Spring 5-17-2017

Investigating Read/Write Aggregation to Exploit Power Reduction Opportunities Using Dual Supply Voltages

Gu Yunfei
Washington University in St Louis

WASHINGTON UNIVERSITY IN ST. LOUIS

School of Engineering and Applied Science

Department of Electrical Engineering

Thesis Examination Committee:
Xuan Zhang, Chair
Roger Chamberlain
Shantanu Chakrabartty

Investigating Read/Write Aggregation to Exploit Power Reduction Opportunities Using Dual Supply Voltages

by

Yunfei Gu

A thesis presented to
School of Engineering and Applied Science
of Washington University in St. Louis in partial fulfillment of the
requirements for the degree of
Master of Science

May 2017
St. Louis, Missouri

Table of Contents

List of Figures

List of Tables

Acknowledgments

Firstly, I would like to thank my thesis advisor Professor Xuan Zhang of Electrical Engineering at Washington University in St. Louis. She consistently instructed my research and this paper to the right direction whenever she thought I needed it.

Secondly, I would also like to thank my thesis committee members: Professor Roger Chamberlain and Professor Shantanu Chakrabartty. Without their passionate participation and feedback, my thesis defense could not have been successfully conducted.

Finally, I would like to acknowledge all the members in Professor Xuan Zhang's lab as the second readers of this thesis, and I am gratefully indebted to their very valuable comments on this thesis.

Yunfei Gu

Washington University in St. Louis

May 2017

ABSTRACT OF THE THESIS

Investigating Read/Write Aggregation to Exploit Power Reduction Opportunities Using Dual Supply Voltages

by

Yunfei Gu

Master in Electrical Engineering

Washington University in St. Louis, 2017

Professor Xuan Zhang, Chair

Power consumption plays an important role in computer system design today. On-chip memory structures such as multi-level cache make up a significant proportion of total power consumption of CPU or Application-Specific Integrated Circuit (AISC) chip, especially for memory-intensive application, such as floating-point computation and machine learning algorithm. Therefore, there is a clear motivation to reduce power consumption of these memory structures that are mostly consisting of Static Random-Access Memory (SRAM) blocks. In this defense, I will present the framework of a novel dual-supply-voltage scheme that uses separate voltage levels for memory read and write operations. By quantitatively analyzing the cache trace for SPEC2000, Parsec, and Cortexsuite benchmarks and comparing the Read/Write sequence characterization of different computing application types, I discover that memory-intensive applications have high potential to generate long consecutive Read/Write sequences, which can be leveraged by our proposed dual-supply framework. I then perform a limit study based on ideal Read/Write re-ordering to obtain the maximum possible power saving estimate.

Finally, as a case study, I apply this framework to a custom machine learning ASIC accelerator design to showcase its viability.

Chapter 1: Introduction

In this chapter, we will describe the motivation and the basic knowledge that used in this thesis.

1.1 Motivation

Nowadays, processors are using in many different engineering applications and research areas, as shown in Figure 1.1, including mobile system-on-chip (SoC), wireless sensor and battery-less internet of things (IoT). As we all know, these applications and areas usually have several challenges and one of those challenges is the power consumption. The most important reason why the power it becomes the challenge for these is that the mobile SoC and wireless sensors are usually working in battery-operated (the battery has low power capacity) devices and battery-less IoT even doesn't have battery supply.

Figure 1.1 Mobile SoC, Battery-less IoT and Wireless Sensors [1]

In addition, the power cost of processors has abstracted researcher or engineer's attention as a necessary part of these applications so far. Because the processor power usually cost more power than most of other components in these application hardware design. For instance, Schandy et. al. [17], Table 1-1 shows the power breakdown of a wireless sensor

work. And Figure 1.2 shows the power breakdown of a smartphone suspended state from Carroll et. al. [18].

Table 1-1 Energest results with CCR at 4Hz and an observe period of 60s [17]

$Node\ state$	$DC(\%)$	$P_{avg}(\mu W)$
CPU	0.61	114.51
TX	0.11	64.35
RX	0.35	227.4
LPM	98.53	163.5
IRQ	0.40	75.6

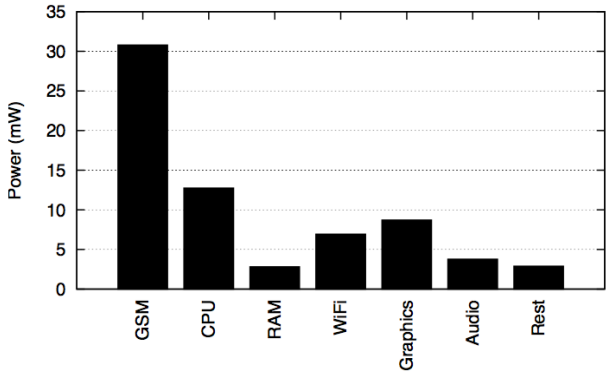

Figure 1.2 Power breakdown in suspended state of smartphone [18]

From Table 1-1 and Figure 1.2, we clearly see that no matter in wireless sensor or mobile device hardware design, the CPU cost second or third most energy of total system. Therefore, the power consumption of CPU is a critical part that has to be taken into consideration.

Moreover, when researchers or engineers think about saving the power of processor, they are always focusing on computational units and network-on-chip (NoC). However, we are trying to find other venues to reduce the power consumption of processor. And then we do some analysis in Section 1.2 to look for power saving opportunities.

1.2 Cache Power Analysis

Firstly, in order to find other power saving chance, we look into the detailed die photos of three different type processors to analyze the characteristics of them. These three type processors include CPU, ASIC chip, and ARM SoC, as Figure 1.3 (a) (b) (c) shows.

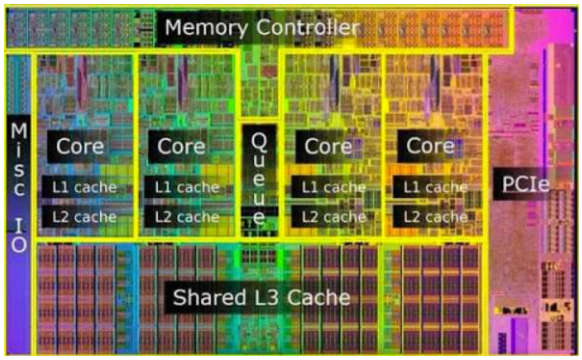

Figure 1.3 (a) Intel CPU die [2]

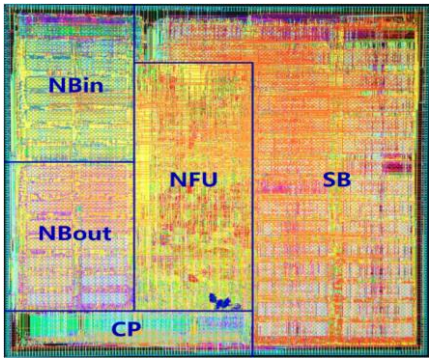

Figure 1.3 (b) DianNao ASIC chip die [3]

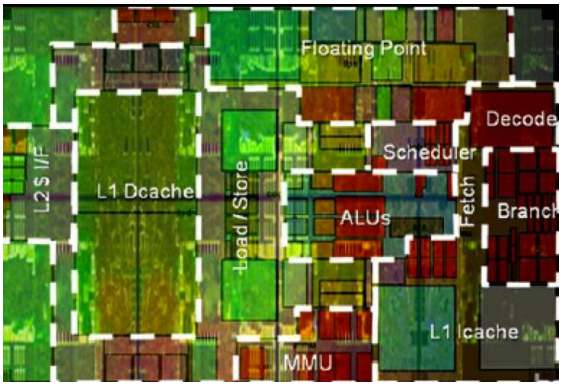

Figure 1.3 (c) Nvidia ARM SoC die photos [4]

In Figure 1.3(b), the input buffer for input neurons (NBin), output buffer for output neurons (NBout), and buffer for synaptic weights (SB) [2] are built by SRAM which is the same as how L1cache, L2cache and L3cache are built in CPU or ARM SoC. Obviously, we notice that the cache structure takes a large part of total area in these different die examples. Therefore, we intuitively think that the cache should also take a large percentage of total power consumption, at least of total leakage power.

Secondly, we use McPAT [5] and Gem5 [6] with the configuration that is shown in Table 1-2 to get the leakage and dynamic power breakdowns of CPU simulation results. And these results proved our previously intuitive conclusion.

