Exploring Energy, Accuracy and Cost Trade-offs in Cache Architectures for Approximate Computing

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

Vinayak Bhargav Srinath

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

David J. Lilja

June, 2015

Acknowledgements

I would like to thank Prof.David J. Lilja for his guidance and patience during my Master's research in Approximate Computing. His motivation, direction and trust in my abilities have helped me complete my work with ease.

I am also thankful to the rest of my thesis committee Prof.John Sartori and Prof.Antonia Zhai, for their suggestions and insight. The C-SPIN research group under Prof.Lilja namely William Thuoy and Cong Ma, have provided me with very helpful direction with regards to my thesis and I'm grateful for their help.

I would also like to thank M. Shoushtari and Alireza Shafaei Bejestan for their help in regards to gem5 and PCACTI simulators

Last but not the least I would like to thank my parents, Srinath.A.L, Ambujakshi.S.P and my sister Vaishnavi for their love and encouragement.

This work was supported in part by National Science Foundation grant no. CCF-1438286. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

Dedication

I dedicate this thesis work to my loving parents and sister who have always supported me.

Abstract

A processor's power consumption can be most efficiently reduced by lowering the supply voltage. But with reduced voltage levels comes the major concern of failure of memory circuits. ASIC designers define a minimum operable voltage of the processor's on-chip cache often referred to as the Vcc_{min} which is the voltage level below which the processor's memory-subsystem is no longer reliable. This guard-banding mechanism adds an additional overhead on the processor's memory-subsystem which does not allow it to operate below this voltage, and its important to note that the processor's memory-subsystem is one of the major contributors of its overall energy consumption. Guard-banding mechanisms are not just limited to increased minimum operable voltages and they result in large overheads. If certain restrictions are relaxed on the reliability of the output we can obtain significant savings in energy by eliminating these guard-banding mechanisms. This work explores different configurations of architectures suitable for low voltage operation of image and video applications by outlining the energy, accuracy, area and performance trade-offs.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The current trend of CMOS technology has been towards scaling and driving down the transistor sizes to attain higher density and thus be able to add more and more functionality on the chip. Reduced chip area helps reduce costs and over the years the transistor has scaled down according to the Moore's law but the processor's operating voltages have not scaled down in a similar fashion [2]. This has resulted in higher power consumption and leakage. On-Chip memories have been hogging the power and area resources of the processor.

Recent processors incorporate multiple cores and each of these cores have their own private caches and shared caches which are generally CMOS SRAM cells. With the increase in cache sizes over the years and the increase in hardware variability due to reduced feature sizes as a result of technology scaling, has resulted in the need for Memory gaurd-banding by various methods. The V_t fluctuations at lower technology nodes , the short Channel effect [3] etc., leads to an over-design and causes a lot of overhead such as Error Correcting Cache, Higher Cache Supply Voltages etc. Increased error correction abilities result in increased area and higher supply voltage causes higher leakage in the SRAM cells.

However with the shift towards mobile technologies in the recent years have led the ASIC Design engineers to be conscious of the power budget of the chip. The focus has shifted towards architectures that are more power efficient and employ novel methods to explore this domain. The most effective approach to reducing the power consumption on-chip is by reducing the voltage because dynamic power is a quadratic function of

voltage and leakage power is exponentially dependent on voltage [4]

Processor's power consumption has been the most important aspect in modern day mobile technologies and with the growing need to have more and more devices becoming portable, the need to increase the efficiency of the processors with regards to its energy utilization has grown. The key factor in consideration of design of embedded/mobile or wearable technologies has been its battery life. As the semiconductor industry pushes down the transistor sizes in accordance with the Moore's law , the device variability has gone up. ITRS predicts the variability to increase over the coming decades[5]. Thus, power reduction in processors has become a challenging task due to the voltage margins that are introduced to allow for error free operation of memories. If we relax the constraints on the reliability of the hardware by allowing for accuracy and performance trade-off due to low voltage operations, we can obtain significant energy savings.

Computing platforms are designed based on the principle of exactness and that all computations must be executed in a precise manner. Approximate computing deviates from this approach and is application aware, wherein it exploits intrinsic application error resilience and takes into consideration perceptible quality of the output [6] Several applications are inherently error tolerant, thus, by capitalizing on such factors, we can work with faulty and unreliable hardware.

In this work, we use Hybrid memories which are a combination of 8T [7] and 6T SRAM cells for storing approximate data and a 6T SRAM array at higher voltage to store data which is critical and cannot be approximated. First we analyze the impact of approximation on the output because of voltage lowering and also the impact on output quality with increased area due to increased 8T cell count in the Hybrid array. We conclude that there is an optimal ratio for 8T and 6T cells in the Hybrid array and it has been used in this architecture. This architecture is targeted towards Image and Video applications and the reasons for this are discussed in future chapters. We also a discuss a scheme to map data to reliable or unreliable accordingly to avoid application failures. Lastly we discuss the trade-off for Accuracy and Cache capacity for Energy reduction in detail by defining identifying a suitable metric for output fidelity and compare different proposed cache architectures.

- Chapter 2 introduces the concerns of scaled CMOS technologies and the motivation behind the need for low voltage architectures and some of the different

architectures that are relevant.

- Chapter 3 discusses the proposed architecture and the functionality.
- Chapter 4 discusses the experimental setup, the simulator modifications and the simulation methodology.
- Chapter 5 talks about the applications that are suitable for approximation and where approximation can be handled.
- Chapter 6 analyses the results for the benchmarks and its impact on Energy, Accuracy, Area and performance by comparing and contrasting between different configurations.
- Chapter 7 concludes the work by interpreting the results from the analysis of different configurations presented in this thesis.

Chapter 2

Background and Motivation

The following sections briefly describe the need for new energy efficient architectures that aware of application's properties to tolerate errors and introduces approximate computing as a field along with some examples of recent work on energy accuracy trade-offs. It also provides basics of SRAM failures and hybrid memories.

2.1 Approximate Computing

Several techniques [8, 9, 10] are introduced to mitigate hardware concerns regarding reliability and many software mechanisms [11, 12] are being explored to perform error recovery. However all these techniques are done in an application agnostic environment where software and hardware are decoupled from one another. It is useful to note the trend in hardware manufacturing has evolved over time and the traditional architecture using reliable CMOS has changed and now includes some software error correction and hardware error correction while using unreliable structures. But, it is necessary to understand that several applications are inherently error tolerant [6] which can use unreliable memories and reduce these guard-banding overheads.

Approximate computing is an emerging design perspective which capitalizes on these error resilience properties. Error resilience is property shared among a large set of applications such as image, audio and video processing, communications, data mining, computer vision and search. These applications have very few side effects as a result of memory faults and with proper identification of regions of application which are error

tolerant we can achieve significant energy savings.

2.2 Recent work

There has been an increased interest in the field of approximate computing and this has sparked extensions to existing programming languages like Java for energy efficient computation[13] which helps add new data-types for approximate data and also enforces very expressive rules for handling of these data-types and demonstrate a savings of 10-50% in energy.

Energy reduction in main memory has been shown to be effective when there is a tolerable amount error. It has been shown that we can leverage trade-offs in energy and hardware correctness [14] and save 20-25% energy consumption in a mobile device with minimal changes to hardware.

Adding ISA extensions to handle regions of code for which error recovery has to be performed [15], moves the burden of error recovery to software and relaxes the hardware recovery support by providing a hardware organization that has helped relax the reliability considerations of the hardware. It also allows programmers the ability to utilize the ISA by providing compiler support. Similar work which extends on adding approximate data-types [13] by providing a micro architectural design that supports the ISA extensions shows potential power savings with acceptable QoS degradation.

QoS is an important metric as we have to ensure so that the approximation introduced does not cause the results to be indiscernible and is within specified bounds. Dynamically adapting the level of approximation to ensure the QoS constraints specified by the programmer by monitoring the run-time behavior [16] has shown improved performance and energy consumption.

Reducing the Cache capacity to allow for low voltage operation [17] by disabling cache-words that are failing at low voltages allows for very low voltage operation and saves significant amount of energy. As discussed previously, error tolerant qualities of a large set of applications help in utilizing the faulty regions of cache rather than causing a reduction cache capacity [18] by providing knobs to the programmer to control the level of approximation based on voltage levels. Creating a scheme for mapping of data into appropriate locations and drive the programmer to notify the regions of application

which can tolerate error can help achieve significant leakage energy reduction due to voltage lowering in the SRAM Cache.

The work described in thesis focuses on Image and Video applications. However, we believe that it can be extended to other forms of applications as well. Image and Video data, which is stored in the form of pixels, has a larger dependence on the MSB bits of the pixel data as it contains a majority of the information. Hence using different tolerance levels to different bit-positions can help improve the output quality and achieve even further voltage reduction. This priority based scheme [19] having different ratios of reliable and unreliable bits in the cache has been shown to have very promising results with respect to costs as compared to using a completely reliable memory which has high area cost to output accuracy.

2.3 SRAM Failure

The Hybrid Cache structure[19] used in this work comprises of 8T and 6T SRAM cells(see Figure 2.1). A brief description of both these cell structures are given below along with the SRAM failure mechanisms for both and the resulting bit error rate graphs.

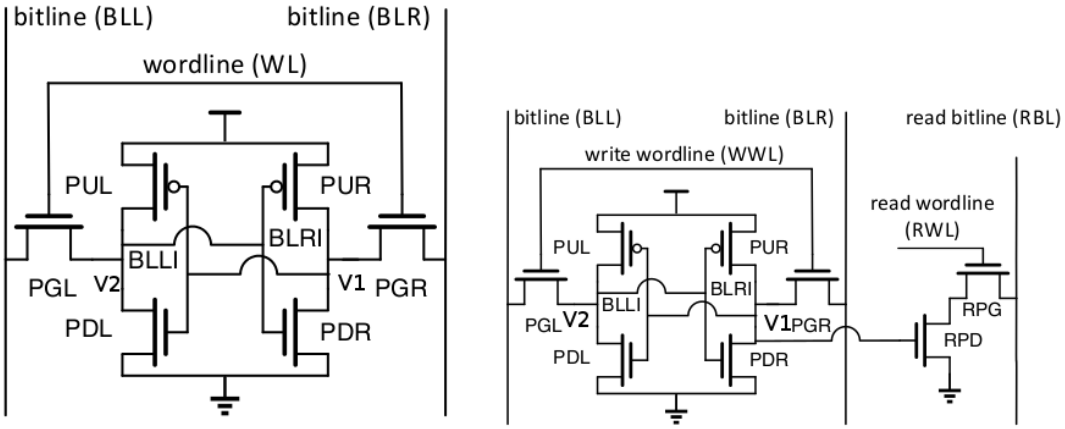


Figure 2.1: 6T and 8T SRAM cell schematic

The SRAM cell is a memory element which stores either a 0 or 1 at the node V1. Data is read or written into the cell by using the wordline(WL) and bitlines(BL*). During

the SRAM read operation, both bitlines are precharged and then disconnected from the precharge circuitry. The cell contents are then read out by activating the wordlines. The differential pair formed across the BLL and BLR due to the node voltages V1 and V2 is sensed by external circuitry and helps determine the contents of the cell. The write operation is done by pulling down one of the bitlines and then writing the value to the cell by turning on the wordline.

The SRAM cell has to be designed appropriately so as to avoid cell failure due to hardware fluctuations. A larger device size in general helps counteract the hardware fluctuations but the cost of the chips go high with increased area. Thus designers have to work with area constraints while designing the cell. The standard 6T SRAM cell faces conflicting conditions while sizing the PU, PG and PD transistors for read and write operations and is susceptible to parametric variations. The 8T as shown in the Figure 2.1 is identical to the 6T cell albeit the addition of two transistors RPD and RPG which are solely used for read. This decouples the read and write operations and helps improve the SNM of the cell (Figure 2.2).

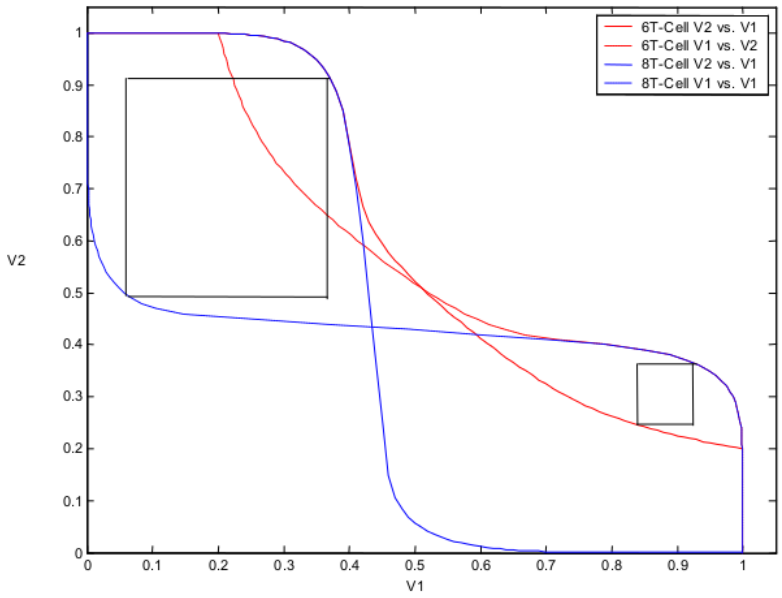


Figure 2.2: SNM Curves for 6T and 8T cells

The SNM is the Static Noise Margin of the cell and the size of the window in this

curve determines the maximum amount of voltage variation that the cell can tolerate while assuring reliable operation. Parametric variations causes the window size to shrink and at a certain voltage level the cell would no longer operate reliably. As it can be seen the 6T cell has a smaller SNM for the same voltage as a 8T cell. However the 8T cell is much larger than the 6T cell and has 30% area overhead.

When the cache structure is fabricated, all the cells are not exactly the same and this can be attributed to hardware variations. This causes some cells in the cache to be less robust than the others. The voltage level that assures with a certain confidence that no cells are faulty is the minimum operable voltage called the Vcc_{min} for the L1 Cache its generally 700-725mV. This voltage dependent failure probability for a cell is called the *Bit Error Rate*(BER). The SRAM failure rates for 8T and 6T are shown below in Figure 2.3(as per data from [20]).

