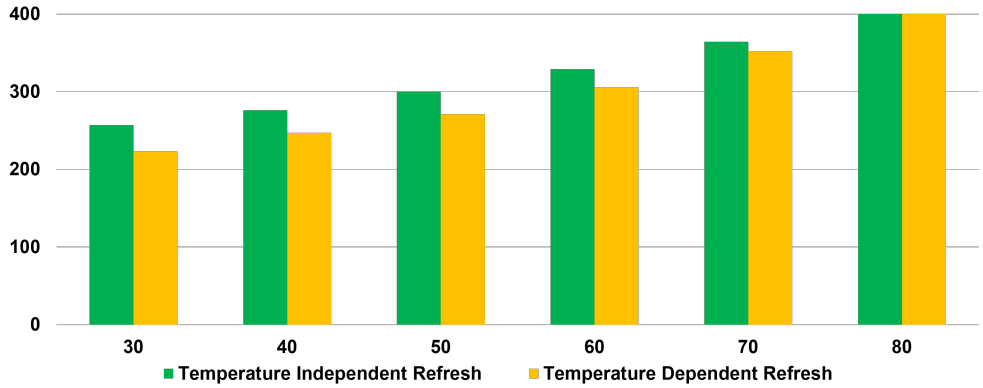


Figure 4.8: Temperature Dependance on Avaerage Power Consumption during Sleep Mode

memory temperature, the refresh assist circuit will have to periodically refresh the cell contents with Nthreshold values from 80C and uses the same for all other temperatures. Thanks to temperature feedback, the Nlife increases and for a given sleep duration, at lower temperatures we can live with fewer number of periodic refreshes. Evaluations in Figure 4.8 shows that at 30C temperature dependent pulsed periodic refresh saves approximately 20% average power.

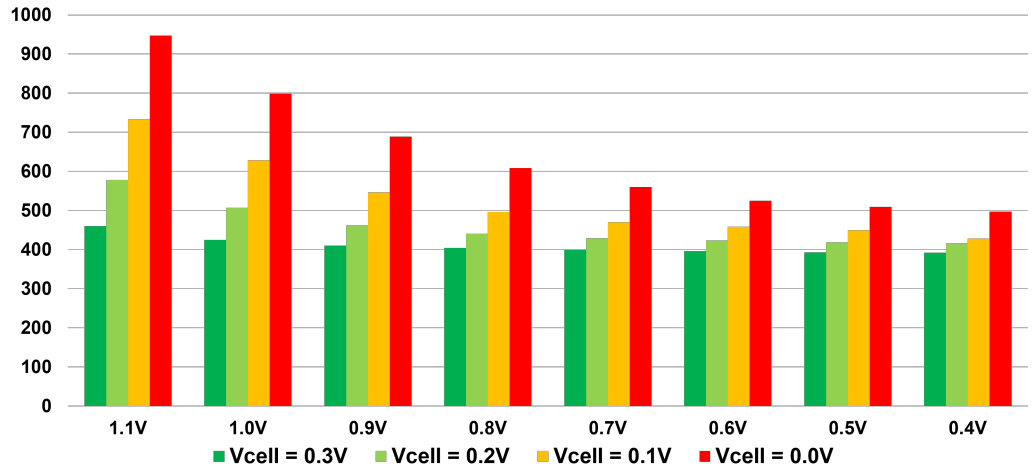


Figure 4.9: Average Power in microwatts at 80C during Sleep Mode at different Refresh Voltage

The main objective of performing a periodic refresh is to turn on the M4 PMOS transistor and pull up the node N1 to a higher value. Our evaluations shows that even for Vrefresh of 0.4V, the CN1 node is successfully refreshed. Figure 4.9 captures the average power consumption at different Vcell for various Vrefresh voltages. It is interesting to note that, if a Vrefresh of 0.4V is used to perform periodic pulsed refresh, the power benefit is 18%.