Table 1-2 System Configuration of McPAT and Gem5

Architecture	X86
Temperature	380K
Clock freq.	3.4Ghz
Feature size	32nm
Functional units	alu 6 mul 1 fpu 2
Branch Prediction	2KB BTB, 16-entry RAS
fetch	16-entry buffer, Min. 8 cycles fetch-dispatch time
L1 Dcache	32kB, 8 associativity, 2 cycle hit latency
L1 Icache	32kB, 4 associativity, 2 cycle hit latency
L2 cache	private 256kB, 8 associativity, 6 cycle access latency
L3 cache	shared, 2MB, 16 associativity, 14 cycle access latency
Coherence protocol	MESI (modified, Exclusive, Shared, Invalid), directory
Main Memory	2GB DDR3-1066

Figure 1.4 shows the leakage power breakdown of CPU simulated by McPAT. The CPU leakage power breakdown shows that L1cache and L2cache cost about 23.67% of total leakage power and last level cache (LLC) consumes 43.25% leakage power approximately. It indicates that cache takes a large percentage of leakage power in total.

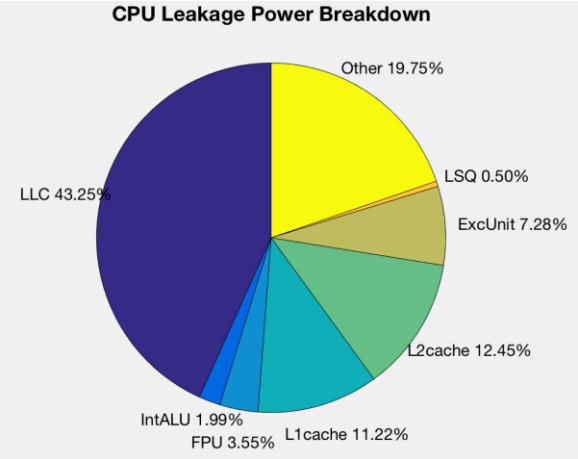

Figure 1.4 CPU leakage power breakdown

After analyzing leakage power breakdown, we turn to Gem5-McPAT CPU dynamic power breakdowns simulation result, as shown in Figure 1.5 (a) and Figure 1.5 (b). Here, Figure 1.5 (a) shows the CPU dynamic power breakdown running result without loss of generality. Since this run time analysis is based on different type of benchmarks, containing SPEC2000 [6] and Cortexsuite [7]. SPEC2000 is the next-generation industry-standardized CPU-intensive benchmark suite. It provides a comparative measure of compute intensive performance across the widest practical range of hardware. The implementation resulted in source code benchmarks developed from real user applications developed from real user applications. These benchmarks measure the performance of the processor, memory and compiler on the tested system [6]. And Cortexsuite is a Synthetic Brain Benchmark Suite which captures an emerging workload that is clustered around providing information processing capabilities that human brains have today but that computers are only just now beginning to become good at [8]. In order to get the CPU dynamic power breakdown without loss generality. After analyzing the CPU dynamic power breakdown of each type benchmark, we use this result to come out the average CPU dynamic power breakdown, as Figure 1.5 (b)shows.

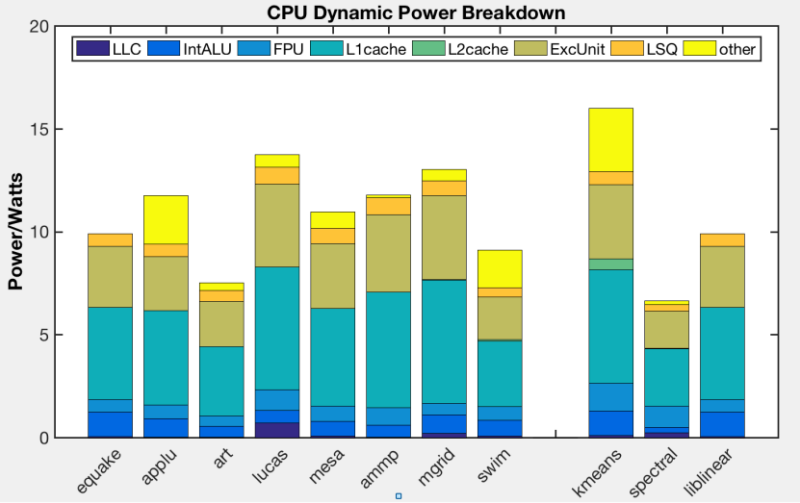

Figure 1.5 (a) Dynamic power breakdown of CPU

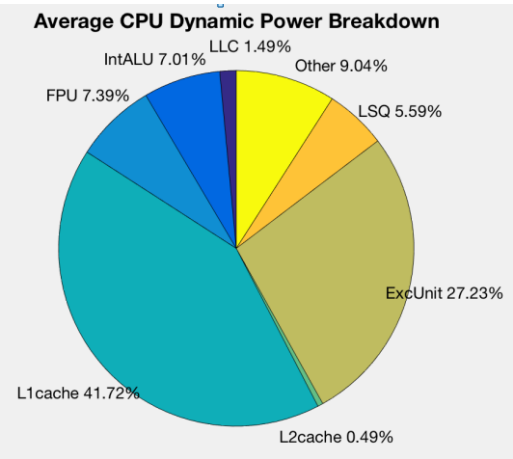

Figure 1.5 (b) Average dynamic power breakdown of CPU

These two figures show that L1cache and L2cache cost 42.21% of total dynamic power, and LLC consumes 1.49%. Combine Figure 1.4, Figure 1.5 (a) and Figure 1.5 (b) together, we conclude that cache costs a much power of total power consumption of CPU. Therefore, there is a change to lower the CPU power consuming by reducing the power cost of cache. Since cache usually consists of SRAM blocks, we focus on exploiting opportunities to save the power of SRAM.

Chapter 2: Background

In this chapter, we will introduce the background of our work.

2.1 6 Transistor (6-T) SRAM

SRAM can retain its stored information as long as power is supplied. The structure of a 6-T SRAM cell, storing one bit of information, can be seen in Figure 2.1.

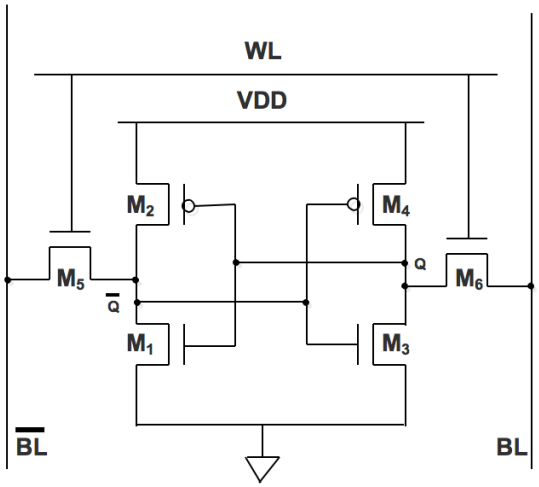

Figure 2.1 6T SRAM

The core of the cell is formed by two CMOS inverters, where the output potential of each inverter V_{out} is fed as input into the other V_{in}. This feedback loop stabilizes the inverters to their respective state. The access transistors and the word and bit lines, Word line (WL) and Bit line (BL), are used to read and write from or to the cell. In standby mode the word line is low, turning the access transistors off. In this state, the inverters are in complementary state. When the p-channel MOSFET of the left inverter is turned on, the potential $\bar{Q}$ is high and the p-channel MOSFET of inverter two is turned off, Q is low. To write information the data is imposed on the bit line and the inverse data on the inverse bit line, $\overline{BL}$. Then the access transistors are turned on by setting the word line to high. As

the driver of the bit lines is much stronger it can assert the inverter transistors. As soon as the information is stored in the inverters, the access transistors can be turned off and the information in the inverter is preserved. For reading the word line is turned on to activate the access transistors while the information is sensed at the bit lines. [9]

2.2 Static Noise Margin (SNM)

Static Noise Margin (SNM) is an important evaluation of the SRAM cell stability. It can be extracted by nesting the largest possible square in the two voltage transfer curves (VTC) of the involved CMOS inverters, as seen in Figure 2.2. In this figure, V_R means the voltage of the right transistor M6 and V_L means voltage of the left transistor M5.