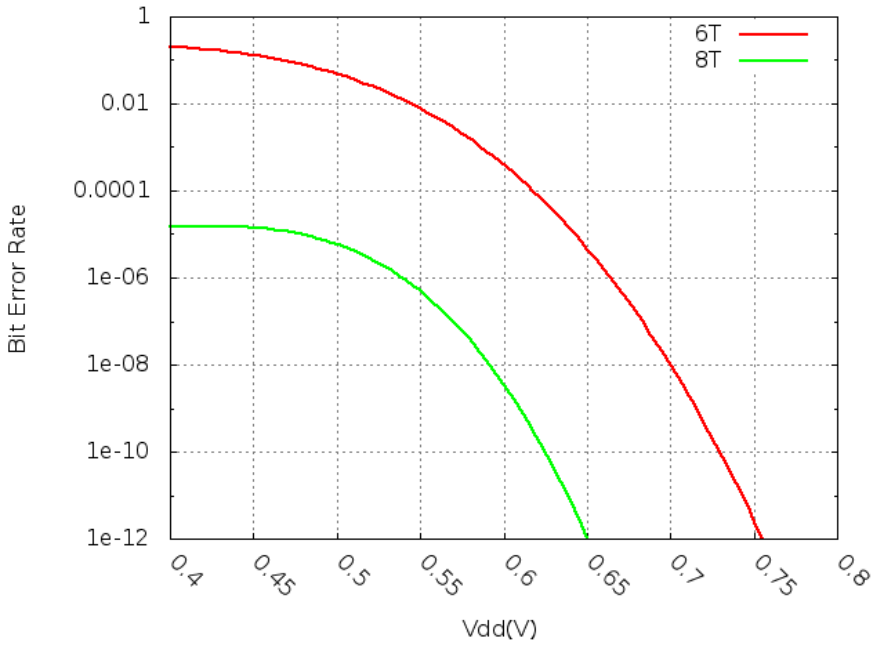


Figure 2.3: Bit Error rates of 6T and 8T SRAM cells

In this work we use a cache which has both 8T and 6T SRAM cells called a Hybrid cache and we define an optimal ratio of these 8T to 6T cells for this Hybrid cache in chapter 3.

Chapter 3

Architecture

We discuss in this chapter the programmer decisions on data classification and the knobs provided to the programmer to help achieve the required output quality and the software support to accomplish this. We also discuss the hardware architecture utilized [18] to map the data that is partitioned into *approximate* and *precise* into their respective locations. We also discuss the incorporation of Hybrid memories into the cache architecture.

3.1 Programmer Control

While using unreliable memories it is vital for the programmer to map the data accordingly. The programmer has to decide the data which is non-critical and those that are critical to avoid unexpected program behavior.

3.1.1 Data Classification

Accurate Data Classification and Mapping is the first step towards low voltage computation. For example, one cannot store the instructions to be executed in unreliable regions of memory. Pointers and data that effect the control flow of the program should not be approximated. There are existing works[13] that introduce checks to provide a safe and general programming construct. However, this work relies on the programmer's knowledge to determine the data which is non-critical and critical and the compiler is modified to interpret the language extensions.

3.1.2 Knobs to Control Output Quality

The programmer should also be able to fine tune the output accuracy for acceptable results and hence obtain an optimal trade-off of energy for quality because always operating at the lowest available voltage is not ideal and sometimes results in increased execution times. Here, the programmer is given the ability to set the cache voltage levels at either V_H, V_M or V_L which are high, medium and low voltages respectively. More importantly, it is necessary for the programmer to keep in mind that switching voltages causes several cycles of stall as the cache has to be flushed so as to ensure that the critical data is not corrupted.

3.2 Hardware

The hardware architecture for supporting the tune-able approximation has been described in the Figure 3.1 below and is marked by the blue border along with the modifications to existing cache architectures denoted by shaded gray blocks.

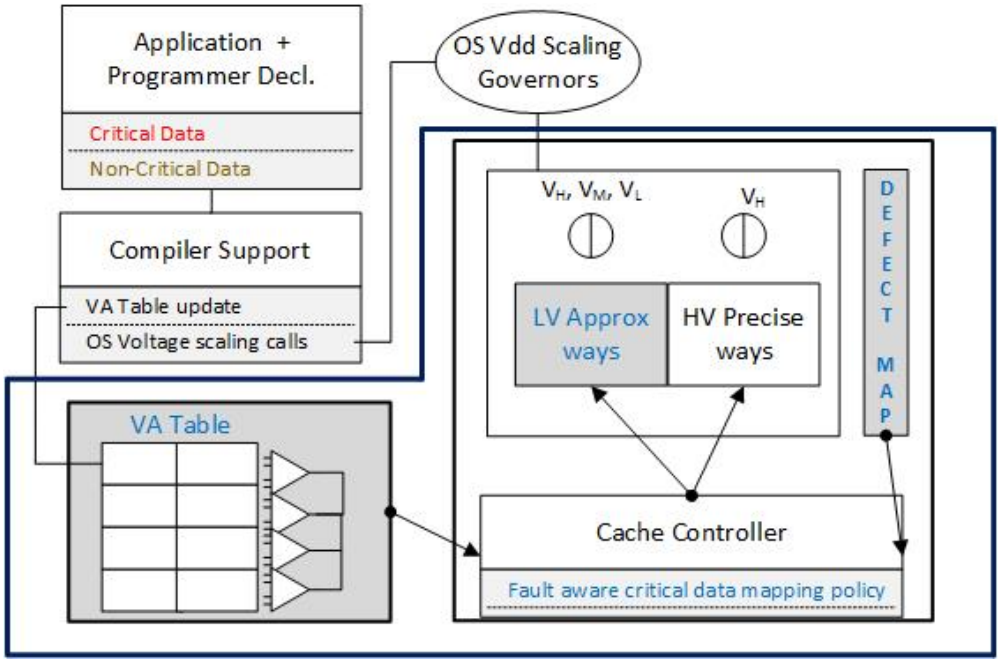


Figure 3.1: Hardware modifications to existing cache architecture

3.2.1 Virtual Address Table

The Virtual Address Table (VAT) is an on chip register file to denote the address space in the program that can be approximated. It contains a start address and end address and a validity bit as shown in the Figure 3.2. Before the data is mapped onto the

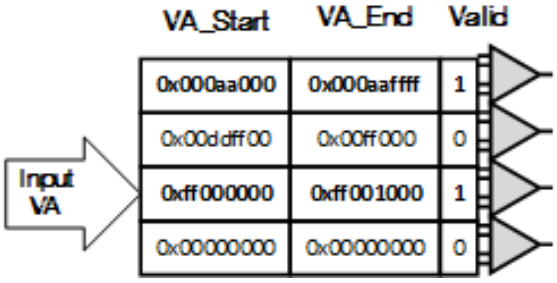


Figure 3.2: Virtual Address Table (VAT)

appropriate block, the virtual address of the data is run through the VAT which checks if the incoming address is within range of any of the VA_Start and VA_End blocks using a set of comparators. The Validity bit is used for re-use of these registers. When a new approximate declaration of a block of data is made, it is entered into the VAT with Valid=1, once this block goes out of scope or the approximate block is freed(block has to be *undeclared* as approximate) the Valid bit is set to 0. The entries with Valid=0 are potential replacement candidates for new declarations of approximate blocks.

If the Input VA is within range of any of these sets of address spaces and the Valid=1 then it is deemed as an approximate data and can be mapped to the unreliable parts of the memory by the cache controller.

3.2.2 Defect Map

As discussed previously, memories operate above a voltage $\mathrm{Vcc}_{\mathrm{min}}$ that ensures fault free operation. But in this architecture, where the cache operates at a low voltage, memory faults can occur. It is necessary to identify the faulty locations and keep a track of these so that we don't accidentally map critical data into these faulty locations. The defect identification granularity is set at the cache block level. A defect map has been introduced [17][18] that represents if the cache block is defective at a given voltage. In this work we employ three different voltage levels which the approximate bank of ways

can switch to: V_H (High Vdd), V_M (Medium Vdd), V_L (Low Vdd). A list of possible values in the defect map for a block is listed in the Table 3.1. For example, the defect map value of '01' implies that the corresponding cache block has no faults for V_H and V_M voltage levels and is faulty at lower voltages.

Defect map value	Clean Voltages
00	V_H
01	V_H, V_M
10	V_H, V_M, V_L

Table 3.1: Defect map values and corresponding Vdd levels

This Defect Map can be an on-chip memory that is populated at the processor boot-up using a BIST mechanism like a March Test[21] running at V_H, V_M, V_L and finding out the voltage level at which the cache block becomes faulty . If this is stored in a non-volatile memory it need not be performed at every boot-up of the processor. Since we are using just 2bits per cache block of size 64B to keep a track of the block's health, the overhead is negligible. The Defect Map size is less than 0.2% of the cache size.

3.2.3 Cache Controller

The cache controller has to aware of the type of the data, i.e if the incoming block for replacement is a critical block or not and also where to map the incoming block of data. The cache-controller makes this decision based on (1) current voltage level of the approximate ways of the cache, (2) the defect map (which denotes whether a cache block is clean or faulty) and (3) the VA table (which denotes whether the incoming block for replacement is a critical block or not). We define here a criticality aware LRU replacement policy and we have explored two kinds of replacement policies. Block replacement is not in the critical path for read and write access and hence does not impact the delay. The performance with regards to misses for these two methods have been explained in chapter 6. The simple flow of each of these policies are shown in Figure 3.3.

- **Split LRU** where the critical blocks are stored only in the *Precise*(High Voltage) ways of the cache
- **Shared LRU** where the critical block can be stored either in *Precise*(High Voltage) ways or in the *Approximate*(Low Voltage) ways of the cache(see Figure 3.1) provided the block is not faulty.

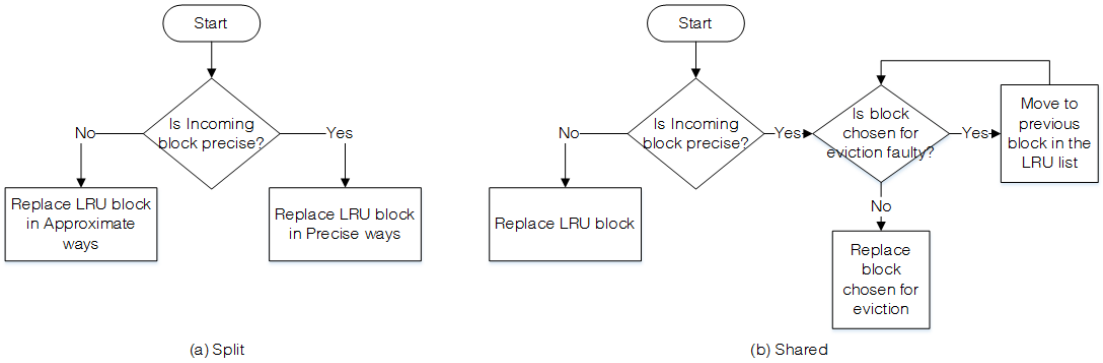


Figure 3.3: Replacement Policy of Cache Controller

3.2.4 Cache Architecture

The L1-DCache has been modified into two separate sections. For a cache that is N-way set associative, k-ways are chosen as high voltage precise ways which are held at V_H to ensure reliable operation and N-k ways are used for approximate storage where the voltage level can be tuned according to the QoS required. In this work, we have used a Hybrid-Memory array comprising of a combination of 8T and 6T SRAM cells for the N-k approximate ways so as to provide higher quality output. The accuracy gains have been shown in chapter 6.

A 4-way set associative L1-DCache has been used in this work and we have experimented with (1) **a3p1**: 3-Approximate-ways and 1-precise way (2) **a2p2**: 2-Approximate and 2-precise ways and demonstrated the energy trade-offs for these.

Image and Video applications are the target applications for this architecture and the major portion of the data that can be approximated is pixel data which are stored as 8bit blocks. These pixels have higher sensitivity to errors in the MSB bits as it can

potentially change the color value if there is a bit-flip in the MSB while the magnitude of error due to a bit-flip in the LSB is lesser. In order to address this issue we have used a combination of 8T-cells for MSB bits and 6T-cells for LSB bits. This is because 8T has a lower BER as compared to 6T (Figure 2.3) however it comes at the cost of increased area. 8T cells utilize 30% more area than a 6T cell. Hence we need to determine the optimal ratio of 8T:6T cells since increased area results in increased costs.

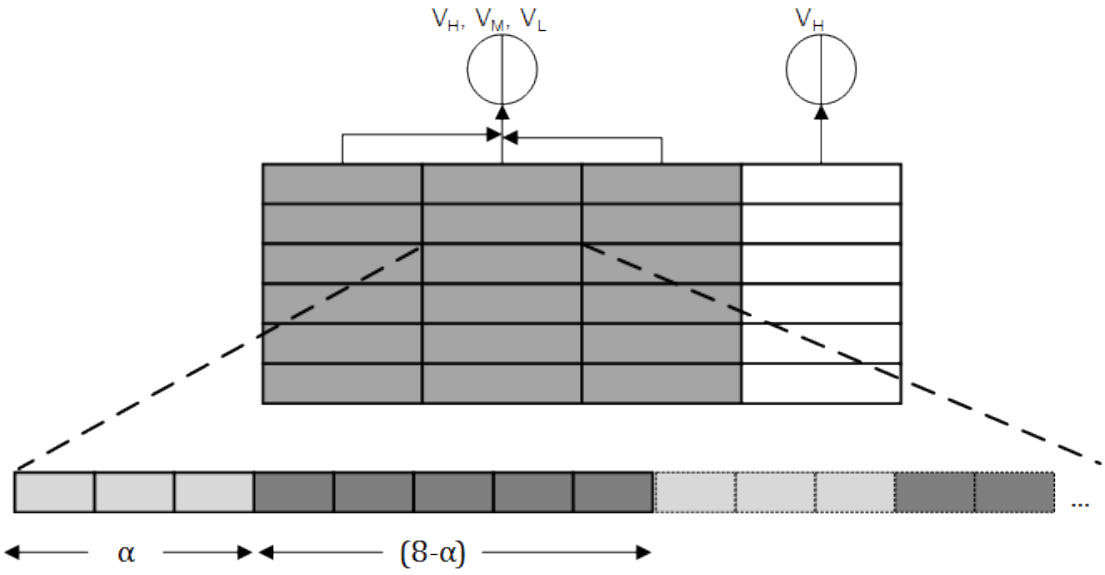


Figure 3.4: Hybrid-Memory layout

The Hybrid-Memory array is laid out as given in the Figure 3.4. As shown, each cache block in the tunable low voltage array is broken into chunks of 1Byte where α bits are 8T cells which are more reliable and $8-\alpha$ bits are 6T cells. This way we ensure a higher protection for MSB bit at the cost of increased area, however, it significantly improves the output quality.