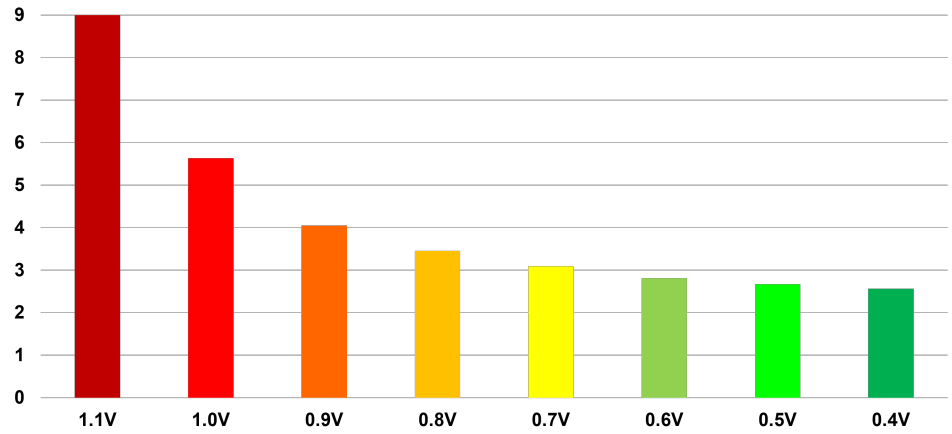


Figure 4.10: Refresh Dynamic Power in mW

Since refreshes can be potentially done with any voltage levels, it is interesting to understand the dynamic power spent during periodic refreshes for different Vrefresh values. From Figure 4.10, if we perform refresh at 1.1V, we have to pay as much as 3.6x than when done at 0.4V.

Analysis in Figure 4.11 shows the average power consumed by the sub-array in active mode for a duration of 2000ns. Baseline configuration maintains data retention voltage level when there are no accesses while the DRAM-like refresh mechanism can assume cell supply voltage of Vcell. For the purpose of worst case analysis, Vcell of 0.3V is used and refreshes are performed at Vrefresh = 1.1V whenever required. Since this analysis is for active mode, the refresh assist circuit does not pay any dynamic refresh overhead when there are read and write accesses. Even with the worst case configuration, our proposed technique outperforms baseline technique by 19%. As notified earlier, when

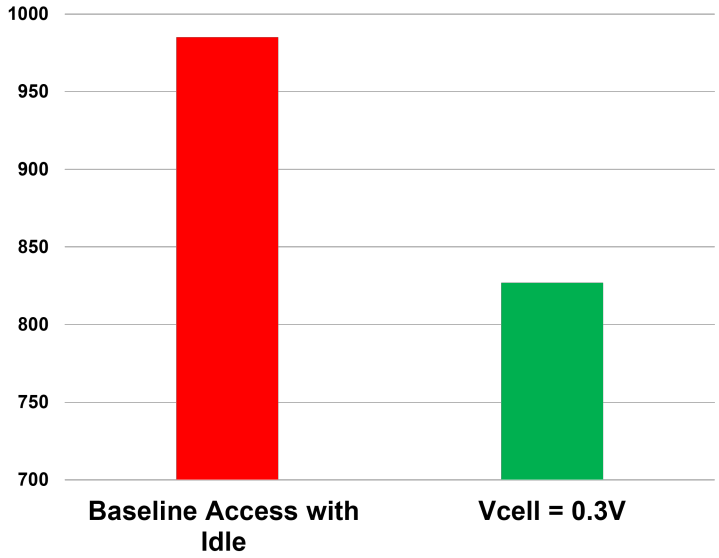


Figure 4.11: Average Power in mW during Active Mode

refreshes are performed as Vrefresh = 0.4V, the benefits are as high as 25%.

Advantages:

- The DRAM-like refresh has sleep control logic local to the sub-array and hence accurate and immediate
- Any access (read/write) before Nlife is an automatic refresh and we need not pay dynamic refresh overhead with periodic pulses
- Vcell can be as low as 0.0V when a memory intensive application is executed
- Vcell and SE can be dynamically controlled based on the demand of the application. Reduced power overhead of Vrefresh compared to periodic refresh
- During sleep mode, the refresh assist circuit just needs four refresh trigger SE pulse to restore the complete sub-array
- Potentially any idle cycle will be seen as a power saving window

Challenges: The main objective in DRAM-like refresh is to monitor life cycles and enable the refresh assist circuit to pull up Vcell to Vrefresh when needed. Challenges involved;

- Implementation of refresh trigger signal based on application behavior
- Choice of threshold counters

  a. *Local dynamic counters* - The dynamic power paid to keep track of maintaining threshold counts in four counters

  b. *Local op-amp based sensors* - Complexity in designing an accurate voltage level based op-amp circuit to measure and trigger SE before data in cell goes beyond repair

4.2 Fine-granular Column Sleep Assist - Active Mode

Motivation: Coarse-granular power management techniques identifies potential sub-arrays that can be maintained at Data Retention Voltage with the aim of reducing leakage power. Whenever the cache or subset of a cache is idle, the corresponding sub-arrays are triggered by the global sleep controller to enter sleep mode. Acknowledging the sleep request, the individual sub-arrays transition from active state to data retention state in a staggered fashion. One interesting point to note here is that, even during active state of the sub-array there are portions of the sub-arrays that are not being used and hence they consume leakage power. Conventionally, during any access even when only a 64b word is being accessed, all the four 64b words are maintained active. Understanding the scope of reducing leakage power, we propose a fine-granular sub-array power management scheme - Column Based Sleep mechanism that continuously monitors the addresses being accessed and activates only a subset of sub-array while keeping the rest of the array in data retention mode.