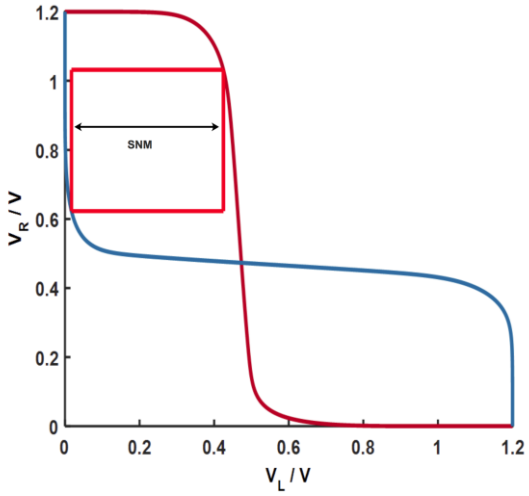

Figure 2.2 SNM [10]

The SNM is defined as the side-length of the square, given in volts. When an external DC noise is larger than the SNM, the state of the SRAM cell can change and data is lost [9]. SNM helps to analyze the reliability of read and write operations, the purpose is to guide the size of 6-T unit to achieve the nominal operating voltage to achieve balanced reading

and writing capabilities. The SNM is obtained by the overlapping voltage transfer curve (VTC) of the cross-coupled inverter in the cell.

2.3 Read Static Noise Margin (RSNM) and Write Static Noise Margin (WSNM)

During measuring RSNM, the Read-SNM decreases. Since that Read-SNM is calculated when the word line is set “1” and BL is still pre-charged “1”. If there is a read access, then WL is set to “1” and the bit-lines are already pre-charged to “1”. The internal node of the bit-cell representing a zero gets pulled upward through the access transistor due to the voltage dividing effect across the access transistor and drive transistor. During the read operation, a stored “0” can be overwritten by a “1” when the voltage at node V1 reaches the V_{th} of nMOS N1 to pull node V2 down to “0” and in turn pull node V1 up even further to “1” due to the mechanism of positive feedback. This results in wrong data being read or a destructive read when the cell changes state [20].

The write noise margin is defined as the minimum BL voltage needed to flip the state of cell. During a write operation, the input data are sent to BL, and then the word lines are activated to access the cell. BL that is charged to ‘0’ pulls the node of the cell storing ‘1’ to ‘0’ causing the cell to flip state. Since the cross-coupled inverters have complementary data, their VTCs are measured using different circuits. The circuit that represents the inverter with ‘1’ at its output and its BL is connected to GND to simulate a write ‘0’ to that node. A DC voltage sweep is applied at node V1 and the voltage output at node V2 is measured, when the BL is connected to GND and WL is charged to VDD [20].

Figures 2 (a) and (b) show the read SNM (RSNM) and write SNM (WSNM) of the 6-T unit, which indicates that it is able to successfully read and write at that voltage [19].

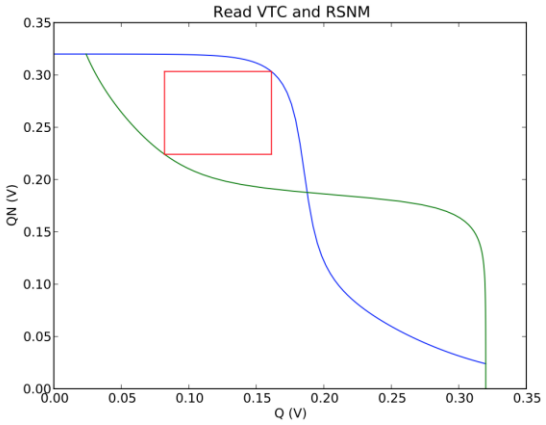

Figure 2.3 (a) RSNM [19]

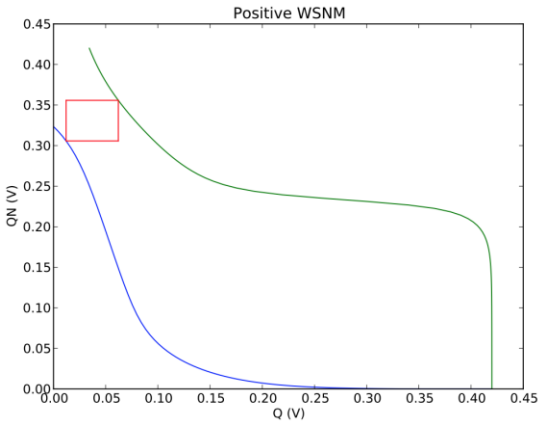

Figure 2.3 (b) Positive WSNM [19]

From previous reported [19], the sizing of the 6-T cell is determined by SNM analysis to have balanced read and write stability at its nominal voltage, and read and write reliability will change when VDD scales. Figure 2.4 shows the nominal WSNM and RSNM degradation with VDD. Here, WSNM is higher than RSNM at high VDD, but WSNM is smaller than RSNM at low VDD, which is resulted from a much steeper slope of WSNM than RSNM, meaning WSNM degrades with VDD scaling much faster than RSNM does.

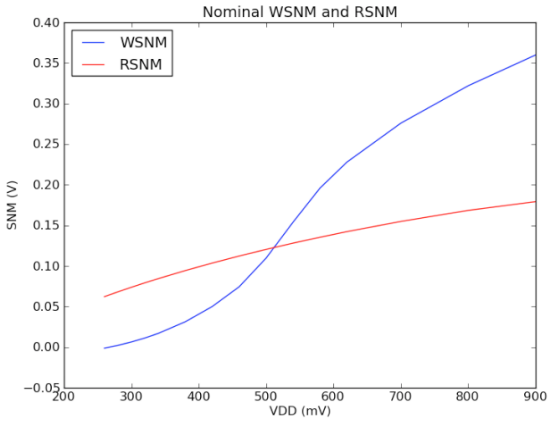

Figure 2.4 Nominal WSNM and RSNM degradation with VDD [19]

From the nominal WSNM and RSNM degradation with VDD property, we hypothesize that there should be a voltage overhead for read operation at low voltage, which means maybe the minimum voltage for reliable read operation is lower than the minimum voltage for write operation at low voltage level.

2.4 Reliable Read/Write Voltage Characterization

Refer to the Cadence and Cacti simulation results From Yan et. al. [10], our hypothesis in Sectioin 2.3 is proved to be solid. In Figure 2.5, it shows SRAM reliable Read/Write voltage vs. frequency characterization. And in Figure 2.6, it shows SRAM voltage vs. power characterization.

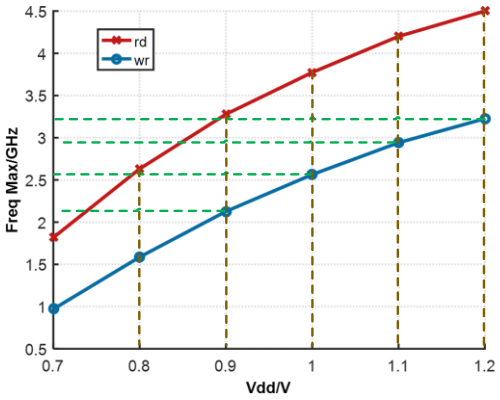

Figure 2.5 SRAM reliable Read/Write voltage vs. frequency Charaterization [10]

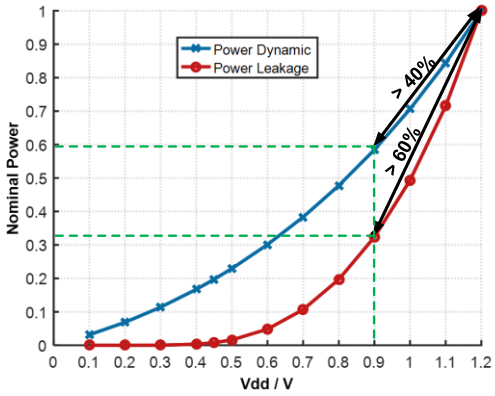

Figure 2.6 SRAM voltage vs. power characterization [10]

From Figure 2.5, we notice that the voltage for reliable read operation is lower always lower than write operation at the same frequency. For example, V_{read} = 0.9V while V_{write} = 1.2V, at max frequency 3.2GHz. And Figure 2.6 shows that if we sweep the voltage supply from 1.2V to 0.9V, the dynamic power will be reduced over 40% and the leakage power will be reduced over 60%. It indicates that there is the power overhead for read

operation. Therefore, we have the chance to reduce the power consumption of SRAM by separating the voltage supply for read and write operations respectively, which is setting read supply voltage to 0.9V and setting write supply voltage to 1.2V.

2.5 Core-level Fast Switching Circuit

In order to separate the voltage supply for read and write operations respectively, we refer the core-level fast voltage switching circuit shown in Figure 2.7 from Booster [12].