The effective area(A_{eff}) for a N-way set associative cache where k-ways are used as precise, given that the 8T cells are β times larger than 6T cells is given by

$$A_{\text{eff}} = \frac{(\alpha\beta + (8-\alpha))(N-k) + 8k}{8N}$$

The area increase as compared to a cache which is made of only 6T cells for a a3p1

configuration with ($\alpha = 3$) 8T bits per Byte is 8.4% and for a a2p2 configuration it is 5.6%. In order to determine the optimal value of α we ran simulations for different configurations and determined that there is a point beyond which the gains in output accuracy diminishes as the area increases. The output accuracy metric used is Structural Similarity Index(SSIM)[22] where a SSIM of 1 indicates that the output is exactly the same as an error free output. The worst case SSIM is taken into consideration for each value of α and the $SSIM/A_{\text{eff}}$ is plotted as shown in the Figure 3.5 and it is found that $\alpha = 3$ is an optimal value. Thus for further analysis we consider a Hybrid-Memory with this optimal configuration.

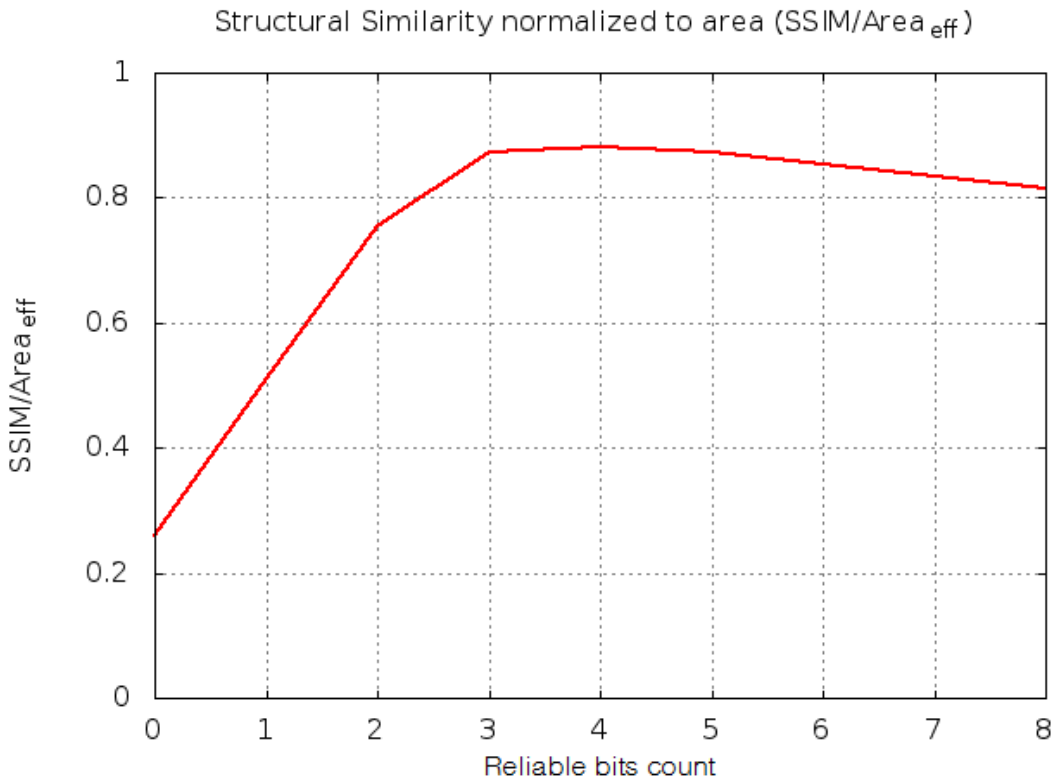


Figure 3.5: Determining optimal value for number of reliable bits α

In this architecture we include the Hybrid array only in the section that allows for tunable low voltage operation. This is because we do not need this extra area overhead in the High Voltage array as it ensures error free operation. This has helped reduce the

area costs for a same sized cache completely fabricated in its entirety as a hybrid array. Table 3.2 shows the area overhead for different configurations as compared to the base case of having a cache consisting only of 6T SRAM cells.

Configuration	Area Overhead
All 8T array, $\alpha = 8$	30%
Entire Array as Hybrid, $\alpha = 3$	11.25%
a3p1 Hybrid, $\alpha = 3$	8.4%
a2p2 Hybrid, $\alpha = 3$	5.6%

Table 3.2: Area overhead for different configurations

Chapter 4

Experimental Setup

In this chapter we discuss the experimental setup used in this work: the simulator modeling, the simulation methodology and the data extraction methods.

4.1 Simulator Modeling

For this work we have use the gem5 [23] and PCACTI [24] (Cache simulator based on CACTI6.5). We have also used the ITRS MASTAR tool to obtain voltage dependent device parameters to feed to PCACTI. The detailed description of source code changes and tool use has been explained in the Appendix B.

4.1.1 gem5

gem5 [23] is a cycle accurate simulator which is capable of full system simulation. It supports X86, ARM and ALPHA ISAs and it is most widely used in the field of Computer Architecture. This work focuses on the memory model section of the simulator. gem5 has two different memory models: *Classic* and *Ruby*. The *Classic* memory model has a simpler and faster memory system with basic reconfigurability however the *Ruby* model is more complex and flexible and is written in SLICC. We have used the *Classic* memory model for this work due to its simplicity.

Language Extensions

The programmer has to able to specify the regions of code that are to be declared as approximate. This in-turn must be mapped by gem5 onto the the VAT which keeps a track of the approximate data in the application. m5_DECLARE_APPROXIMATE (START_VA,END_VA) and m5_UNDECLARE_APPROXIMATE (START_VA,END_VA) as shown below in Figure 4.1 are used by the programmer to declare and undeclare the approximate data. gem5 Pseudo-Instructions have been used to implement the language extensions so as to map these into the op codes that write into the VAT which is accessed by the cache controller.

```
105         pic->img.plane[i] = x264_malloc( size );
106         if(i==0)
107         {
108         m5_DECLARE_APPROXIMATE((uint64_t)(pic->img.plane[i]), (uint64_t)( pic->img.plane[i] + size ));
109         }
```

(a)

```
136         m5_UNDECLARE_APPROXIMATE((uint64_t)(pic->img.plane[0]));
137     for( int i = 0; i < pic->img.planes; i++ )
138         x264_free( pic->img.plane[i] );
```

(b)

Figure 4.1: Approximate declarations in the program

Hybrid Cache architecture

The hybrid cache model has been incorporated in the *Classic* memory model of gem5. A voltage dependent bit error model (see 2.3) has been introduced. The faults in the cache are modeled having an uniform distribution with a probability given by the BER at the voltage. A Defect Map of the faults generated at each voltage are detected by the March Test during the BIST and automatically written into an on-chip memory accessible by the cache controller.

Only the L1-DCache is modified to contain the hybrid cache and the rest of the cache architecture is left intact. Whenever data written or is being read from a faulty cache block, the faults are injected into it based on a fault injection mechanism which mocks the behavior of an actual faulty cache.

The LRU cache replacement policy has been modified so as to map the data based

on criticality as we are dealing with unreliable sections of the memory. The cache controller is provided access to the Defect Map model and also the LRU counters so as to perform the selection of the cache block to evict. We have implemented both the *Split* and *Shared* policies discussed in 3.2.3. Table 4.1 shows the gem5 simulator parameters used for simulation.

Components	Parameters	Value
System	ISA	X86
	Type	Detailed (OoO)
	Core count	1
	Simulation mode	Syscall Emulation
	Cache Line Size	64B
L1-DCache	Voltage Levels	V_H, V_M, V_L
	Size, Assoc	32KB, 4-way
	Way Split	a3p1, a2p2
	Replacement Policy	Split LRU, Shared LRU
	8T Cells/Byte	3
L2 Cache	Size, Assoc	256KB, 8-way
Main Memory	Type, Size	LPDDR3, 512MB

Table 4.1: gem5 Parameter Setting

4.1.2 PCACTI

We have used PCACTI[25] to obtain the Leakage energy and Dynamic Access energy values for the L1-DCache and L2 cache. PCACTI has been used in this work as it is an extension to CACTI [24] that models the 8T SRAM Cell. CACTI is a widely accepted cache power and area simulator in the computer architecture field. It is not as accurate as HSpice but it does give a fair estimate.

PCACTI however provides statistics for the cache based on a fixed Vcc_{min}. But, as discussed previously we operate the cache at multiple voltage domains and some

are below the Vcc_{min}. It is important to note that several parameters change with the change in the operating voltage of the Cache. Some of the major parameters are V_t (threshold voltage), mobility, NMOS-PMOS on and off currents which effect the delay and power calculations.

We have used the MASTAR [26] tool from ITRS to obtain these values and modified the PCACTI device files so as to obtain a reliable estimate of the power and performance parameters as a result of voltage reduction. The values of power obtained from PCACTI is multiplied with the corresponding events in gem5 to get the overall power/energy consumption.

4.2 Data extraction method

The chart in Figure 4.2 describes the complete setup used for data extraction. Perl was the main scripting language used to automate the experiment runs and data extraction. We have used Matlab scripts [22] and MSU VQMT tool for extracting the SSIM values.

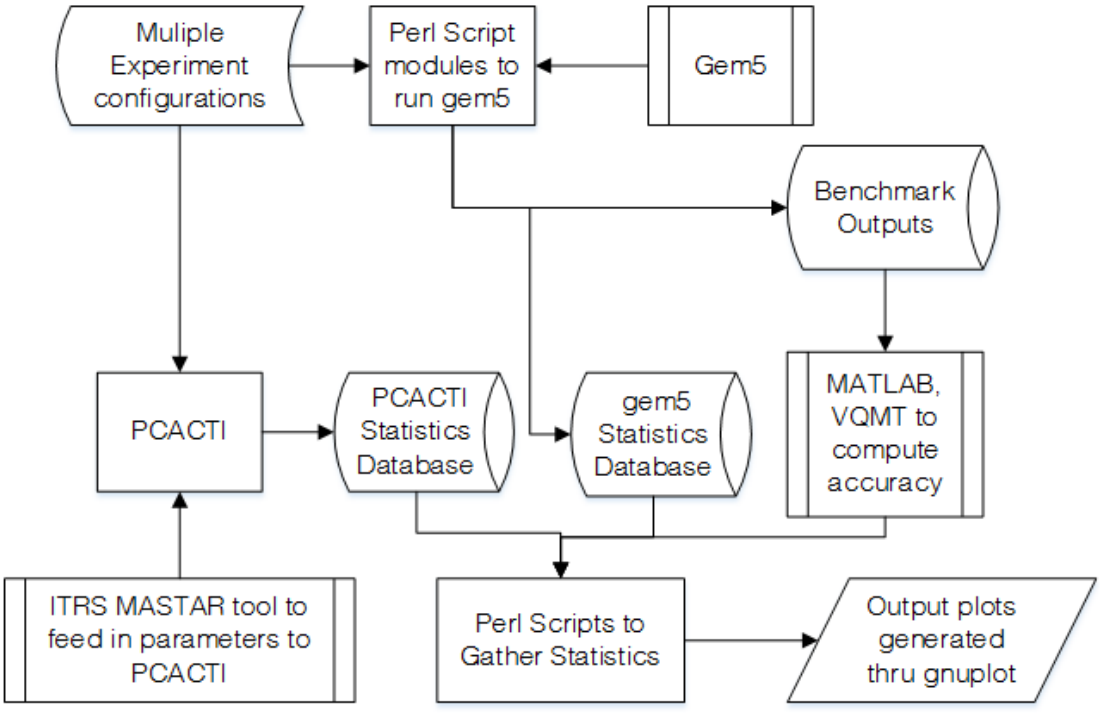


Figure 4.2: Data Extraction Flowchart

The cache architectures, replacement policies, number of reliable bits α , operating voltage, the benchmark to be run and the inputs for the benchmarks are toggled through a configuration file which is read by perl scripts that invoke different instances of PCACTI and gem5 to run these configurations.

PCACTI generates a output file which contains Power and Energy numbers for different events for each configuration that is run. In this work we are concerned with the L1 Cache leakage power and L2 Cache access energy at different voltages for the different configurations of the cache architecture discussed in chapter 3. gem5 generates a statistics file for each configuration and for each benchmark run. We consider the execution time as a performance metric and the events such as L2 tag accesses and L2 read and write hits to measure L2 Dynamic energy. These event numbers from gem5 are multiplied with values from PCACTI to get the total energy consumption.

The voltage dependent fault model that is introduced in gem5 injects faults in the L1 Cache and we keep track of the blocks that are faulty and not faulty to measure the cache capacity degradation.

Each invocation of gem5 for a specific configuration and benchmark generates an output. This is either a portable graymap image (PGM) or a H.264 encoded video. The image accuracy with respect to the *golden* output is obtained by running the SSIM implementation in Matlab. Similarly, for the video outputs, the SSIM values for each frame is computed using the MSU VQMT tool. All the above results are then consolidated using perl scripts to generate files read by gnuplot for plotting the graphs.

Chapter 5

Applications

This chapter talks about the applications that are investigated, their working and also the regions of the application that are suitable for approximation. We will also define a metric that we use to measure the accuracy of the output.

5.1 Application Set

Our primary target set of applications for this architecture are Image and Video applications. We have chosen the SUSAN Low Level Image Processing package [27] which includes *Edge Detection* and *Image Smoothing*. It is a part of the MiBench [28] benchmark suite that targets embedded processors which are expected to be low power devices and hence is a perfect target application suite for this analysis. For Video application, *x264* an open-source video encoder for the H.264 formats has been chosen.

To perform approximation, it is important to understand which regions of application are tolerant to errors. We have to ensure that any approximation to the code does not cause some irrecoverable errors and lead to change in the control flow of that application or cause segmentation faults. Hence, understanding the application working also becomes a vital part of approximate computing, highlighting the importance of programmer controlled approximation which can help localize approximation to regions which he deems appropriate. We will briefly describe these benchmarks and highlight the error tolerant regions of the application and their inherent immunity to errors.