Strategy: Before hashing out the fine-granular sleep strategy, it is worth revisiting the write operation in a sub-array. During any write access, the column decode and address assist circuity selects one of the four columns based on the lower address bits. The column being written is powered by generated lower voltage of Vdd - Vt. The write assist circuit maintains the voltage level of half-selected or unselected columns at Vdd. In summary, during any write 3/4th of the sub-array is in read mode at a higher Vdd. We potentially convert the write assist circuit to implement a sleep assist circuit that

can maintain 3/4th of the sub-array in data retention mode while only one column is kept active.

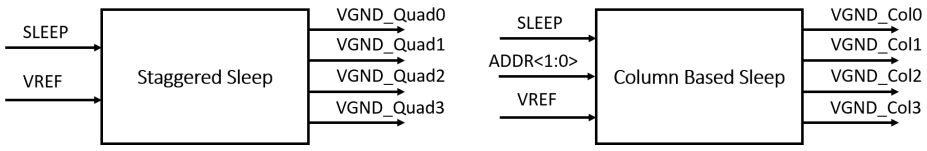


Figure 4.12: Staggered Sleep Circuitry (Left) and Fine-granular Column Sleep Assist Circuitry (Right)

Implementation: The existing staggered sleep circuitry in Figure 4.12 powers down quadrant by quadrant upon receiving sleep assert request from the global sleep controller. In sleep mode, the ground of all the 6T SRAM memory elements in all the four quadrants are raised to a Virtual Ground Reference Level Vref of 0.3V. By doing so, the cell leakage is reduced exponentially and all the memory elements retain data. This way the entire sub-array is put in data retention mode for the entire duration of sleep. After exiting sleep mode, all the quadrants are in nominal Vdd supply voltage.

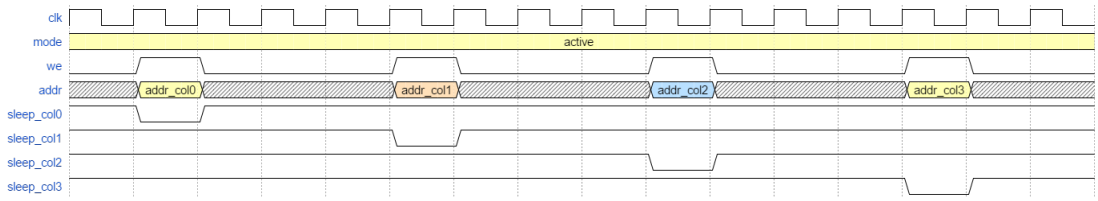


Figure 4.13: Sleep Transition in Fine-granular Column Sleep Technique

The sleep circuitry can be enhanced by modifying the quadrant by quadrant sleep to column based sleep as shown in Figure 4.12. This implementation is inspired by the way the existing Write Assist mechanism works. But in column sleep assist, based on the address being accessed, the selected column is held at nominal voltage and unselected columns can be maintained at data retention voltage. This ensures at any given point in time, only one of the four columns are active and rest can be in sleep mode as shown in Figure 4.13. This comes with the obvious advantage of eliminating the global sleep assertion and desertion and makes the sub-array self-sufficient. Additionally, transition to and from sleep mode is dynamic based on the address being written or read. By

maintaining the half-selected columns in data retention mode, even during active sub-array phase we can achieve upto 75% power savings.

The power savings comes along with the penalty of implementing the sleep circuit with an area overhead of approximately 4-5% since now every column has its own virtual ground controller and this local controller has the intelligence of switching modes based on the address decoded. The memory occupancy of the sub-array remains unaffected since these logic can be implemented in the available peripheral circuit area. The sleep penalty of the implemented circuit is 2-3 clock cycles and the wake-up penalty is half-cycle. This implies that, even for the worst case column major streaming accesses, the proposed column based sleep will not suffer penalty in comparison to the baseline configuration.