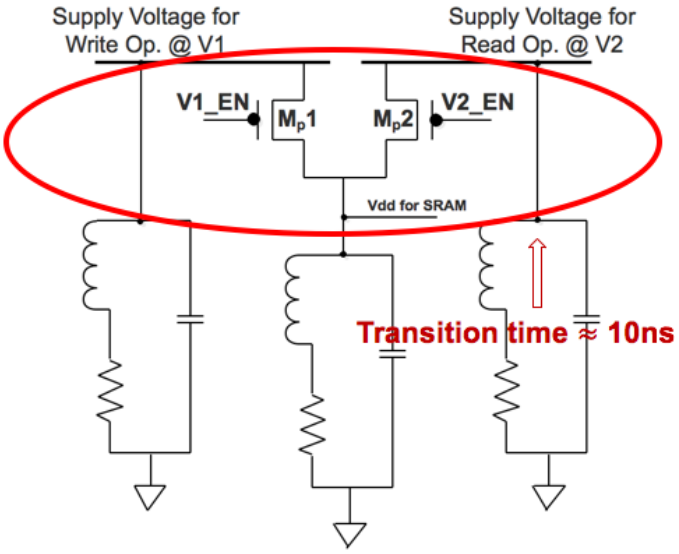

Figure 2.7 Core-level fast voltage switching circuit [12]

As Figure 2.7 shows, if V1_EN is set to 1 and V2_EN is set to 0, then M_p1 ON and M_p2 OFF, the VDD for SRAM switch to supply voltage for write operation @ V1. On the contrary, if V1_EN is set to 0 and V2_EN is set to 1, then M_p1 OFF and M_p2 ON, then VDD for SRAM switch to supply voltage for read operation @ V2. Also, the power supply could be switched from V1 to V2 or switch from V2 to V1 but with approximate 10 nanoseconds delay, as Figure 2.8 shows. Even though there is a delay to be reckoned in this fast switching circuit, it can make the voltage supply switch possible.

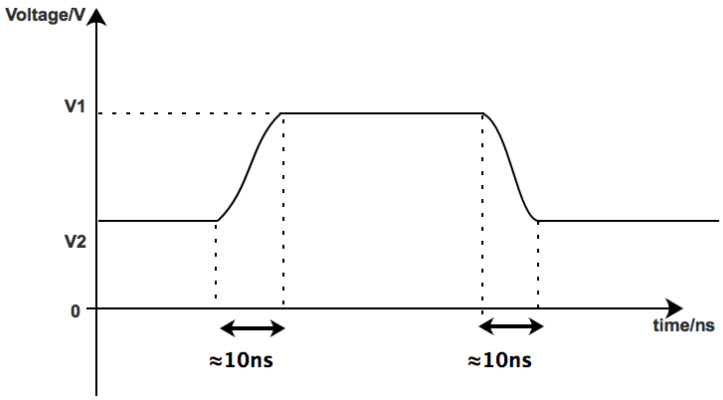

Figure 2.8 Delay for core-level fast voltage switching circuit [12]

2.6 Related Framework

Combine the conclusion and basic work in Section 2.2 and Section 2.3, we plan to use the framework proposed in Yan et. al. [10] as the baseline for my following work. The related framework contains two parts, multi-voltage supply switches circuit design and multi-voltage supply switches control scheme. Figure 2.9 shows the multi-voltage supply switches circuit design.

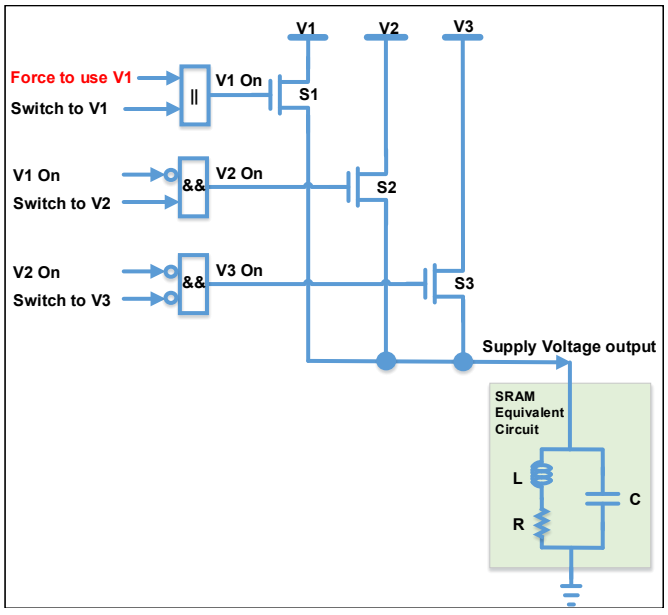

Figure 2.9 Multi-voltage supply switches circuit [10]

In this circuit design, V1 (write voltage) > V2 (read voltage) > V3 (hold voltage). And Supply Voltage output port connects with VDD of SRAM.

Figure 2.10 shows the multi-voltage supply switches control scheme.

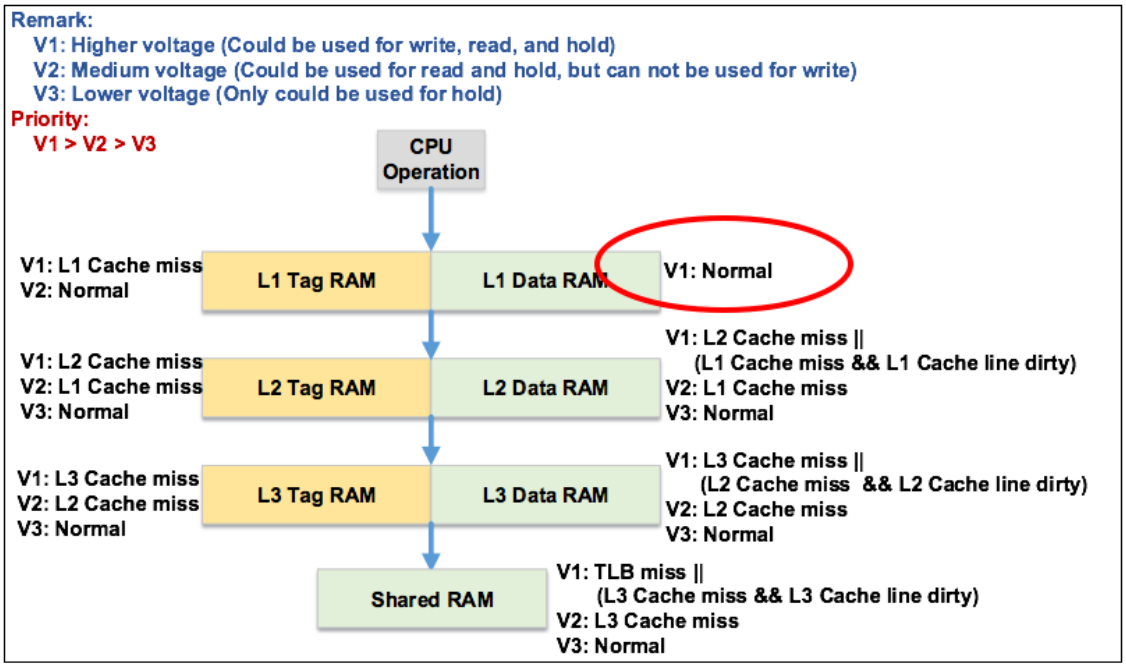

Figure 2.10 Multi-voltage supply switches control scheme [10]

By analyzing this multi-voltage supply switches control scheme work associated with multi-voltage supply switches circuit design, we agree with that this system could reduce the power consumption of L1 Tag RAM, L2cache and LLC as much as possible. However, in this frame work, since L1cache is connected with CPU, and it will never be in idle status, it cannot use the information such as cache miss or cache line dirty to decide whether there is read or write operation and then control the voltage supply to switch voltage. Hence, the voltage supply for L1-Dcache are always set to V1 which is write voltage (higher voltage). It improves nothing to L1-Dcache. However, there should be some chances that could reduce the power consuming of L1-Dcache. Because there

are indeed two different type operations between CPU and L1-Dcache, including Read operation or Write operation. Therefore, we try to exploit the chance to lower the L1-Dcache power using dual supply voltages for Read/Write operations respectively.

Chapter 3: Limit Study

In this chapter, we will do several limit studies built upon the assumptions derived in Chapter 2. And we will analyze and discuss the result of different scenarios in this chapter.

3.1 L1cache Power Saving in Ideal Case

The Ideal Case is that we assume that Booster core-level fast voltage switching circuit will cost nothing. So that we aggressively switch the dual voltage supply to V1 once there is a single write operation and then switch it to V2 once there is a single read operation, as Figure 3.1 shows.