5.1.1 Edge Detection

The SUSAN principle for image recognition uses masks with a center nucleus pixel. The brightness of each pixel in the mask is compared with the brightness of the mask's nucleus. The area of the mask which has the same brightness of the nucleus is defined as the "USAN"(Univalue Segment Assimilating Nucleus).

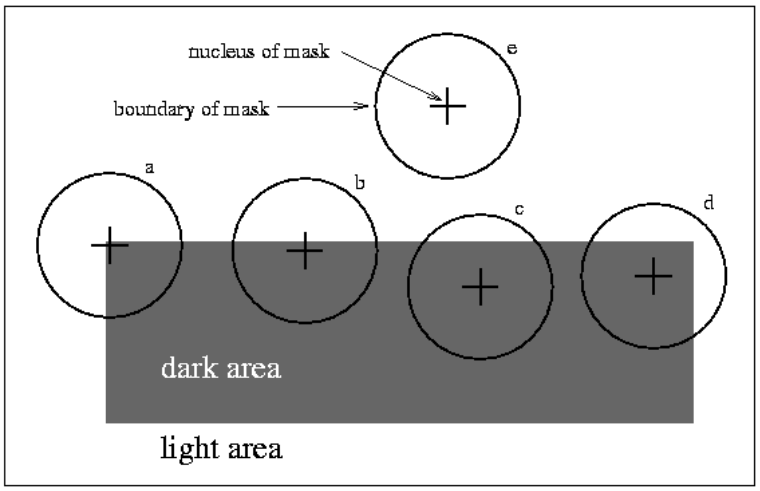


Figure 5.1: Shows circular masks at different places of an image with an edge

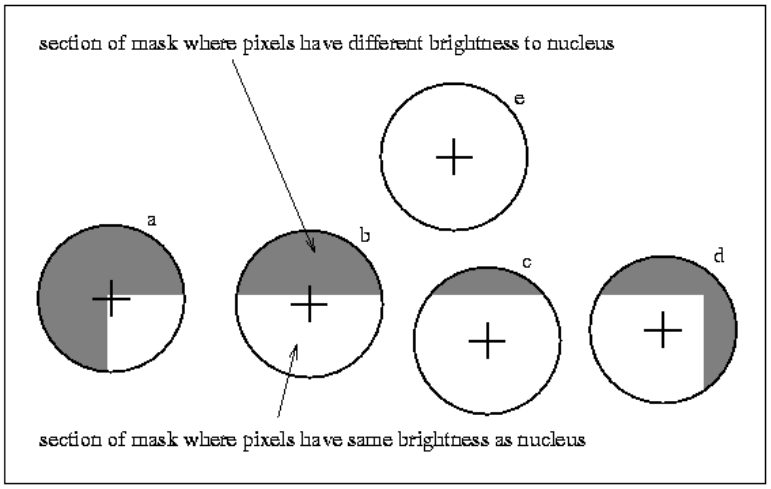


Figure 5.2: Shows the masks with USAN in the white parts of the mask

As it can be seen the USAN area (white region in the circular mask in Figure 5.2) is highest when the mask lies flat on the surface and is lower at the edges and even lower at corners. The edges are detected based on this principle. The application is shown to be resilient to Gaussian noise as the USAN area does not change much with the change in the pixel value of a few pixels in the mask. We have thus allowed for approximate storage of the source image pixels which are gathered for this computation.

5.1.2 Image Smoothing

The SUSAN image smoothing is a noise filtering algorithm which preserves the image structure by smoothing over the neighbors which form a part of the same region as the central pixel of the mask. It averages over all of the pixels in the locality which lie in the USAN. It is a similar technique as the edge detection and hence we have employed the same technique of storing the source image pixels in unreliable storage. Any errors in the source, as a result, be averaged out by the filtering technique. Thus, this benchmark has shown high tolerance to random errors in the source image and still provides a good result.

5.1.3 x264 Video Encoder

x264 is an open-source encoding tool that is based on the H.264 Advanced Video Coding (AVC) standard [29]. H.264, also known as MPEG-4 Part 10/AVC, is the latest MPEG standard for video encoding which allows for low bandwidth high quality transmission. It can compress a video file to less than 80% of the size compared to a Motion-JPEG compression and more than 50% as compared to MPEG4-Part2 thus utilizes low network bandwidth and storage space or give a very high quality output for the same bitrate/size. Thus several video streaming sites like YouTube has adopted this standard. It has become a standard format for mobile devices and network cameras.

The H.264 standard uses several techniques to reduce the file size, one of which is to prevent transmission of static elements and exploits both spatial and temporal redundancy. Video encoding is done using done using frames and this uses 3 basic frames called I,P and B frames. The I-Frames are basic frames and is encoded without reference to other frames. The P-Frames are predicted from I-Frames and the B-Frames

or Bi-predictive frames reference both I and P Frames. Figure 5.3 illustrates the three frames and how they relate to one-another. The motion estimation between frames are done using macro-blocks which are blocks of non-overlapping pixels [30].

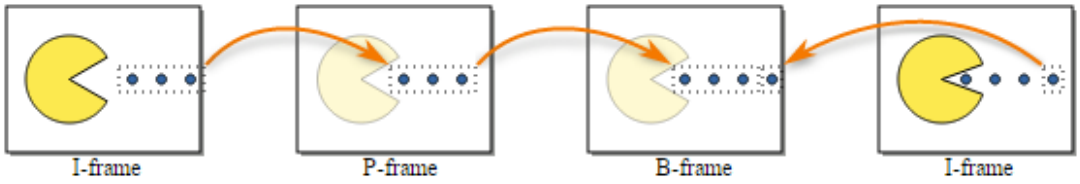


Figure 5.3: I, P and B Frames of a sequence of images (Img src: Wikipedia)

Each pixel derives its value from YUV values which represented with one "luminance" component called Y (equivalent to grey scale) and two "chrominance" components, called U (blue projection) and V (red projection) respectively. In most common formats, the Y plane components of a pixel are dominant and comprise of 66.67% of the pixel value. Since these Luma pixels are the most used, it is beneficial if this is stored in a low voltage array. These "Luma pixels" are 8bits in length and it represents a gray scale image. The following Figure 5.4 shows a 4:2:0 sub-sampling scheme and how chroma pixels are shared among pixels in different rows.

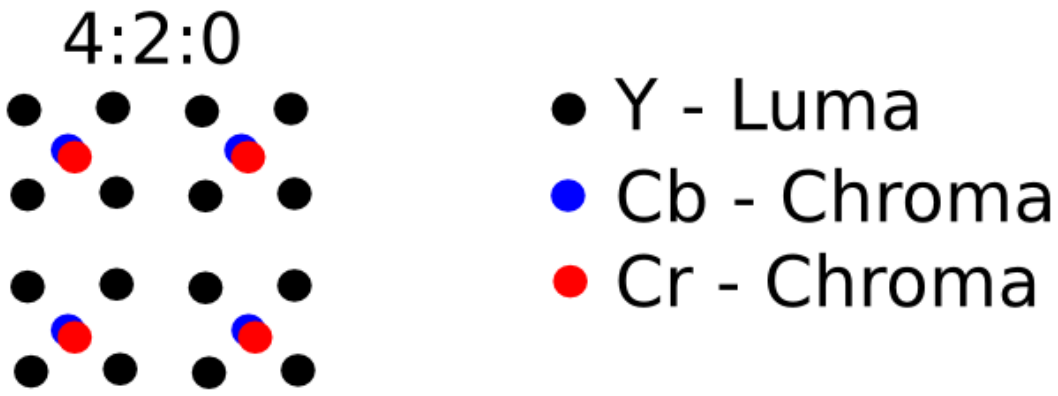


Figure 5.4: YUV 4:2:0 pixel format (Img src: stackoverflow)

5.2 Accuracy Metrics

Working at lower voltages and unreliable memories result in lower quality outputs. Here we define a metric to measure the quality degradation as compared to a output obtained by using reliable hardware running at higher voltages. The goal of a fidelity metric here is to compare two images/video frames to the error free version which shall be referred to as the *golden* output. MSE (Mean Squared Error) and PSNR has been quite dominant as a fidelity metric in image processing. Given a noise-free mxn monochrome image I and its noisy approximation K, MSE is defined as:

$$MSE = \frac{1}{m\,n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2$$

The PSNR is a function of the MSE and is defined (in dB) as:

$$PSNR = 20 \cdot \log_{10}(MAX_I) - 10 \cdot \log_{10}(MSE)$$

Here, MAX_I is the maximum possible pixel value of the image. When the pixels are represented using 8 bits per sample, its value is 255. Here we see that MSE essentially comprises of differences between pixels of an image. However, it does not consider temporal or spatial relationships between signals. This means that if the pixels in the image is re-ordered in the same way MSE will be the same. Also, it goes to show that all pixels of the image are equally important. However, visual perception of quality is very different. The human visual system is highly adapted to process structural information. Structural information is the idea that pixels have a strong inter-dependencies when they are spatially close. Thus, a fidelity metric called SSIM (**S**tructural **SIM**ilarity) [22] has been chosen which measures structural distortion.

The SSIM algorithm typically breaks down images into patches of 8x8 pixels and measures similarities of three elements of the image patches: the similarity *l(x, y)* of the local patch luminance (brightness values), the similarity *c(x, y)* of the local patch contrasts, and the similarity *s(x, y)* of the local patch structures which is the inner-product of the correlation between two images. SSIM is computed as (generally $\alpha = \beta = \gamma = 1$):

$$SSIM(x,y) = [l(x,y)]^{\alpha} \cdot [c(x,y)]^{\beta} \cdot [s(x,y)]^{\gamma}$$

The change in the structural information between a reference (In our case, the *golden* output without distortions) and the distorted image is related on a scale of -1 to 1. A value of SSIM=1 means that the two images are perfectly identical. In this work the noise introduced is mostly random noise and PSNR sometimes fails to catch the degradation of image quality. The following Figure 5.5 shows how SSIM is a better fidelity metric than MSE/PSNR. The MSE is the same even if the entire image is made brighter, has increased contrast or injected with a noise etc.

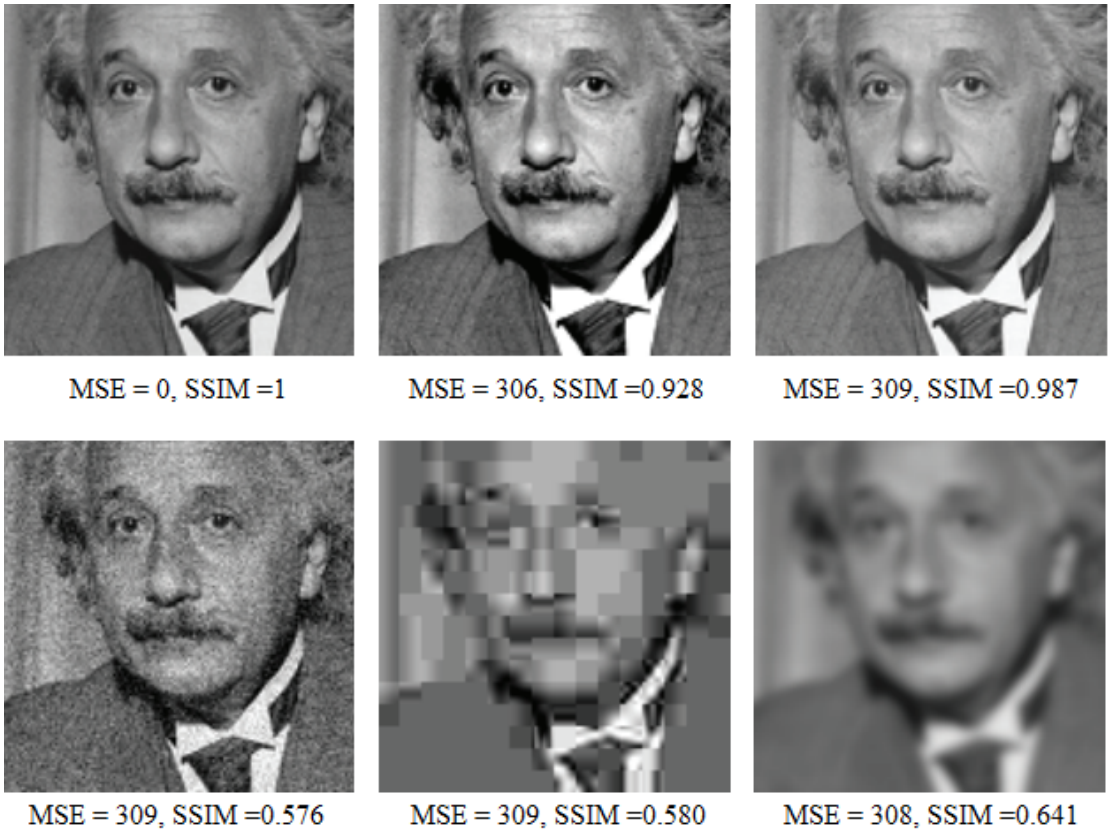


Figure 5.5: Images having nearly same MSE but entirely different perceptual quality[1]

Thus, considering the factors mentioned above, SSIM was chosen as the fidelity metric over MSE/PSNR.

Chapter 6

Analysis

This chapter presents the results and analysis for different architectures proposed in chapter 3. We discuss the energy savings, the output accuracy degradation due to voltage lowering and the best configuration for the set of benchmarks considered. The Table 6.1 describes the benchmarks and the corresponding inputs used in this work.

Benchmark	Input	Format	Size
Edge Detection	Lena test image	pgm	512x512 Grayscale
Image Smoothing	Lena test image	pgm	512x512 Grayscale
x264 encoder	Akiyo	yuv	30 frames, CIF 352x288

Table 6.1: Benchmarks and inputs

6.1 Energy Analysis

This section discusses the Energy analysis for the three different benchmarks. We have considered the L1 Leakage energy and the L2 Dynamic Access energy for analyzing the energy efficiency of the cache architecture. The L2 accesses are done upon misses in the L1, and L2 misses result in a request for the block from main memory. The following graphs show the dependence of Energy consumed on different cache architectures and different cache replacement policies.