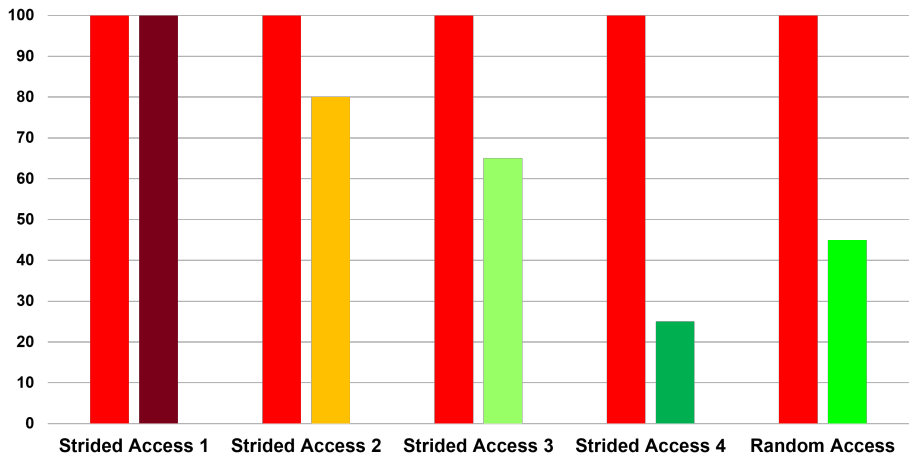


Figure 4.14: Normalized Average Power Analysis of Fine-granular Column Sleep

Results and Analyses: Figure 4.14 shows the benefit of using fine-granular sleep assist mechanism in comparison to staggered sleep. The column based sleep implementation will be performing at the best when the access pattern is such that it activates a column for a longer time in duration so that the rest of the column can be put to sleep. As expected, the "Strided Access 4" performs better in comparison to other accesses and "Strided Access 1" performs similar to baseline as the accesses keeps sweeping all the columns in subsequent cycles. Also, for a random access pattern, this method saves

approximately 55% on an average.

4.3 Bitline Segmentation - Active Mode

Motivation: Fine-granular approach to reduce power during active mode is aggressive for the fact that every cycle there is some scope for saving. While this is certainly interesting, there are additional opportunities to reduce more power even during write and read cycles. For instance, lets consider a write operation for the cells along the wordline closer to the sense amplifier. During the write cycle, the value to be written into the memory cells are kept in the bitlines and the entire long bitlines are charged even though we had to access nearby memory. This eats up write power and introduces long latency writes. Similarly, during read of memory elements closer to sense amplifiers, the entire bitlines are discharges and thus read energy is wastes resulting in dynamic power consumption. This can be avoided by breaking the bitlines and only the required length of the lines are charged and discharged [28], [29].

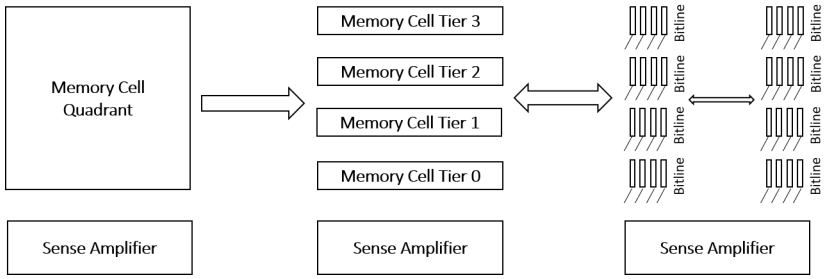


Figure 4.15: Bitline Segmentation using Isolation Transistors

Strategy: Bitlines consumes about 50% of the total cache power [27]. At any given point in time, only one of the wordlines of the sub-array is activated and irrespective of which one is accessed, the entire long bitlines are charged and discharged. These long bitline capacitors consume dynamic write and read energy. Additionally, the bitlines leak and adds to the total leakage power consumption during sleep mode. The above mentioned dynamic and leakage can be significantly cut short by segmenting the bitlines using isolation transistors. These isolation transistors can be dynamically enabled based on the address being accessed.