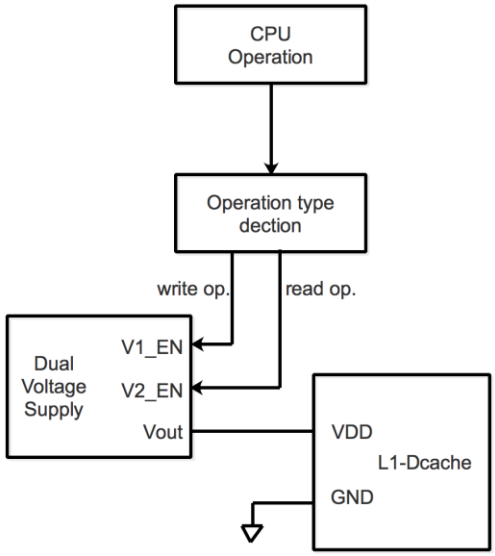

Figure 3.1 Dual supply voltages aggressively switch scheme

In this scenario, the dual supply voltages will switch to V2 for each read operation. It will save 60% dynamic power and 40% dynamic power per read operation based on the voltage vs. power characterization that is shown in Section 2.2. And if we implement this ideal case to analyze power saving for different benchmarks, including SPEC2000 and

Cortexsuite using Gem5 still with configuration shown in Table 1-1, we will get the power saving result shown in Figure 3.2.

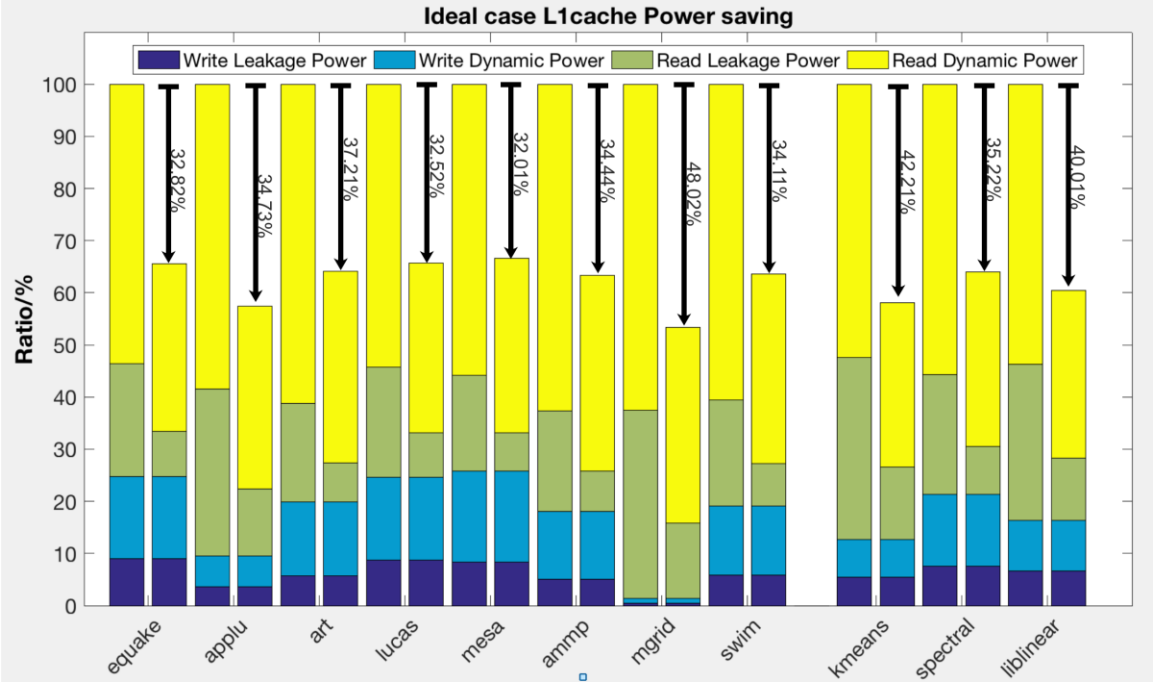

Figure 3.2 L1-Dcache power saving

From Figure 3.2, we could clearly get that the power consumption of L1-Dcache will be reduced by about 32%~48% in this ideal case.

However, there is a limitation that we cannot ignore that nearly 10 nanoseconds delay within each switch for switching the voltage supply from V2 to V1. Therefore, we analyze the delay penalty in Section 3.2.

3.2 Delay Penalty

Due to the limitation that is mentioned in Section 3.1, we take that approximate 10 nanoseconds delay into consideration. When there is a write operation, which is shown in Figure 3.3 red zone. The dual supply voltages switch from V2 to V1. But there is a nearly 10ns delay each switch from V2 to V1. Before the supply voltage reached V1, the write

operation has to stand by since the voltage range from V2 to V1 is not reliable for write operation. So that there is delay penalty when the dual supply voltages switch from V2 to V1. However, when there comes a read operation that is shown in Figure 3.3 green zone. The dual supply voltages will switch from V1 to V2. In this voltage range from V1 to V2 it is always reliable for read operation. Hence, the read operation does not have to stand by when the dual supply voltages switch. There is no delay penalty when dual supply voltages switch from V1 to V2.

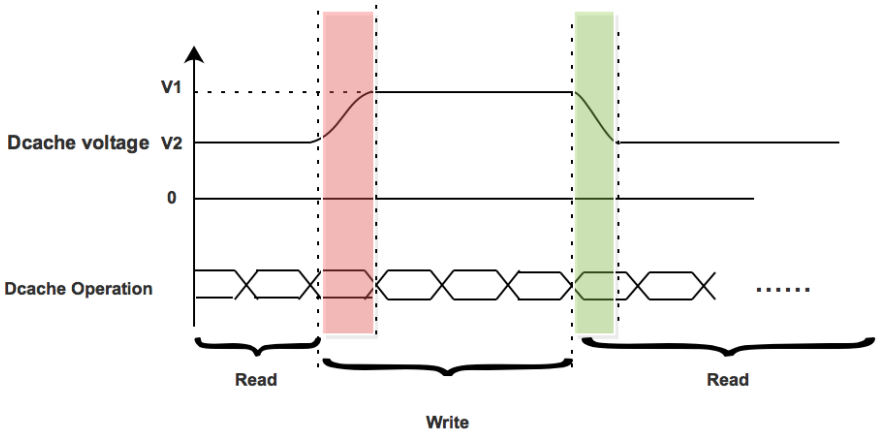

Figure 3.3 Delay penalty analysis

After analyzing the principle case delay penalty in Figure 3.3, we use Gem5 getting the cache trace of several benchmarks to analyze the specific delay penalty of each benchmark. The table 3.1 shows the delay penalty of some benchmarks from SPEC2000 and Cortexsuite. In this table, *Total accesses* means the total number of all read operations and write operations. *Execute times* indicates the entire execute time to run this benchmark. *V2->V1 switches* indicates that the number of total times that the dual supply voltages switch from V2 to V1. *Additional delay* is calculated by V2->V1 switches times 10ns. And we use the *Delay penalty evaluation factor* to represent that

how the delay penalty worse for that benchmark. It is computed by additional delay / execute time. And the evaluation criteria is that the factor is bigger, the delay penalty for this benchmark is worse.

Table 3-1 Delay penalty of benchmark

benchmark	total accesses	Execute time(s)	V2->V1 switches	Additional Delay(s)	Delay penalty evaluation factor
SPEC2000.equake	935129301	1.73996	123876868	1.23876868	0.71195239
SPEC2000.applu	1122349933	1.93352	172334187	1.72334187	0.89129767
SPEC2000.lucas	5248665450	4.1319	1183496319	11.83496319	2.864290808
SPEC2000.mesa	522399301	0.67724	56352468	0.56352468	0.832090071
SPEC2000.ammp	5517823360	11.68834	990013392	9.90013392	0.847009406
SPEC2000.mgrid	991123344	1.07396	113029845	1.13029845	1.052458611
SPEC2000.swim	768312248	0.63521	45393923	0.45393923	0.714628595
Cortexsuite.kmeans	943510231	1.07342	21337074	0.213370745	0.198776569
Cortexsuite.rbm	4225909112	6.99832	102807119	1.028071193	0.14690257
Cortexsuite.liblinear	5300981029	8.22453	114633702	1.14633702	0.139380247

If we compare the data in Table 3-1 vertically, it is not hard to find that the last three benchmarks delay penalty evaluation factors are smaller than others relatively. If we trace the original data to deliberate this result, we will to notice that the *V2->V1 switches* are much smaller than other benchmarks relatively. Therefore, the last three benchmarks cache trace has smaller number of dual supply voltages switching from V2 to V1, which means they has the smaller times that read operation jumps to write operation. In other words, it seems that there are less opportunities that the write operation could interrupt the long consecutive read sequence in last three benchmarks, which indicates that there should be a large number of long consecutive read sequences in Cortexsuite benchmark cache trace.