Edge detection is an interesting benchmark with regards to approximation because it does show resilience to errors and performs well at low voltage using unreliable hardware. The bars for leakage energy (Figure 6.1) in the L1-DCache drop with the lowering of voltage however the L2 Accesses increase. This is because of a reduced number of *non-faulty* cache blocks. We notice that the *Split* and *Shared* cache replacement policies have different L2 Access Energy consumption. The shared case performs better because it makes use of the *non-faulty* blocks in the approximate ways of the cache. Thus resulting in fewer misses. This is especially visible at 560mV. The Split-LRU case effectively treats the approximate and precise ways of the cache bank to be separate. However, the Shared-LRU treats it as a unified cache.

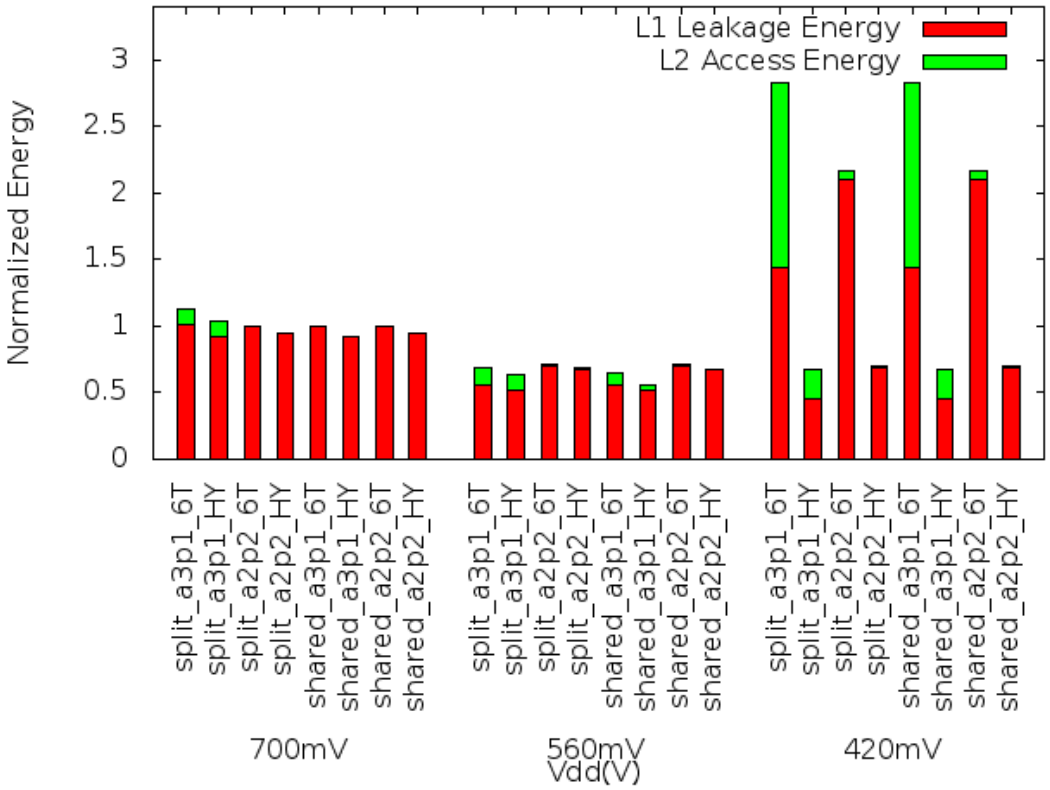


Figure 6.1: Energy distribution for Edge detection application

The following Figure 6.2 shows the cache capacity degradation. The Shared-LRU scheme helps the precise data to be mapped into the non-faulty (grey bars in the graph)

regions of the approximate bank, effectively giving the visualization of a larger cache. These grey regions can be used interchangeably by precise and approximate data.

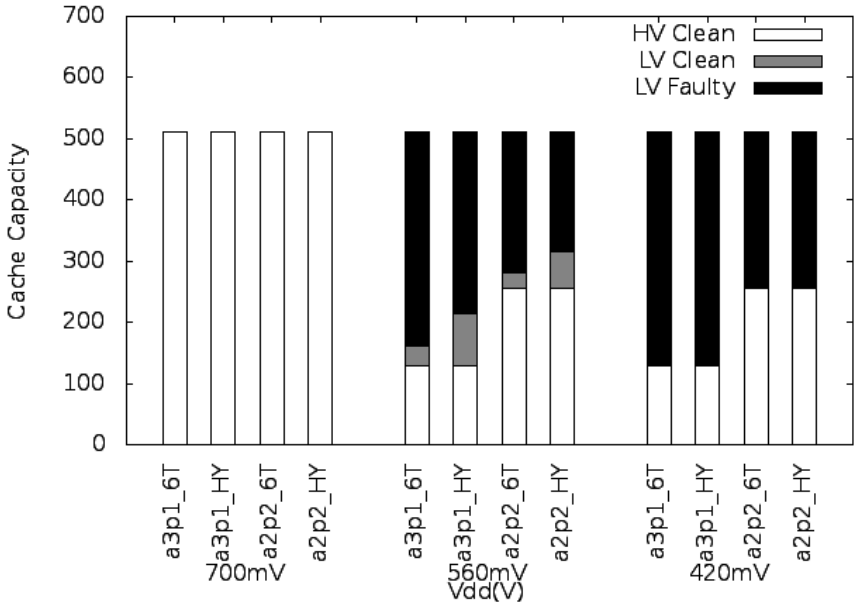


Figure 6.2: Cache capacity reduction with drop in voltage

The a3p1 and a2p2 have same energy consumption at the baseline 700mV but at lower voltages, the leakage energy of the a3p1 configuration is much lower. This is because, effectively, only one cache line is at V_H and the rest operate at lower voltages. But this causes the number of misses to increase for the precise data in the program since the precise data effectively sees only a direct mapped cache at low voltages, causing L2 Access Energy to go up.

The most noticeable result is the V_L (420mV) case where the energy bars for the cache architecture using only 6T SRAM cells are much higher than the base case. This is because the execution time has increased by 3.5x and it can be attributed to the high number of faults which in-turn causes a lot of false edges to be generated and detecting each edge adds to the run-time. It is interesting to note that the Hybrid cache architecture outperforms the purely 6T SRAM array and goes to show that this image processing application is indeed sensitive to position of error.

Image smoothing shows a higher resilience to error as compared to edge detection

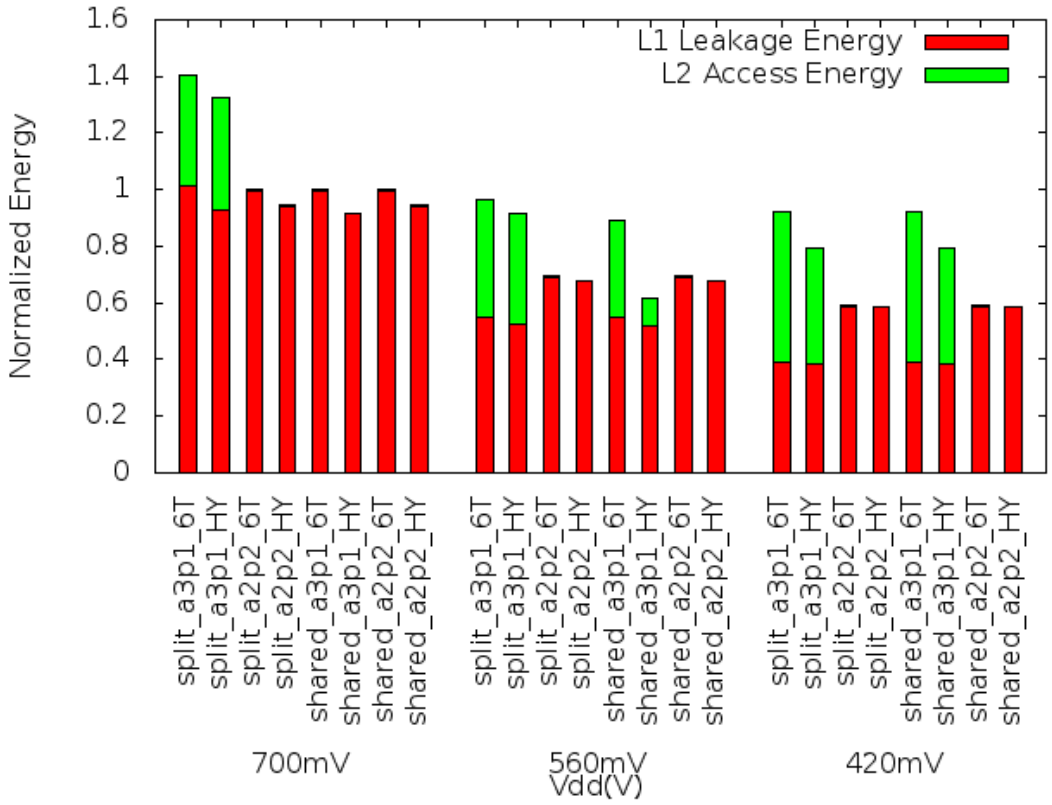


Figure 6.3: Energy distribution for Image smoothing application

as the application run-time remains nearly unaffected. This can be attributed to the nature of the application which is in-effect a noise filtering algorithm which smoothens/averages the pixels over the USAN area. In the Figure 6.3 we can see that the leakage energy at 420mV in both the Hybrid and 6T case for a2p2 configuration is nearly 41% lesser than the base-case at 700mV and a3p1 is 61% lesser. However the a3p1 configuration has increased L2 Accesses because of the cache capacity for the precise data reducing to 1/4 of the cache size. The total energy reduction for the a3p1 configuration at 420mV including the L2 Access energy is just 7.3%. We can see that from these two image processing applications, the performance of the a2p2 configurations in regards to total energy is much better at lower voltages than the a3p1 configuration.

x264 video encoding application shows a large dependence on input source quality in order to perform encoding. It can be seen in the Figure 6.4.

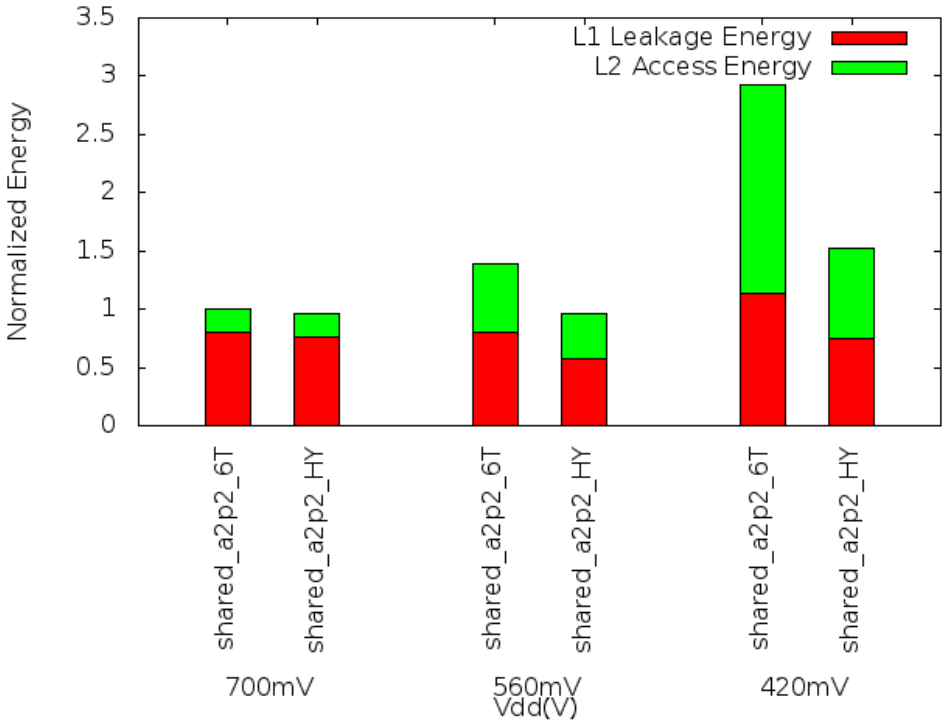


Figure 6.4: Energy distribution for x264 Video encoding application

The Voltage reduction has resulted in increased execution times. This is because x264 video encoding is done based on the differences between frames. As discussed before in chapter 5, the encoding is done using I,P and B Frames. Since we are fetching the frame data approximately, the differences between two frames start to increase if there is a lot of noise introduced due to faults. Hence the encoder thinks that each frame as a result is entirely different from the other. This results in multiple I-Frames as compared to P and B frames which are larger in size because it contains the entire image data for that frame. P and B frames contain only the non-static information between frames. Writing these I-Frames into the encoded video file takes longer, which results in increased execution times.

However, it is interesting to note that the Hybrid memory performs much better than the purely 6T case because the amount of error introduced in the MSBs is much smaller. Hence, the built in noise-reduction capabilities of x264 is capable of eliminating

the noise caused due to faults in the LSBs. This has resulted in reduced I-Frame count and better quality output which is described in the following section.

6.2 Accuracy Analysis

In this section we explore the accuracy trade-offs due to low voltage operation and working with unreliable memories.

Edge detection benchmark output accuracy depends closely on the level of approximation. As seen in the Figure 6.5 below the output image quality drops steeply at low voltages.