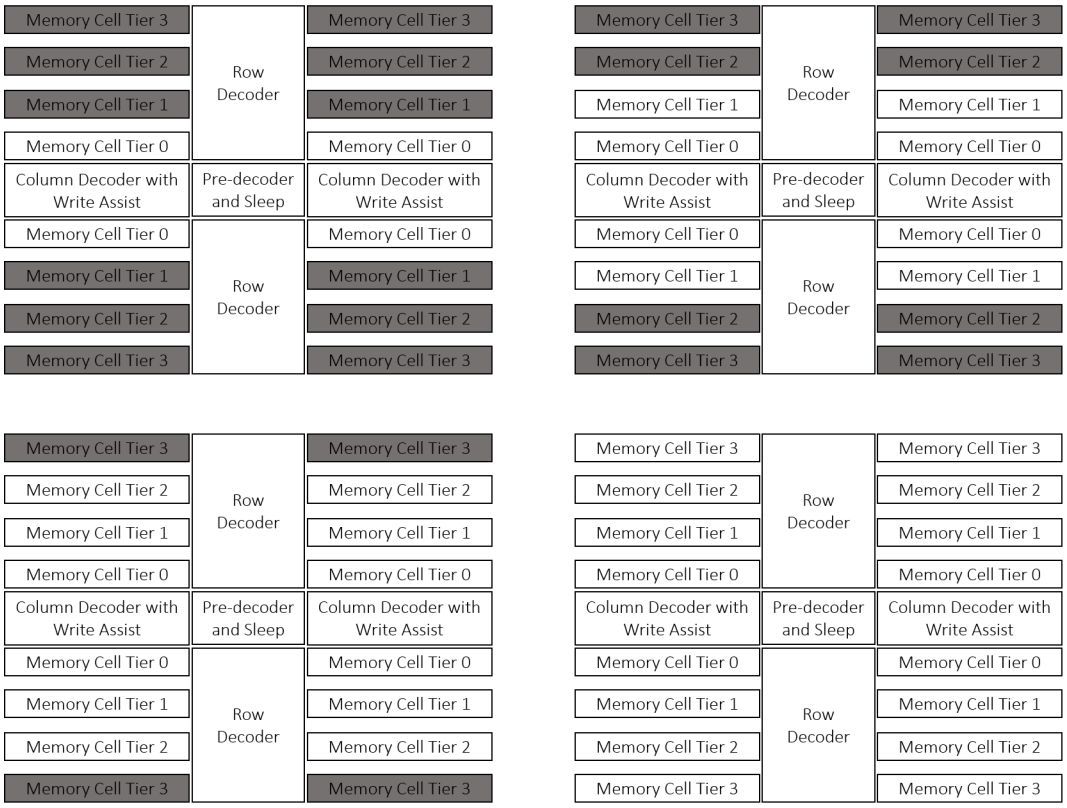


Figure 4.16: Near-memory (white) and Far-memory (grey) regions for different configurations.

Implementation: Thanks to bitline segmentation, the sub-array can be divided into near-memory and far-memories as shown in Figure 4.15. NMOS, PMOS or transmission gates can be used to segment the long bitlines. After segmentation, for accessing the elements closer to sense amplifier, only the near-memory is enabled and far-memories can be isolated to save dynamic read and write power. Figure 4.16 shows near-memory and far-memory regions for different configurations. The bitlines in the far-memories will be floating and hence reduce the leakage power. Additionally, all the isolation transistors can be turned-off by disabling all the segments essentially making the complete sub-array as far-memory. This case is similar floating bitline [cite] implementation. Therefore, during data retention mode, all the bitlines will be floating and leakage power from bitlines can be completely eliminated.

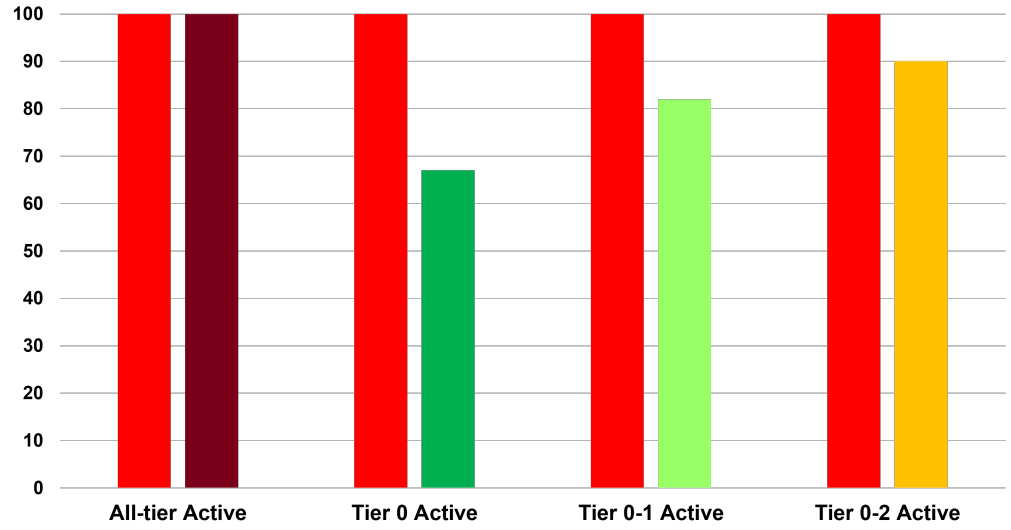


Figure 4.17: Normalized Average Access Power Analysis of Bitline Segmentation

Results and Analyses: Bar graph in Figure 4.17 shows the normalized dynamic read and write power savings of segmented bitlines in comparison to the baseline configuration. When the near-memory consists of only a few wordlines, approx. 33% access energy is saved. In contrast, when the near-memory consists of three active tiers, the access power savings is as low as 10% over the baseline. In summary, smaller the near-memory, maximum is the power savings due to segmentation.