3.3 Cumulative Distribution Function (CDF) of Consecutive Read Sequence

In order to verify the above hypothesis, we will introduce a kind of Cumulative Distribution Function (CDF) of consecutive read sequence to represent the long

consecutive read sequence potential in different benchmark. The following formula shows how does this CDF work.

$$F(X \geq x) = \frac{\sum_n s \times t}{r} \times 100\%$$

$F(X \geq x)$: Cumulative probability of consecutive read sequence.

n: The number of total categories of consecutive read sequence with different size

s: $\log_{10}$(The size of consecutive read sequence for each category)

t: The occur times of each consecutive size read sequence category

r: The number of total read operations

Figure 3.4 shows an example of a CDF of consecutive read sequence testing by SPEC2000 ammp benchmark.

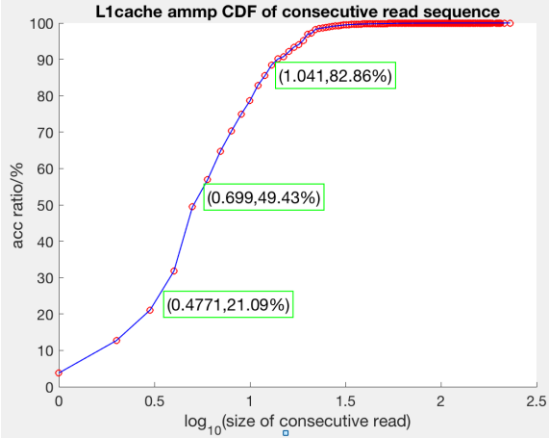

Figure 3.4 ammp CDF of consecutive read sequence

In Figure 3.4, there are three text points, '0.4771,21.09%', '0.699,49.43%' and '1.041,82.86%'. These three text points indicate that $F(X \leq 10^{0.4771}) = 21.09\%$, $F(X \leq 10^{0.699}) = 49.43\%$ and $F(X \leq 10^{1.041}) = 82.86\%$ respectively. $10^{1.041}$ is a 10s number, which means that 10s consecutive read sequence takes a large percent of total read

accesses. Therefore, the ammp benchmark cache trace does not have a good potential of existing long consecutive read sequence.

For the CDF of consecutive read sequence curve, the greater the value of the horizontal axis corresponding to these three percentages (20%, 50% and 80%), the higher long consecutive read sequence potential it has in that benchmark cache trace.

In section Section 3.2, we assume that the reason why Cortexsuite benchmarks have small delay penalty evaluation factor is that there should be a large number of long consecutive read sequences in these benchmarks. Therefore, we compare the ammp, kmeans and rbm CDF of consecutive read sequence that are shown in Figure 3.4, 3.5 and 3.6 respectively. In Figure 3.5, we could calculate that the consecutive read sequence size corresponding to 20%, 50% and 80% are 10^{0}, $10^{2.794}$ and $10^{2.993}$. In Figure 3.6, these three numbers are $10^{0.9031}$, $10^{3.654}$ and $10^{4.021}$ which are almost one magnitude order larger than before. Obviously, SPEC2000 ammp benchmark has the worst potential of long consecutive read sequence and Cortexsuite rbm benchmark has the best potential.

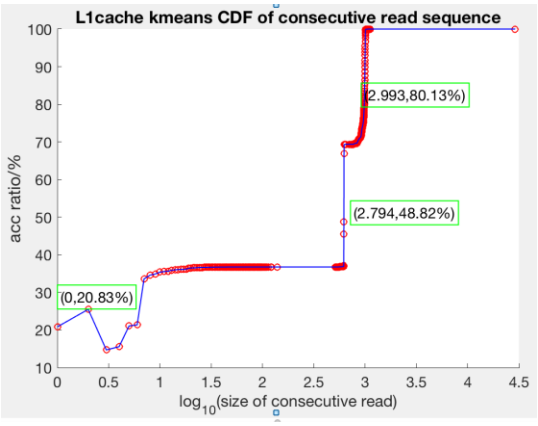

Figure 3.5 kmeans CDF of consecutive read sequence

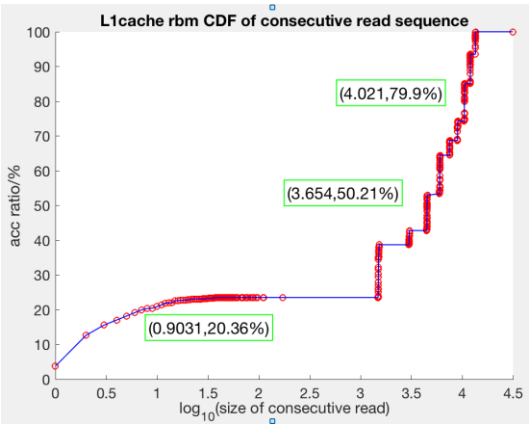

Figure 3.6 rbm CDF of consecutive read sequence

And from the category property of these benchmark, we conclude that memory-intensive benchmark like machine learning benchmark has high potential existing lots of long

consecutive read sequence in their cache trace, but computation-intensive benchmark does not. Hence, memory-intensive application has more opportunity to lower cache power consuming using dual supply voltages.

3.4 Read/Write Prediction Exploration

From the delay penalty analysis in Section 3.2, we conclude that the delay penalty will occur when the dual supply voltages switch from V2 to V1 because of write operation coming, but the voltage range from V2 to V1 is not reliable for write operation. So that if we can know there is a write operation coming in advance, we will eliminate this kind of delay penalty. Then we try to detect some useful information in history cache trace and assembly code. By analyzing the property of program, we hypothesize that whenever there is a function call or loop, the program should jump to a pointer and repeat some same Read/Write operations periodically. In order to verify this assumption, we write an assembly pseudo code with two loops in it, as table 3-2 shows.

Table 3-2 Assembly pseudo code with loop

Assembly Pseudo Code Loop

.cfi_startproc
pushq %rbp
.cfi_def_cfa_offset 16
.cfi_offset 6, -16
movq %rsp, %rbp
.cfi_def_cfa_register 6
movl $0, -4(%rbp)
movl $0, -8(%rbp)
jmp .L2 *// jump into L2 loop*
.L3 // *L3 loop*
movl -8(%rbp), %eax
cltq
movl array_test1(,%rax,4), %eax
addl %eax, -4(%rbp)
.L2 // *L2 loop*
cmpl $15, -8(%rbp)
jle.L3 *//jump to L3 loop if l equals to*
movl $0, %eax
popq %rbp
.cfi_def_cfa 7, 8
ret
.cfi_endproc
.LFE0

In table 3-2 shown assembly pseudo code, we know that there are two loops, L2 loop and L3 loop. In this code, these two loops are nested, which is that L2 loop contains L3 loop. Therefore, we assume that there should also be two nested loops in Read/Write operations of L1-Dcache trace. Then we use sliding window FFT to analyze the L1-Dcache trace. And we find that if we set a proper stride we will get the sliding window FFT result as Figure 3.7 (a), (b), (c), (d), (e) and (f) shows.