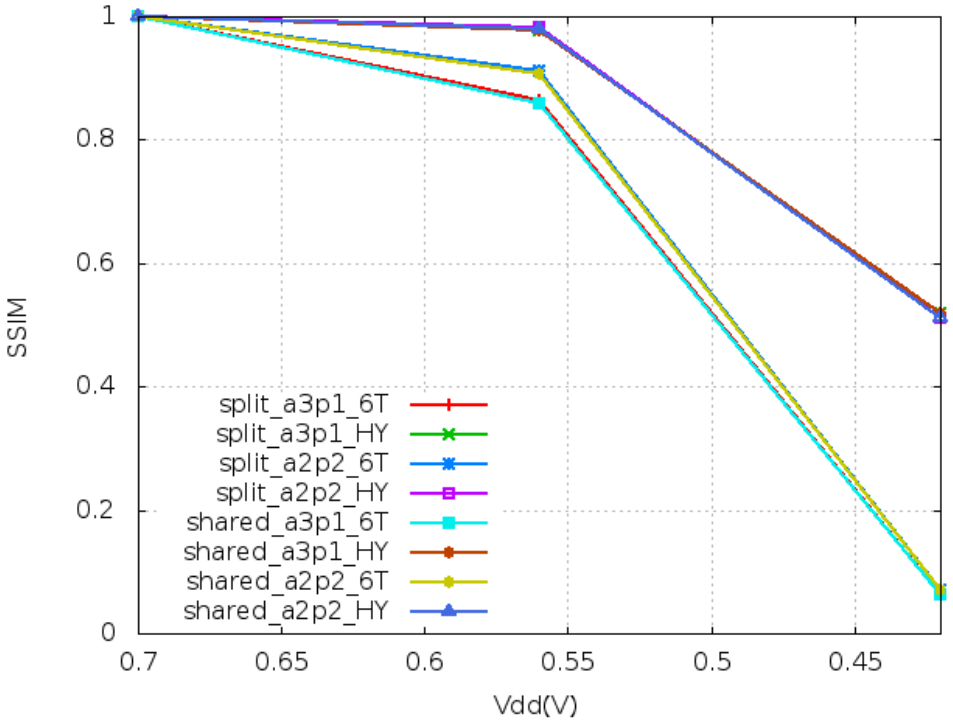
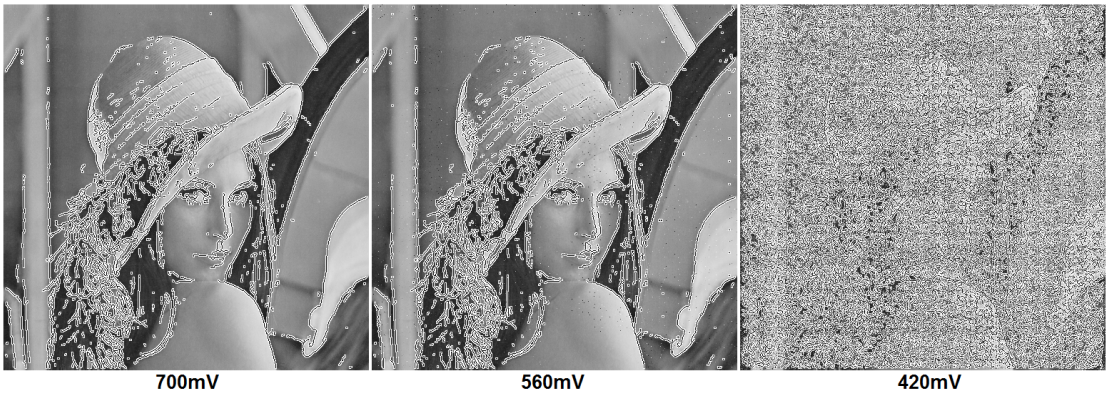


Figure 6.5: Edge Detection accuracy results for all configurations

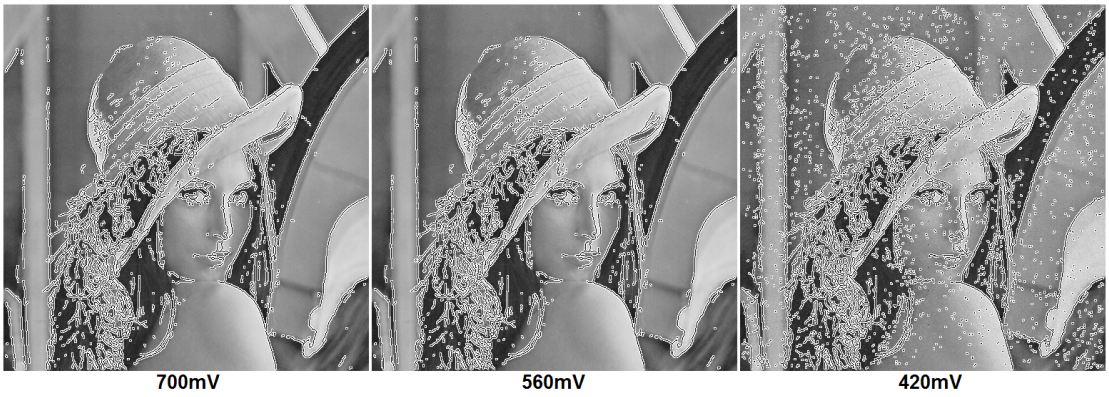
At lower voltages the cache starts to become more and more faulty and hence creates more faulty pixel data. This results in artificial edges being created and the number of these artificial edges rises exponentially. Since the purely 6T cache has a uniform BER throughout, the MSB fault can result in a complete change in the pixel color, say

from a white pixel to a black pixel or vice versa. However the Hybrid cache enforces protection on the MSB bits. Thus the number of false edges introduced in the Hybrid cache is much lesser.

Figure 6.6 compares the output quality degradation of a purely 6T cache and a Hybrid cache. Which shows that low voltage operation is possible with a Hybrid cache whereas at 420mV the 6T has indiscernible outputs. It has also been shown that the potential energy savings in the 6T case is lost at very low voltages due to the increased execution time which in-turn causes an increase in the total energy consumption.



(a)



(b)

Figure 6.6: Output image quality comparison for Edge Detection in (a) Only 6T cache (b) Hybrid cache, at 700mV, 560mV and 420mV

The SSIM values at 420mV for Hybrid cache is 0.512 whereas for the 6T cache is

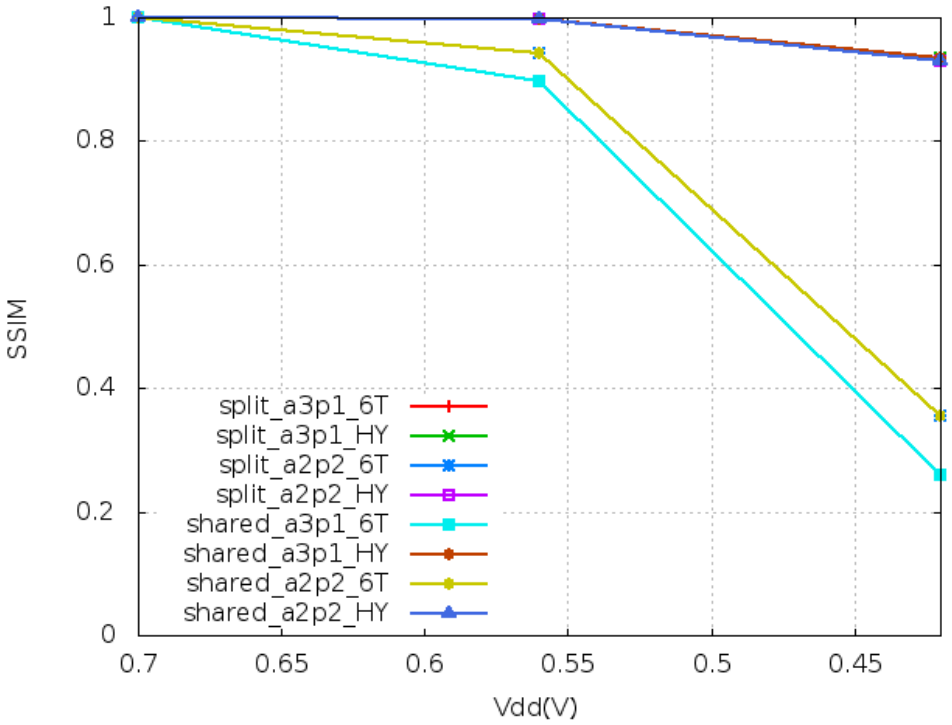


Figure 6.7: Image smoothing accuracy results for all configurations

0.062. This shows that we cannot aggressively scale voltage for this benchmark and a threshold of 560mV should be set for this application to obtain acceptable results. Even at 560mV the Hybrid has about 8% better quality output.

Image Smoothing has shown the most error resilience among the benchmarks chosen in this work. The inherent noise filtering function of the image smoothing has resulted in higher quality outputs. However the errors in the MSB cause significant reduction in the output quality. The SSIM values for the purely 6T cache at 420mV drops as low as 0.258 whereas the Hybrid cache maintains a very high quality image with a SSIM of 0.935. This is because image smoothing by itself is some sort of mechanism to remove variations in the LSB bits of a pixel in a particular region of the image by applying filters.

The output image quality comparison of the same test image is shown in the Figure 6.8 where it can clearly be seen that the Hybrid cache has the same or better output

(a)

(b)

Figure 6.8: Output image quality comparison for Image Smoothing in (a) Only 6T cache (b) Hybrid cache, at 700mV, 560mV and 420mV

quality at 420mV than a purely 6T cache at 560mV. This shows a potential voltage reduction of 140mV without compromise on quality. The purely 6T cache suffers from high noise levels causing the smoothing algorithm to fail because of very minimal number of pixels under the USAN area to perform the smoothing.

x264 Video Encoder provides an interesting take on the approximation. In this case the encoder reads in a raw video file and performs video encoding based on the latest H.264 video encoding standard. Here the input raw frames are read in and computation for encoding is performed on these frames which are stored approximately at low voltages. Here we are dealing with multiple output frames being encoded, so

we present the SSIM values (Figure 6.9) for each output frame as read by a decoder and compare it to the baseline encoded video file at 700mV. These values are obtained using the MSU Video Quality Measurement Tool [31] which compares a video file with respect to a reference.

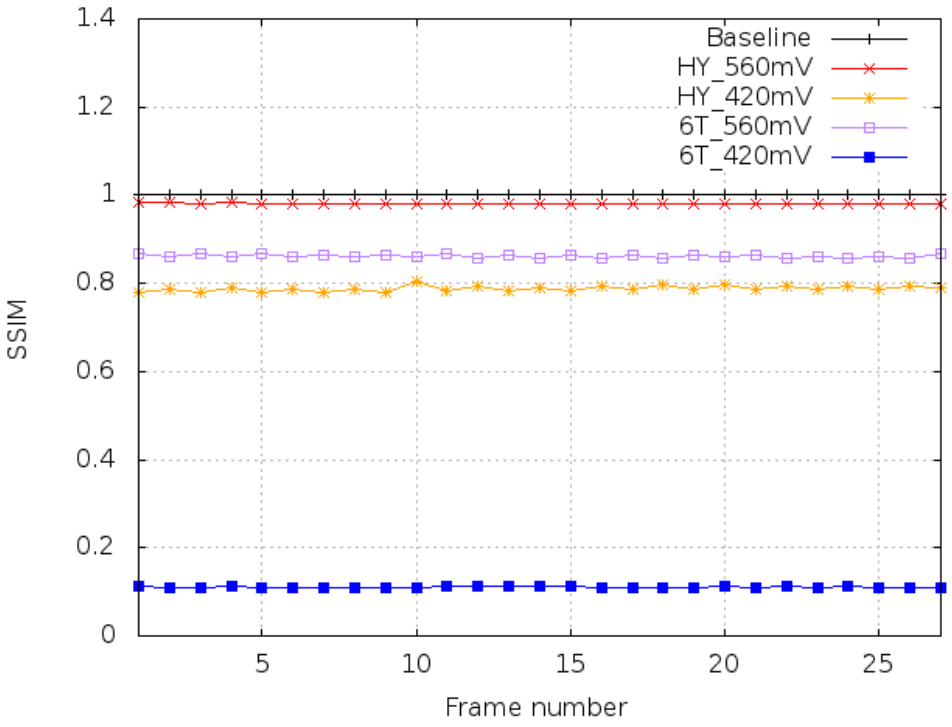


Figure 6.9: x264 accuracy results for each frame for different voltage levels

We have grabbed the same frame from all encoded video files for fair comparison and it can be seen in the Figure 6.10. The purely 6T cache has very low quality output at 420mV, however, the Hybrid cache has nearly the same output quality at 420mV as the 6T case at 560mV. But, its important to note that in this benchmark, we do not obtain any energy savings by lowering voltage due to the increase in the execution time(see 6.1). Thus, this work differentiates from other recent work by correlating energy and accuracy as interdependent variables for designing an error tolerant cache rather than separate components. This also stresses the tolerance threshold of applications to errors in the inputs. The x264 Video Encoder thus may not be a suitable application

for approximation considering the energy and accuracy trade-offs. Future work can explore the decoder section instead of the encoder as it's objective is only the rendering of the frames based on the encoded information and we presume that approximating this would not result in increased overheads.

(a)

(b)

Figure 6.10: Output image quality comparison for x264 Video Encoder in (a) Only 6T cache (b) Hybrid cache, at 700mV, 560mV and 420mV

6.3 Performance Analysis

In this section we explore the degradation in performance for the different benchmarks due to the reduced voltages on different cache architectures. We consider the normalized execution time as the metric to determine the performance of the application in low voltage conditions. The Figure 6.11 gives a summary of normalized execution time for the three benchmarks considered in this work.