Challenges:

- Inserting isolation transistors in the bitline and bitline-bar without impacting the memory occupancy and regularity
- Dynamically changing the firing of the sense amplifier enable signal based on the number of tiers activated
- On the fly isolation transistor control based on the address being accessed

Chapter 5

Architectural Exploration and Future Work

Adaptive power management schemes introduced in the previous chapter achieves dynamic and leakage power savings irrespective of the application access pattern. The local sleep circuits adapts to the address being accessed and triggers low power mode upon possible idle cycles. Our initial evaluations shows that each of these techniques can perform more efficiently if these circuits can dynamically reconfigure based on the application access pattern. We present in the following sections about the architectural study and propose re-configurable power management schemes.

5.1 Memory Access Profiling

The sub-array used to build different levels of cache hierarchy are the same. If we use static schemes as proposed in chapter 4, we may not be able to properly capitalize a variety of sleep regions for different levels of cache hierarchy. There is a need to explore memory access pattern and adaptability trigger different power management schemes. Based on the exploration, power management scheme has to be identified for L3, L2, L1 and other SRAM based structures like registers, TLBs etc. Below is a summary of interesting identification after performing architectural evaluations in CMP$im [29] and from literature survey.

- L3/L2/L1 Accesses are non-deterministic
- L3 Cache is idle for 99% of time
- L2/L1 Idle duration is less, idle frequency is high
- L3/L2/L1 Frequent accesses on a subset of cache lines
- L3/L2/L1 More read and less write operations

5.2 Future Work

Proposed adaptive power management techniques can perform better if these techniques can be tuned based on the demand of the application. The following sections presents an outline of possible future work that can be experimented.

5.2.1 DRAM-like Refresh in SRAM

Choice of controlling parameters: Our initial evaluations showed that DRAM-like refresh technique can be used even during active phase of sub-array and achieves better power reduction. There are multiple parameters that determine the overall performance of DRAM-like refresh policy. Vcell, SE and Vrefresh can be dynamically chosen and Refresh Assist Circuit can be reconfigured on the fly based on memory access pattern. Access based active feedback on recommended settings will also vary based on the cache hierarchy. At higher levels of cache hierarchy, most of the sub-arrays will be active and Vcell can be set to 0.0V and SE frequency will be less. Reads and writes will keep the data refreshed in a timely fashion and the dynamic refresh overhead will be minimum. For LLC, the cache will be idle for longer duration and Vcell of 0.3V and high frequent SE will be required.

Static vs Dynamic Configuration: The modeled refresh assist circuit will perform column by column refresh based on the idleness of the accesses to the column. In active mode, the address being accessed determines the number of refreshes and the column to be refreshed. There can be different address mapping schemes possible in the

given sub-array design. It will be interesting to understand how each of the following mapping schemes will benefit DRAM-like refresh.

- *Least Significant Address Bits* - If LSB bits are chosen for column multiplexing, DRAM-like refresh is expected to perform better. This is due to the fact that, there will be more occurrences of reads and writes and this is an automatic refresh to keep the data alive.
- *Most Significant Address Bits* - With MSB bits, the benefits of DRAM-like refresh is expected to be average since subsequent accesses will most of the time be pertained to the same column and the threshold counters of unselected columns will be busy monitoring and refreshing periodically to keep the contents of memory cells in their control un-corrupted.
- *Other Address Bits* - There can be other combination of address bits that might be performing better with DRAM-like refresh. These bits can be different for different application and different levels of cache. It will be interesting to do a profiling on the collected trace and find a recommended 2 bit combination that gives maximum benefits for a variety of commonly executed applications.

5.2.2 Fine-granular Column Sleep Assist

Our evaluations were based on the baseline address mapping for column multiplexing that used LSB bits. Benefits can be different if address mapping schemes are changed. We intend to study the behavior of our fine-granular column sleep under different address mapping schemes. The objective is to use profiling to determine the address mapping that keep accesses longer in one column so that majority of the columns can be maintained at data retention during the execution of a application. It will be interesting to understand how each of the following mapping schemes will benefit fine-granular column sleep.