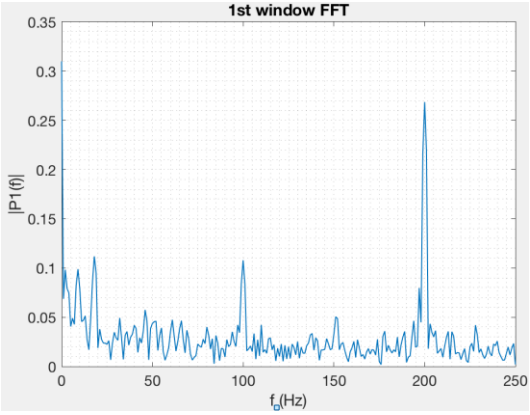

Figure 3.7 (a) 1st window FFT

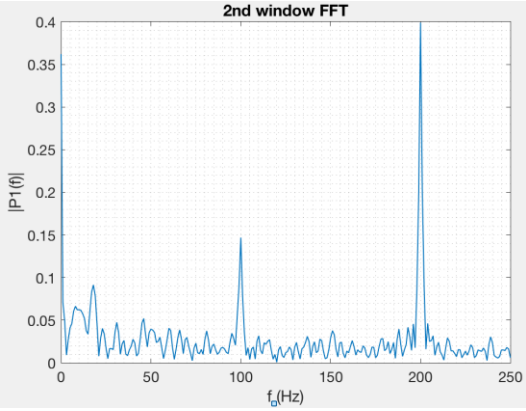

Figure 3.7 (b) 2nd window FFT

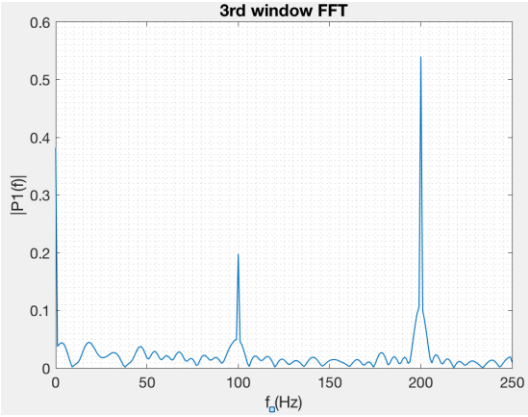

Figure 3.7 (c) 3th window FFT

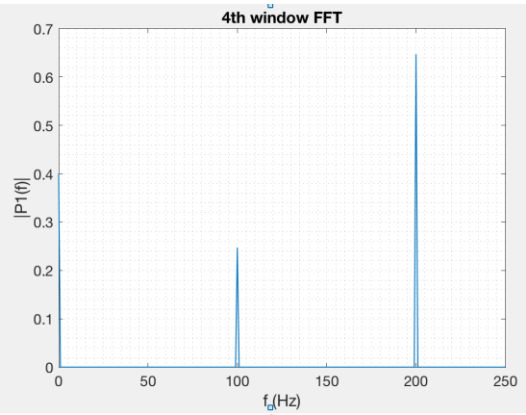

Figure 3.7 (d) 4th window FFT

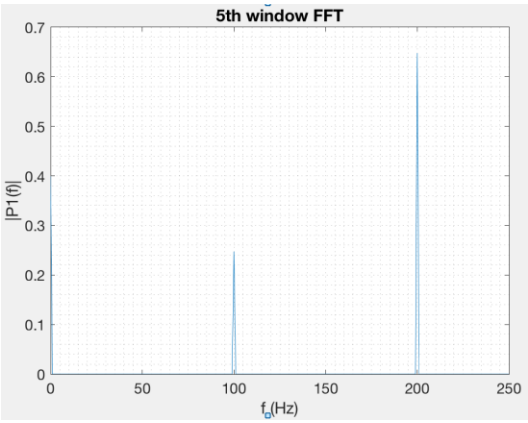

Figure 3.7 (e) 5th window FFT

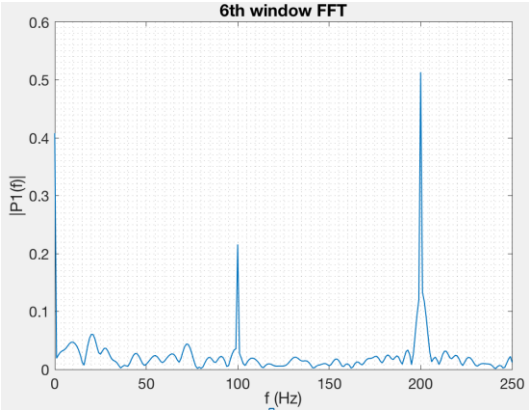

Figure 3.7 (f) 6th window FFT

In Figure 3.7 (a), (b) and (c), there are two peak with large magnitude, but also with some miscellaneous peaks of small magnitude. These characterizations indicate that there are not only two loops in the 1st, 2nd and 3rd window, but also some irregular operations in those windows. However, in Figure 3.7 (d) and (e), there are two clearly peaks without any miscellaneous peak. They indicate that there are exactly two loops without any other irregular operation in 4th and 5th window. And in Figure 3.7 (f), it is in the same situation with Figure 3.7 (a), (b) and (c) approximately, which means the loop pattern has been passed by 6th window, and this pattern is locked in 4th and 5th window.

Therefore, we can use the sliding window FFT algorithm to recognize the repeat pattern in the entire cache trace associated with instruction trace. Assume that if we record the instruction and address corresponding with the approximate start point of the loop when we detect clear peaks using sliding window FFT algorithm, we will be able to predict the read and write sequence once there is another loop that we have recognized, especially for the system call function and loop computation since the system call function has a fixed data access sequence as well as loop computation.

However, there is a limitation to predict read and write sequence of L1-Dcache using fast sliding window FFT. Since the L1-Dcache is connected with CPU. The L1-Dcache access speed is almost equals to CPU nearly 3.2~3.4GHz. It is impossible to do such fast speed sliding window FFT analysis. Even there is some other kinds of pattern recognition algorithm to do such the similar loop detection thing, maybe the power we cost by doing such analysis is much more than we can save.

Simultaneously, we also try to use other predictor mechanisms such as cache prefetching and branch predictor to predict read and write operation sequence. Figure 3.8 shows the cache prefetching stream buffers.

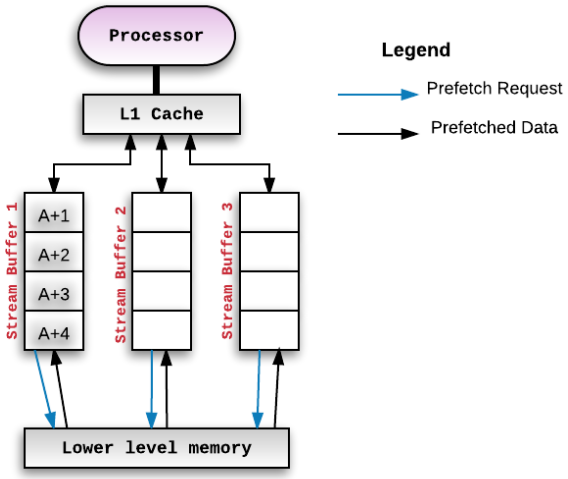

Figure 3.8 Cache prefetching stream buffers [13]

In Figure 3.8, there are several stream buffers between L1 cache and lower level memory. The main purpose of this structure is to address the memory access latency. Since the stream buffers will prefetch the data that maybe used in the near future. It seems that it could help us to predict the near future read and write operation. However, the data in the stream buffer is not the exactly data that will be firmly used to be accessed by L1 cache.

The prediction error rate may bring more penalty. Also, it can do nothing to predict the read and write operation between CPU and L1 cache. Therefore, the cache prefetching mechanism maybe a not good choice to predict the Read/Write sequence.

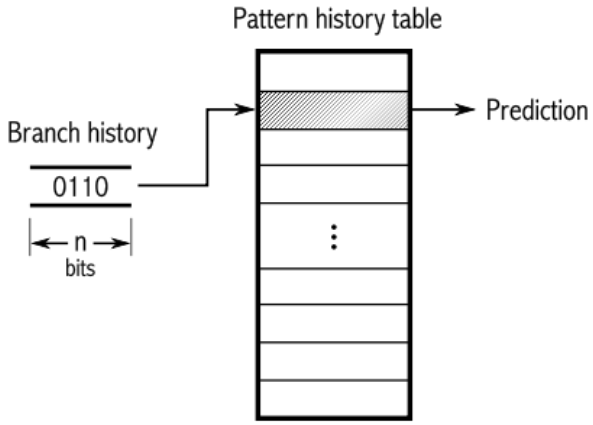

Figure 3.9 Two-level adaptive branch predictor [14]

And Figure 3.9 shows the two-level adaptive branch predictor [14]. This two-level branch predictor is referred to as correlation-based predictor, uses a two-dimensional table of counters, also called “pattern history table”. The table entries are two-bit counters. it can predict any repetitive sequence with any period if all n-bit sub-sequences are different. However, this repetitive sequence is not as the same as the periodically repeated Read/Write sequence. The repetitive sequence means here is periodically branch taken or not taken. Therefore, it won’t help a lot to predict the Read/Write sequence for L1-Dcache.

Since it is not easy to predict Read/Write sequence for L1-Dcache, we turn our mind to find another way to lower the delay penalty which we use dual supply voltages for L1-Dcache. Inspired by the long consecutive read sequence potential analysis in Section 3.3, we come out an idea that maybe we can aggregate Read/Write operations to generate

long consecutive read sequence as much as possible. And then switch the dual supply voltages when there comes a long consecutive read sequence.