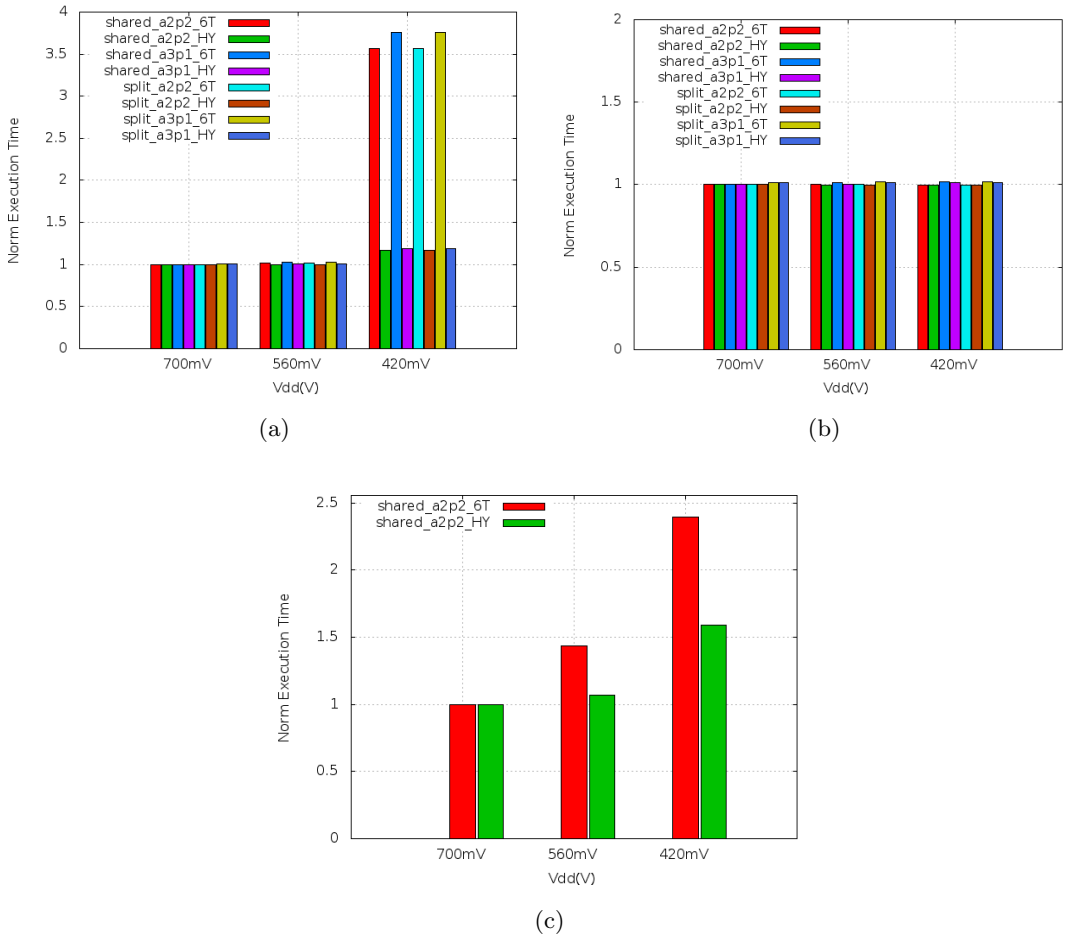


Figure 6.11: Normalized Execution time for (a) Edge Detection (b) Image Smoothing (c) x264 Video Encoder

Edge Detection has been shown to be capable of tolerating moderate levels of error. It has also been shown in 6.1 that the energy consumption for a purely 6T cache at 420mV is higher than the baseline case which is contrary to what we want to achieve. The execution time for this case goes up by 3.76x in the a3p1 configuration and 3.56x in the a2p2 configuration, whilst the Hybrid cache has an execution time of 1.18x in the a3p1 configuration and 1.16x in the a2p2 configuration.

Image Smoothing is nearly unaffected by the faults in the memory and this is a perfect application for low voltage operation. The execution time goes down on an

average by 1%-2% even while running at 420mV for all cache configurations.

The **x264** Video Encoder application is sensitive to errors and it can be seen that the purely 6T cache suffers at low voltage operation. There is a near exponential increase in the execution time from 1.43x at 560mV to 2.39x at 460mV. However, for the Hybrid cache the execution time is increases by 1.07x at 560mV and 1.59x at 420mV.

Execution time is not the only parameter to consider for a video encoder. Since the primary job of an encoder is to compress the raw video files and make it suitable for storage/transmission over the network we must consider the compression factor also as a metric to determine if this is suitable for approximation. Higher compression ratios result in smaller storage space and network bandwidth for communication. The following Figure 6.12 shows the increase in file-size due to low voltage operation as a result of higher number of static I-frames which are larger in size. We see here that low voltage operation is not always beneficial and we have to consider the behavior of the application to errors.

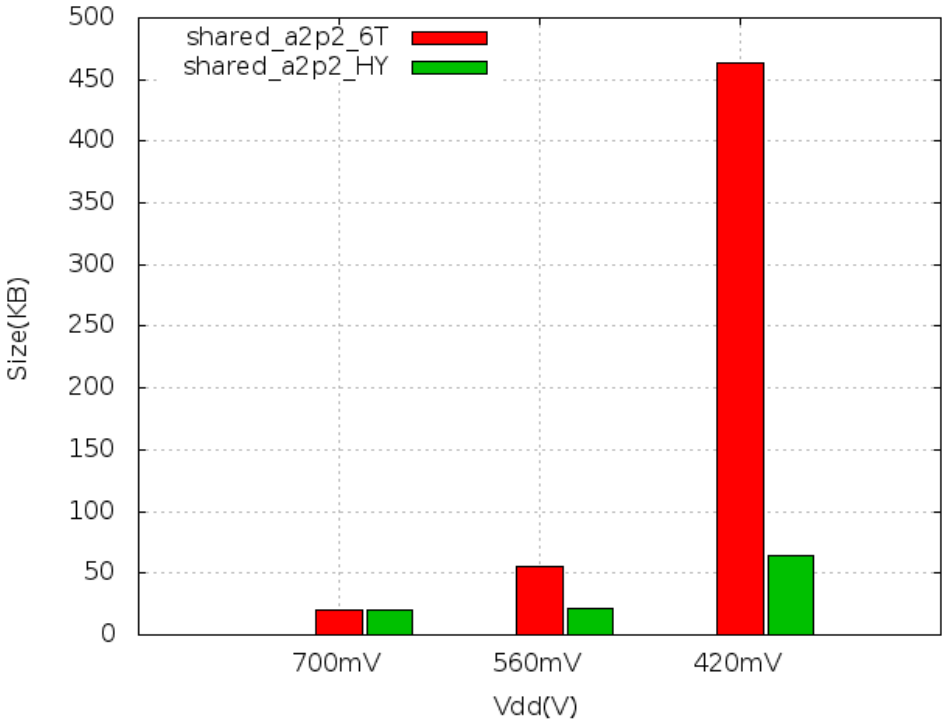


Figure 6.12: x264 encoded video file size comparison

Area overhead is also another trade-off factor with respect to fabrication costs and these overheads are listed in Table 3.2.

6.4 Related Work

Recent research in the field of approximate computing talk about software techniques [15][13][16] and hardware architectures [19][32][14][17][18] to allow for low voltage operation of applications. They also highlight the energy savings mainly with regards to leakage and accuracy degradation due to approximation. However it is important to understand that the application may not behave the same in an approximated environment. This work explores different cache architectures combined with the techniques adopted in [18][19] and [13] and evaluate its behavior during low voltage operation (below voltage margins). We have implemented the proposed set of cache architectures on the L1-DCache and not only explored the result of energy reduction in terms of Leakage due to voltage lowering but also the L2 Access energy which is also an important factor that has been neglected in past works.

This work also shows that merely increasing the error tolerance threshold of the outputs does not mean increase in energy savings because applications tend to be dependent on the quality of the inputs to a certain extent. Certain applications are highly error tolerant, some moderately and some highly sensitive. We have shown all three varieties and their behavior and also stress the need for a such a classification [6]. Thus relating all the parameters such as Energy, Accuracy , Performance and Cost (Area) becomes a necessity for low voltage approximate computing.

Chapter 7

Conclusion and Discussion

CMOS technology scaling over the years have resulted in increased device variability. SRAM is one of the most sensitive device to hardware variations. Design Engineers thus impose several guard-banding mechanisms to enforce fault free operation. Voltage margin is one of the guard-banding mechanism that prevents low voltage operation and thus resulting in high power consumption. A large set of applications, however, are tolerant to error and do not require these guard-banding techniques. The shift to mobile and low power devices has sparked the need to explore such trade-offs.

The focus of this work is to show techniques to improve the existing methods to reduce energy consumption. We explore image and video applications as they would be the best fits for the Cache architectures proposed for low voltage operation and how Hybrid memories help improve accuracy and performance. The detailed analysis of the trade-offs in consideration are shown and we have also made a note of the application error tolerance. We also show that error tolerance is best decided by the application programmer and hence have provided the knobs to the programmer to tune for best trade-off.

We have first discussed the architectural support to handle the programmer declarations for approximation of data and then the different cache configurations with cache replacement policies. Among the cache configurations explored, we have seen that a 4-way L1D-Cache implemented with a combination of 8T and 6T cells having two tunable approximate ways and two high voltage precise ways (a2p2_HY) have the best results in terms of Total Energy, Accuracy, Performance and Area. We have also shown how Split

and Shared replacement policies perform and how we can utilize non-faulty blocks in approximate ways to get performance and energy gains with the caveat that the BIST is capable of catching all faults at boot-up.

The total area overhead for this cache configuration as compared to the baseline is 5.8%. In the case of Edge Detection we have shown that this configuration would help reduce energy by 31% while maintaining an output with a SSIM of 0.9803 and a performance reduction as low as 1%. For Image Smoothing the energy reduction is as high as 41% for a output with a SSIM of 0.935. x264 Video Encoder has been shown to be sensitive to approximation with negligible gains in-terms of energy. However, it provides an interesting take on application error tolerance which shows how cumulative error adversely effects performance and counteracts the energy savings obtained due to low voltage operation.

7.1 Future Work

The important take away from this work is finding an optimum trade-off space among several variables. Thus developing a system that helps provide the best architecture while maximizing for a particular variable would be beneficial. For example, automatic tuning of the number of the approximate ways based on the miss-rates and outputs. Another approach would be to attack the problem at the application level and take into consideration application phases while applying the voltage tuning to reap maximum benefits.

In this work we have used a simple single core architecture and it would be interesting to see the scaling benefits if incorporated in a multi-core architecture while keeping in mind that the cache would have to be invalidated during a context switch, to prevent the new program's data from accessing the old program's data as we are using a virtually addressed cache. We could also explore similar architectures for higher level caches and the main memory while making sure critical information and data is not effected and then determine the energy savings. However, in-order to achieve this, the higher levels of cache should have a different VA Table which would specify the main memory pages that are stored approximately and fills into the cache from these locations would be stored in approximate ways of the cache. Also in the case of evicting a page from the

main memory we must make sure that only approximable data is filled in the unreliable regions of memory otherwise it could lead to catastrophic failures of the application.

GPUs could potentially benefit from this model for approximation as the applications that are generally handled by the GPU are image or video applications. The shared memory in the GPU could be sectioned into reliable and unreliable memory. An aging model could also be added to simulate the transistor aging to automatically update the defect map which allows to keep track of faulty blocks over time.

References

[1] Zhou Wang and Alan C. Bovik. Mean squared error: Love it or leave it? a new look at signal fidelity measures. *Signal Processing Magazine, IEEE*, 26(1):98–117, January 2009.

[2] International Solid-State Circuits Conference. *ISSCC 2014 Tech Trends*.

[3] Fabio D'Agostino and Daniele Quercia. Short-Channel effects in MOSFETs. December 2000.

[4] Jan M. Rabaey, Anantha Chandrakasan, and Borivoje Nikolic. *Digital Integrated Circuits (2nd Edition)*. Prentice Hall electronics and VLSI series. Prentice Hall, 2 edition, January 2003.

[5] The international technology roadmap for semiconductors. Technical report, ITRS, 2012.

[6] Vinay K. Chippa, Srimat T. Chakradhar, Kaushik Roy, and Anand Raghunathan. Analysis and characterization of inherent application resilience for approximate computing. In *Design Automation Conference (DAC), 2013 50th ACM / EDAC / IEEE*, pages 1–9. IEEE, May 2013.

[7] Leland Chang, Robert K. Montoye, Yutaka Nakamura, Kevin Batson, Richard J. Eickemeyer, Robert H. Dennard, Wilfried Haensch, and Damir Jamsek. An 8T-SRAM for variability tolerance and Low-Voltage operation in High-Performance caches. *Solid-State Circuits, IEEE Journal of*, 43(4):956–963, April 2008.

[8] Guanghui Liu. ECC-cache: A novel low power scheme to protect Large-Capacity l2 caches from transiant faults. In *Information Assurance and Security, 2009. IAS*

'09. Fifth International Conference on, volume 2, pages 193–199. IEEE, August 2009.

[9] Albert Meixner, Michael E. Bauer, and Daniel J. Sorin. Argus: Low-Cost, comprehensive error detection in simple cores. In *Microarchitecture, 2007. MICRO 2007. 40th Annual IEEE/ACM International Symposium on*, pages 210–222. IEEE, December 2007.

[10] Man L. Li, Pradeep Ramachandran, Swarup K. Sahoo, Sarita V. Adve, Vikram S. Adve, and Yuanyuan Zhou. Understanding the propagation of hard errors to software and implications for resilient system design. *SIGOPS Oper. Syst. Rev.*, 42(2):265–276, March 2008.

[11] Christoph Borchert, Horst Schirmeier, and Olaf Spinczyk. Generative software-based memory error detection and correction for operating system data structures. In *Dependable Systems and Networks (DSN), 2013 43rd Annual IEEE/IFIP International Conference on*, pages 1–12. IEEE, June 2013.

[12] Milos Prvulovic, Zheng Zhang, and Josep Torrellas. ReVive: cost-effective architectural support for rollback recovery in shared-memory multiprocessors. In *Computer Architecture, 2002. Proceedings. 29th Annual International Symposium on*, pages 111–122. IEEE, 2002.

[13] Adrian Sampson, Werner Dietl, Emily Fortuna, Danushen Gnanapragasam, Luis Ceze, and Dan Grossman. EnerJ: Approximate data types for safe and general low-power computation. In *Proceedings of the 32Nd ACM SIGPLAN Conference on Programming Language Design and Implementation*, volume 46 of *PLDI '11*, pages 164–174, New York, NY, USA, June 2011. ACM.

[14] Song Liu, Karthik Pattabiraman, Thomas Moscibroda, and Benjamin G. Zorn. Flikker: Saving DRAM refresh-power through critical data partitioning. *SIGPLAN Not.*, 46(3):213–224, March 2011.

[15] Marc de Kruijf, Shuou Nomura, and Karthikeyan Sankaralingam. Relax: An architectural framework for software recovery of hardware faults. In *Proceedings of the*

37th Annual International Symposium on Computer Architecture, ISCA '10, pages 497–508, New York, NY, USA, 2010. ACM.

[16] Woongki Baek and Trishul M. Chilimbi. Green: A framework for supporting energy-conscious programming using controlled approximation. In *Proceedings of the 2010 ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '10, pages 198–209, New York, NY, USA, 2010. ACM.

[17] Chris Wilkerson, Honglliiang Gao, Alaa R. Alameldeen, Zeshan Chishti, Muhammad M. Khellah, and Shiih-Liien Lu. Trading off cache capacity for reliability to enable low voltage operation. In *Computer Architecture, 2008. ISCA '08. 35th International Symposium on*, pages 203–214. IEEE, June 2008.