- *Most Significant Address Bits* - If MSB bits are chosen for column multiplexing, row major access will be converted to column major accesses and consecutive accesses will be along the same column. With a column major access pattern, the length of time a single column is kept active is longer and the number of frequent

switches to other columns will be less. This mapping scheme is expected to give higher benefits for any commonly executed application.

- *Least Significant Address Bits* - With LSB bits, the benefits of fine-granular column sleep will be less. While the saving will be higher in comparison to the baseline data retention staggered sleep, since the accesses will be row major, there will be frequent jumps to adjacent columns. Most of the time, the sub-array will spend energy in transitioning to sleep mode and the latency involved on setting up sleep mode will overkill the benefits that would have been originally achieved without frequent transition.

- *Other Address Bits* - There can be other combination of address bits that might be performing better. These bits can be different for different application and different levels of cache hierarchy. It will be interesting to do a profiling on the collected trace and find a recommended 2 bit combination that gives maximum benefits for a variety of commonly executed applications.

5.2.3 Bitline Segmentation

The benefits of bitline segmentation will be maximum if the accesses to the sub-arrays are closer to the sense amplifiers. As noted earlier from our architectural study, accesses to the sub-arrays are non-deterministic and only a subset of sub-arrays are frequently accessed. If we can perform a study and determine the cachelines that are accessed frequently, they can be as well mapped closer to the sense amplifier. This can be achieved in two ways;

- *Static Mapping*: Use profiling to choose the address mapping to keep the frequently accessed cache lines closer to the sense amplifiers

- *Dynamic Mapping*: Find the frequently accessed sets and group those sets closer to sense amplifier to form near-segment. This requires dynamic re-mapping as per access trend. This is certainly challenging because, to bring hot cachelines to near-memory, we need to perform either swapping of chosen cacheline from far-memory to least used near-memory or flushing of unused near-memory

Prior works along the same lines were focused on first level cache [28] and DRAM [26]. We plan to extensive experiment at various levels of cache hierarchies with segmented sub-arrays for a variety of commonly executed application.

5.2.4 Summary

Based on the scope of future work discussed, we intend to perform the following analysis and experiments;

- Performance study of different addressing mapping schemes (static vs. dynamic) for all three techniques
- Back annotating the read, write, idle and refresh power in simulator to find the power consumption for real benchmarks
- Combine all three techniques to find sweet spot recommendation for:
    a. different applications
    b. cache hierarchies

Chapter 6

Conclusion and Discussion

In order to maximize the SRAM leakage and dynamic power saving with reliable data retention, this work explores the limit of SRAM data preservation and introduces adaptive aggressive power management schemes that makes the SRAM sub-array independently enter and exit low power mode without additional architectural support from global sleep controllers. In addition to achieving leakage power during sub-array sleep mode, we introduce methods that can potentially achieve additional power savings by partially keeping portion of sub-array in data retention mode to reduce dynamic and leakage power during active operation of cache. Each of these methods achieve significant improvement over the baseline configuration and save average power without compromising the performance of the cache. On an architectural level, we use the results from application demand explorations to tune the local power management controllers to dynamically adapt schemes to improve leakage and dynamic power savings. We also recommend combinations of these adaptive power management schemes for different levels of memory hierarchy after profiling the memory access pattern of various workloads. Recommendations;

- *DRAM-like Refresh*: L3 —Maximum benefits, L2/L1 —Average benefits
- *Fine-granular Column Sleep*: L1 —Maximum benefits, L2 —Average benefits, L3 —Minimal benefits
- *Bitline Segmentation*: L1 —Maximum benefits, L2 —Average benefits, L3 —Minimal benefits

References

[1] M. Horowitz, Computings energy problem (and what we can do about it), in Proceedings of the IEEE International Solid-State Circuits Conference, Feb. 2014, pp. 1014. DOI: 10.1109/ISSCC.2014.6757323. 50

[2] Balasubramonian, Rajeev, Norman P. Jouppi, and Naveen Muralimanohar. "Multi-core cache hierarchies." Synthesis Lectures on Computer Architecture 6.3 (2011): 1-153.

[3] E. Karl et al., A 4.6 GHz 162 Mb SRAM design in 22 nm Tri-Gate CMOS technology with integrated read and write, IEEE J. Solid-State Circuits, vol. 48, no. 1, pp. 150158, Jan. 2013.

[4] Y. Wang, et aI., "A 4.0GHz 291Mb Voltage Scalable SRAM Design in a 32nm High-k + Metal-Gate CMOS Technology with Integrated Power Management," IEEE J. of Solid-State Circuits., vol. 45, no. 1, pp. 103-110, Jan 2010.

[5] Kim, Nam Sung. Circuit and microarchitectural techniques for processor on-chip cache leakage power reduction. Diss. University of Michigan, 2004.