[18] Majid Shoushtari, Abbas BanaiyanMofrad, and Nikil Dutt. Exploiting Partially-Forgetful memories for approximate computing. *Embedded Systems Letters, IEEE*, 7(1):19–22, March 2015.

[19] Ik J. Chang, Debabrata Mohapatra, and Kaushik Roy. A Priority-Based 6T/8T hybrid SRAM architecture for aggressive voltage scaling in video applications. *Circuits and Systems for Video Technology, IEEE Transactions on*, 21(2):101–112, February 2011.

[20] Brian Zimmer, Borivoje Nikolic, and Krste Asanović. Resilient design methodology for Energy-Efficient SRAM. Master's thesis, EECS Department, University of California, Berkeley, May 2013.

[21] Qikai Chen, Hamid Mahmoodi, Swarup Bhunia, and Kaushik Roy. Efficient testing of SRAM with optimized march sequences and a novel DFT technique for emerging failures due to process variations. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 13(11):1286–1295, November 2005.

[22] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13(4):600–612, April 2004.

[23] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh

Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. The gem5 simulator. *SIGARCH Comput. Archit. News*, 39(2):1–7, August 2011.

[24] H. P. Labs. CACTI 6.0: A tool to model large caches. Technical report.

[25] Alireza Shafaei, Yanzhi Wang, Xue Lin, and Massoud Pedram. FinCACTI: Architectural analysis and modeling of caches with Deeply-Scaled FinFET devices. In *VLSI (ISVLSI), 2014 IEEE Computer Society Annual Symposium on*, pages 290–295. IEEE, July 2014.

[26] The International Technology Roadmap for Semiconductors. *Model for Assessment of CMOS Technologies And Roadmaps*.

[27] Stephen M. Smith and J. Michael Brady. SUSAN a new approach to low level image processing. *Int. J. Comput. Vision*, 23(1):45–78, May 1997.

[28] Matthew R. Guthaus, Jeffrey S. Ringenberg, Dan Ernst, Todd M. Austin, Trevor Mudge, and Richard B. Brown. MiBench: A free, commercially representative embedded benchmark suite. In *Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop on*, pages 3–14. IEEE, December 2001.

[29] Thomas Wiegand, Gary J. Sullivan, Gisle Bjøntegaard, and Ajay Luthra. Overview of the H.264/AVC video coding standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(7):560–576, July 2003.

[30] Axis Communication. *H.264 video compression standard*, 2008.

[31] Dmitriy Vatolin, Alexey Moskvin, Oleg Petrov, Sergey Putilin, Sergey Grishin, and Arsaev Marat. MSU video quality measurement tool (PSNR, MSE, VQM, SSIM) http://compression.ru/video/quality_measure/video_measurement_tool_en.html.

[32] Hadi Esmaeilzadeh, Adrian Sampson, Luis Ceze, and Doug Burger. Architecture support for disciplined approximate programming. *SIGPLAN Not.*, 47(4):301–312, March 2012.

Appendix A

Glossary and Acronyms

Care has been taken in this thesis to minimize the use of jargon and acronyms, but this cannot always be achieved. This appendix contains a table of acronyms and their meaning.

Table A.1: List of Acronyms

Acronym	Meaning
CMOS	Complementary metal-oxide-semiconductor
ASIC	Application-specific integrated circuit
SRAM	Static random-access memory
8T SRAM	An eight transistor variant of the SRAM cell
6T SRAM	A six transistor variant of the SRAM cell
SNM	Static Noise Margin
BER	Bit Error Rate
Vt	Threshold Voltage
ITRS	The International Technology Roadmap for Semiconductors
QoS	Quality of Service
ISA	Instruction Set Architecture
MSB	Most Significant Bit
LSB	Least Significant Bit
VA	Virtual Address

VAT	Virtual Address Table
LRU	Least Recently Used
SSIM	Structural Similarity
Aeff	Effective Area
BIST	Built-in Self Test
MSE	Mean Squared Error
PSNR	Peak Signal to Noise Ratio
PNG	Portable grey map

Appendix B

Simulator Modifications and Wrapper Scripts

A detailed description of the simulator modifications are provided here. This also gives a description of the scripts written to execute the experiments and how to use them.

B.1 gem5

The entire cache architecture implementation was done on gem5 *Classic Memory* model. The simulator was modified to take in new parameters to automate the cache setup and run experiments. gem5 was built for X86 architecture. The gem5 code with the latest changes can be obtained by cloning the repository made in the cspin servers using mercurial. Some of the implementations are similar to the one described in [18].

B.1.1 Parameter Addition

The additional parameters to gem5 such as the cache Vdd, number of precise ways, the Split and Shared LRU cache replacement policy for handling precise and approximate data, and the number of 8T SRAM cells in a pixel block of 8-bits was added. The number of acceptable faulty bits is a mechanism to switch off cache blocks if it exceeds a specified number of faults. However in this work we set this parameter to the total number of bits in a cache block because cache disabling adds unnecessary overheads to

the controller to handle flushing blocks that are being disabled. The help and usage has also been provided and can be accessed by running the following command:

```
./build/X86/gem5.fast ./configs/example/se.py -h

...
  --l1d_cvdd=L1D_CVDD    L1 cache vdd
  --l1d_num8T=L1D_NUM8T
                         number of 8T in L1
  --l1d_afb=L1D_AFB      Acceptable Faulty Bits in L1
  --l1d_cache_mode=L1D_CACHE_MODE
                         L1D cache operating in Split(0) or Shared(1) Mode
  --l1d_precise_ways=L1D_PRECISE_WAYS
                         L1D cache : number of precise ways(1 or 2)
...
```

These parameters are added in the Python Configuration files as well as Python Object classes that correspond to its C++ simulation object classes.

```
./configs/common/CacheConfig.py
./configs/common/Caches.py
./configs/common/Options.py
./src/mem/cache/BaseCache.py
./src/mem/cache/tags/Tags.py
```

B.1.2 Pseudo Instructions

The gem5 pseudo instructions are added to handle the programmer controlled approximation. When the programmer specifies the approximate declarations in the source code, the simulator treats it as a separate instruction.

```
pic->img.plane[i] = x264_malloc( size );
if(i==0)
{
        m5_DECLARE_APPROXIMATE((uint64_t)(pic->img.plane[i]),
    (uint64_t)( pic->img.plane[i] + size ));
}
```

These are added to the m5op to emit the instructions in the compiled code.

```
util/m5/m5ops.h          //Add function number here
```

```
util/m5/m5op.h          //Add function prototype here
util/m5/m5op_x86.S      //Instantiate a TWO_BYTE_OP
```

The program with these approximate declarations now can access the function prototype using the header files and can be cross-compiled as shown below. Note that we need static binaries for execution in the syscall emulation mode of gem5.

```
gcc -W -static -O4 -o <prg_approx> <prg>.c -I <path_to_gem5>/util/m5
<path_to_gem5>/util/m5/m5op_x86.S
```

We also have to overwrite reserved opcodes, the ones that are emitted by the code when compiled with m5op

```
/* src/arch/x86/isa/decoder/two_byte_opcodes.isa */
...
        0x56: DECLARE_APPROXIMATE({{
        PseudoInst::DECLARE_APPROXIMATE(xc->tcBase(), Rdi, Rsi);
                        }}, IsNonSpeculative);
        0x57: UNDECLARE_APPROXIMATE({{
        PseudoInst::UNDECLARE_APPROXIMATE(xc->tcBase(), Rdi);
                        }}, IsNonSpeculative);
```

We have added the functional simulation implementation of these calls in the following files

```
/* src/sim/pseudo_inst.hh */
void DECLARE_APPROXIMATE(ThreadContext *tc, uint64_t arg1, uint64_t arg2);
..
```

```
/* src/sim/pseudo_inst.cc */
void
DECLARE_APPROXIMATE(ThreadContext *tc, uint64_t arg1, uint64_t arg2)
{
    ...
        tc->getSystemPtr()->vAddrMap.addVirtAddr(arg1,arg2);
    ...
}
```

B.1.3 Virtual Address Table

The Virtual Address Table is a table that keeps a track of the data that can be approximated. This table is at the system-level in gem5 where it can be accessed by the core

and the cache. The behavior of this module such as checking if an input VA request is within range of any of the entries and adding and removing entries based on the pseudo instruction has been defined in the following file:

```
/* ./src/sim/system.hh  */
...
class virtualAddressMap
{
...
};
```

B.1.4 Error Model

The Error model that models the voltage dependent BER is is defined in the following files and its values are initialized based on the cache Vdd.

```
/* ./src/mem/cache/tags/base.hh  */
...
typedef struct ber{
int num8T;
    int afb;
    double ber6T;
    double ber8T;
} bitErrorRate;
```

```
/* ./src/mem/cache/tags/base.cc */

/* BER initialized in the constructor */
BaseTags::BaseTags(const Params *p)
...
cacheBitErrorRate.ber6T = f1(vdd);
cacheBitErrorRate.ber8T = f2(vdd);
...
```

B.1.5 Defect Map

The defect map is populated using a BIST mechanism during boot-up and to imitate this we have injected faults to the L1-DCache based on the BER model when the cache is initialized in the simulator environment. The mechanism scans through all the bits in

the cache and injects a fault based on the error probability and then populates a fault map. The fault map is then used to generate a status for each block based on whether a particular block has faulty bits or not. The functions that define these are implemented in the following files.

```
/* ./src/mem/cache/blk.hh */
...
void populateFaultMap(int num8T,
double ber6T,
double ber8T,
std::default_random_engine &re,
bool protectedFlag)
{
...
}
...
void setBlkState(int afb , bool protectedFlag )
{
...
}
```

Some friendly print functions are also added to help debug and prints when running gem5 in debug mode.

B.1.6 Hybrid Cache and Cache Configurations

The Hybrid cache is modeled purely based on the error rate and this is taken into consideration while populating the fault map by setting corresponding BER based on the bit positions. It is modeled in the same file given above. The 2a2p and 3a1p configurations are generated based on the parameter *l1d_precise_ways* passed into the following file which defines the base set associative tag store and initializes the cache.

```
/* ./src/mem/cache/tags/base_set_assoc.cc */
...
BaseSetAssoc::BaseSetAssoc(const Params *p)
...
{ /* Logic to control fault generation based on number of
     precise ways in a N-way set associative cache */
}
```

B.1.7 Cache Controller

The Cache Controller is critical to this architecture as it must make a decision based on multiple inputs. It gets the block's status faulty/non-faulty from the defect map and if the incoming data that's being written in is an approximate data or not. We have included both the Split and Shared LRU policies and can be toggled using the run-time parameter *l1d_cache_mode* to configure the controller accordingly. The logic has been implemented in the the following file:

```
/* ./src/mem/cache/tags/lru.cc */
...
BaseSetAssoc::BlkType*
LRU::findVictim(Addr addr, Addr v_addr) const
{
/* Function modified to map the data using v_addr */
...
    if (cache_mode == 0)
    { /* Split LRU policy implemented here */ }
    else if (cache_mode == 1)
    { /* Shared LRU policy implemented here */ }
    else
    { /* New policies for future extensions */ }
...
}
```

The cache implementation has been modified to pass the virtual addresses of fill requests to run through the VAT and determine whether the data is precise or approximate. Several changes are made in the file following file:

```
/* ./src/mem/cache/cache_impl.hh */
...
/* Pass the Virtual Address of request while allocating
   a new block on multiple scenarios */
/* Passing approximate data to core or sending
   data to higher levels of cache during eviction
   of a dirty block */
...
```

B.1.8 Fault Injection

The data is fetched from the main memory or L2 and if written in a block that has faults we need to make the faulty data available to the processor. These faults injected based on the fault-map generated during boot-up. To emulate the real hardware faults we mask the fault-map which has the fault locations with the actual data. It has been implemented in the following file:

```
/* ./src/mem/cache/blk.hh */
...
void applyFault()
{
/* read in the fault map for the blk and
   apply faults to the data in the blk */
}
...
```

B.2 PCACTI

PCACTI models the 6T and 8T device parameters in .xml files. However they are for a fixed Vdd level and but parameters like V_t (threshold voltage), mobility, NMOS-PMOS on and off currents for different voltage levels depend on voltage. We have used the ITRS MASTAR tool to generate these values. PCACTI is run using a wrapper perl script that generates the the .xml files for the corresponding Vdd levels using the parameters from MASTAR. PCACTI is run for each of these configurations which then outputs several files with power calculations for the cache. They are available in:

```
./pcacti_out/l1
./pcacti_out/l2
```

The Parameters such as Data Array Leakage, Tag Access Energy, Read and Write Energy are multiplied with corresponding events from the gem5 stats file to obtain the total energy for the cache.

B.3 Wrapper Scripts

A few perl modules are written as a necessity to automate the runs of gem5 and compute the performance, energy and accuracy. The scripts are available in:

```
/* Main scripts to generate run files for gem5
   or generate statistics based on outputs */
./run_scripts/bin/

/* The configurations that drive all the runs
   and determine the statistics to output */
./run_scripts/configs/

/* The modules to generate scripts,
   grab statistics, generate plot files for
   gnuplot etc. */
./run_scripts/lib/
```

The modules introduced with several functions to make the runs and data extraction easy is listed below:

```
./run_scripts/lib/MyPerlMods
CactiInfo.pm            //Grab statistics info from pcacti_out
ConfigGrab.pm           //Read in the Configuration files
GenRunScripts.pm        //Generate run-scripts for gem5
GnuPlotFiles.pm         //Generate gnuplot friendly .dat files
StatGrab.pm             //Grab statistics from gem5 runs
StatMultiplier.pm       //Multiply Cacti and gem5 statistics
```

B.4 Matlab

Matlab was used to generate the SSIM values which is used as an accuracy metric. The SSIM metric calculation was done using the scripts provided by [22]. When gem5 is run in the syscall emulation mode the benchmark writes it outputs which in this case is either a image or a video file. gem5 is run without any approximation and this is used as the *golden* file for comparison. Each of the files output by gem5 runs are compared to this *golden* file.

```
/* Excerpt from Matlab ssim calculation */
```

```
...
imageData = imread(pgmFileName);
ssimValues(k,j)=ssim(baseImg,imageData);
...
```

B.5 MSU VQMT

We have used the MSU_VQMT tool [31] to measure the SSIM values for each frame of the encoded video while comparing to the *golden* file. It outputs a csv file with the frame number and the SSIM metric values.