[6] A 1.1 GHz 12/Mb-leakage SRAM design in 65 nm ultra-low-power CMOS technology with integrated leakage reduction for mobile applications, IEEE J. Solid-State Circuits, vol. 43, no. 1, pp. 172179, Jan. 2008.

[7] SRAM design on 65-nm CMOS technology with dynamic sleep transistor for leakage reduction, IEEE J. Solid-State Circuits, vol. 40, no. 4, pp. 895901, Apr. 2005.

[8] T. H. Kim, J. Liu, and C. H. Kim, A voltage scalable 0.26 V, 64 kb 8T SRAM with Vmin lowering techniques and deep sleep mode, IEEE J. Solid-State Circuits, vol. 44, no. 6, pp. 17851795, Jun. 2009.

[9] M. Powell et al,. Gated-Vdd: A Circuit Technique to Reduce Leakage in Deep-Submicron Cache Memories. IEEE-ISLPED 2000.

[10] K. Nii, et al. A low power SRAM using auto-backgate-controlled MTCMOS. In ISLPED, 1998, pp. 293-298.

[11] C. H. Kim et al,. A forward body-biased low-leakage SRAM cache: device, circuit and architecture considerations. IEEE Trans. on VLSI Systems, vol. 13, 2005, pp. 349-357.

[12] A. Agarawal et al., DRG-Cache: A Data Retention Gated-Ground Cache for Low Power, DAC 2002. pp. 473-478.

[13] K. Flautner et al,. Automatic performance setting for dynamic voltage scaling. in Journal of Wireless Networks, pages 260271, 2001.

[14] D. Marculescu. On the use of microarchitecture-driven dynamic voltage scaling. In Workshop on Complexity-Effective Design, June 2000.

[15] Pentium M processor on 90 nm process with 2-MB L2 cache. Intel Corp. .Jan. 2005. www.intel.com/design/mobile/datashts/302189.htm.

[16] K. Nii et al., A 90-nm low-power 32 KByte embedded SRAM with gate leakage suppression circuit for mobile applications, IEEE J. Solid-State Circuits, vol. 39, pp. 684-693, Apr. 2004.

[17] Y. Takeyama et al,. A Low Leakage SRAM Macro with Replica Cell Biasing Scheme. IEEE Journal Of Solid- State Circuits, Vol. 41, No. 4, April 2006.

[18] K. Flautner et al,. Drowsy caches: simple techniques for reducing leakage power. IEEE ISCA, 2002.

[19] S. Kaxiras et al,. Cache decay: exploiting generational behavior to reduce cache leakage power. IEEE-ISCA, 2001.

[20] M.D. Powell et al,. Gated Vdd: A circuit technique to reduce leakage in deep-submicron cache memories. in Proc. IEEE ISLPED, 2000.

[21] D. Nicolaescu et al,. Fast Speculative Address Genera-tion and Way Caching for Reducing L1 Data Cache Energy. Proc. IEEE ICCD, 2006.

[22] R. Bai et al,. Total leakage optimization strategies for multi-level caches in Proc. ACM Great Lakes symposium on VLSI, 2005.

[23] H. Homayoun et al., Adaptive techniques for leakage power management in L2 cache peripheral circuits, in ICCD, 2008, pp. 563569.

[24] Li, Sheng, et al. "CACTI-P: Architecture-level modeling for SRAM-based structures with advanced leakage reduction techniques." Computer-Aided Design (ICCAD), 2011 IEEE/ACM International Conference on. IEEE, 2011.

[25] D. Lee, Y. Kim, V. Seshadri, J. Liu, L. Subramanian, and O. Mutlu, "Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture," in HPCA, 2013.

[26] Bhati, Ishwar, et al. "DRAM Refresh Mechanisms, Penalties, and Trade-Offs." Computers, IEEE Transactions on 65.1 (2016): 108-121.

[27] R. Rao et al., Exploiting non-uniform memory access patterns through bitline segmentation, in WMPI, 2006.

[28] K. Ghose and M. B. Kamble, Reducing power in superscalar processor caches using subbanking, multiple line buffers and bitline segmentation, in International Symposium on Low Power Electronics and Design, 1999, pp. 7075.

[29] A. Jaleel, R. Cohn, C. K. Luk, B. Jacob. CMP$im: A Pin-Based OnThe-Fly Multi-Core Cache Simulator. In MoBS, 2008.

[30] Stine, James E., et al. "FreePDK: An open-source variation-aware design kit." Microelectronic Systems Education, 2007. MSE'07. IEEE International Conference on. IEEE, 2007.