Chapter 4: Write Aggregation Buffer

In this chapter, we will describe the write aggregation buffer we have built and analyze how will this kind write aggregation buffer impact the Read/Write operation sequence between CPU and L1-Dcache.

4.1 Write Aggregation Buffer Structure Overview

In order to reduce the L1-Dcache access times and generate long consecutive read sequence, we propose a Write Aggregation Buffer as Figure 4.1 shows.

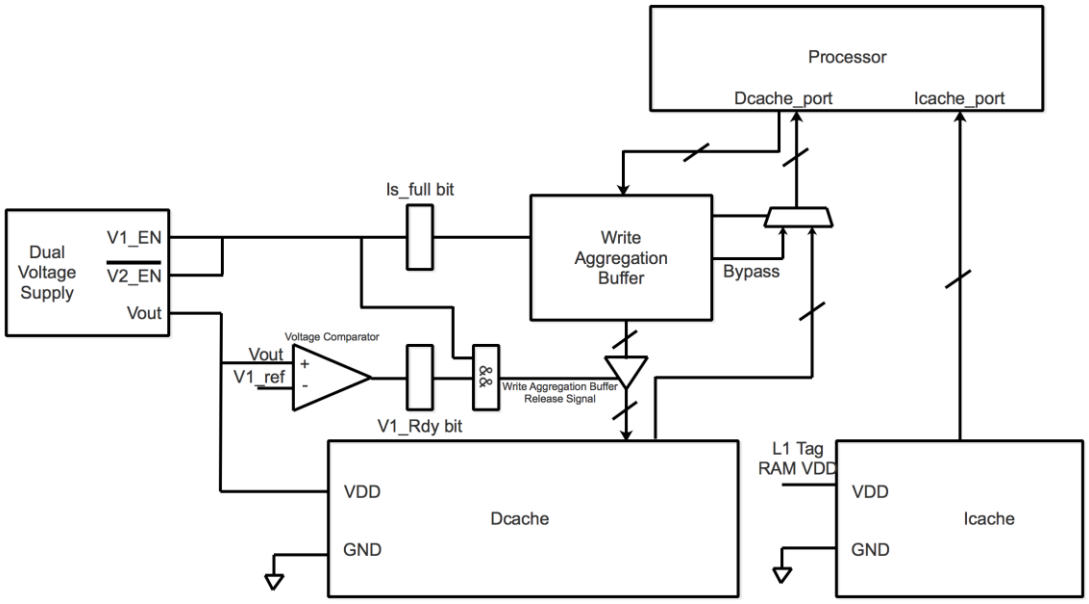

Figure 4.1 Write aggregation buffer structure

This write aggregation buffer will group the several write operations together, try to prevent the write operation interrupting the long consecutive read sequence. And It can reduce the read and write total accesses because of Read Bypass and same address Write data self-update mechanism. Also, this write aggregation buffer structure will provide the dual supply voltages switches with control logic.

From the Figure 4.1, we can conclude that how the write aggregation buffer, dual supply voltages switch L1-Dcache, L1-Icache and processor are integrated together.

For L1-Icache, the L1-Icache voltage supply is connected with L1 Tag RAM VDD which has already defined in related framework in Section 2.4. Because L1-Icache has the similar property with L1 Tag RAM. Usually, there are only read operations between CPU and L1-Icache. However, when L1-Icache read miss occurs, there will be data written from lower level cache to L1-Icache.

The processor Dcache_port has read port and write port. The read port is connected with a mux which will choose the data for L1-Dcache read port or Read Bypass of write aggregation buffer. And the control signal for this mux comes from write aggregation buffer as well. Also, there is a Is_full bit associated with write aggregation buffer, this Is_full bit is used for indicating whether this write aggregation buffer is full up or not. Meanwhile, this Is_full bit connected with dual supply voltages switch V1_EN and $\overline{\text{V2\_EN}}$ port. V1_Rdy bit is set by voltage comparator. And the control signal for write aggregation buffer release tri-state gate enable signal is Is_full bit && V1_Rdy bit.

In the following sections, we will introduce the detailed algorithm to explain how this write aggregation buffer work, including aggregate write operation together, solve Read After Write (RAW) and Write After Write (WAW) data hazard, and how can it lower the power consumption of L1-Dcache.

4.2 Dual Supply Voltages Switch Control Logic

In this section, we will introduce the algorithm of dual supply voltages switch control logic. And Table 4-1 shows the entire algorithm of it.

Table 4-1 Dual supply voltages switch algorithm

Algorithm Dual Voltage Supply Control Logic

if (*Full_up bit* == 1) **then**
  Dual Voltage Supply choose V1 as Vout
end if
else if (*Full_up* bit == 0) **then**
  Dual Voltage Supply choose V2 as Vout
end if
if (Vout == V1) **then**
  V1_Rdy bit set to 1
end if
else if (Vout != V1) **then**
  V1_Rdy bit reset
end if
if (*Full_up* bit && *V1_Rdy* bit ==1) **then**
  Release all write op. from write aggregation buffer to R/W Dcache
end if
if (*Full_up* bit ==0 or *V1_Rdy* bit ==0) **then**
  Hold all write op. in write aggregation buffer
end if

When Is_full bit is set to 1, the dual supply voltages switch will switch the supply voltage from V2 to V1. And if Is_full is reset, the supply voltages switch will switch the supply from V1 to V2. For the voltage comparator, if the value of Vout voltage equals to V1, which means Vout reached to V1, we set V1_Rdy bit to 1. If not, reset V1_Rdy bit. For the control logic of write aggregation release signal, if Is_full bit && V1_Rdy bit equals to 1, the tri-state gate enable signal will be set to 1, which means all write data in write aggregation buffer are released to L1-Dcache at that time. Or the tri-state gate will be OFF and there is no data written from write aggregation buffer to L1-Dcache. It will not only reach the destination that aggregate write operations together to prevent the scattered

write operation interrupting the consecutive read sequence, but also eliminate the delay penalty by using this voltage comparator and combinational logic.

Figure 4.2 explains how this dual supply voltages switch control logic algorithm control the voltage switches and eliminate the delay penalty.

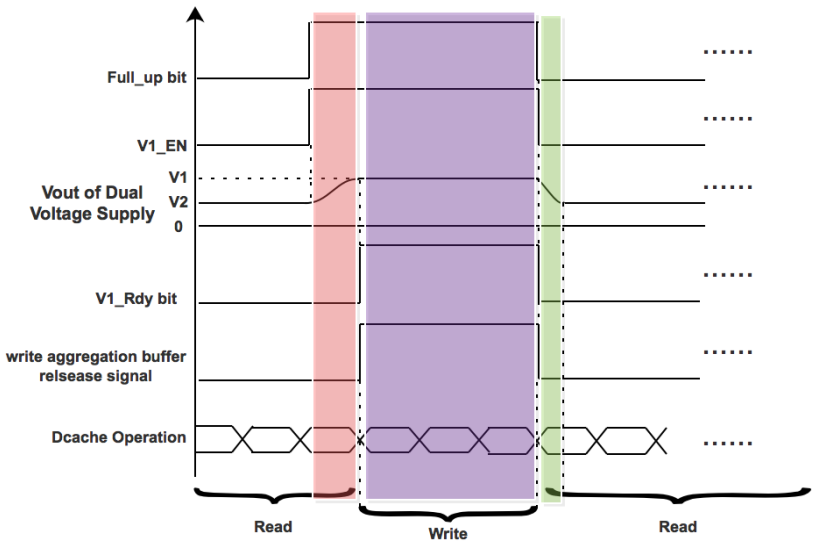

Figure 4.2 write aggregation buffer L1-Dcache

For the red zone time slot, the values/status of each register and component in write aggregation buffer structure are shown in Figure 4.3.

When the write aggregation buffer is full up, the Is_full bit is set to 1, then V1_EN is set to 1, and dual supply voltages switches switch the voltage from V2 to V1. However, there exists about 10ns delay when Vout goes up from V2 to V1. The red zone indicates the time slot that Vout hasn't reached to V1. In this time slot, V1_Rdy is always reset to 0. Because there is a voltage comparator that compare Vout with V1, If Vout =! V1, V1_Rdy bit is reset to 0. So that in the red zone time slot, the access is still read operation since V1_Rdy bit is always set to 0 and then write aggregation buffer release signal is

always set to 0 that make write operation cannot happen at that time slot. And the voltage range from V2 to V1 is still reliable for read operation. Therefore, L1-Dcache does not have to stand by to wait Vout reach to V1, there is no delay penalty here.

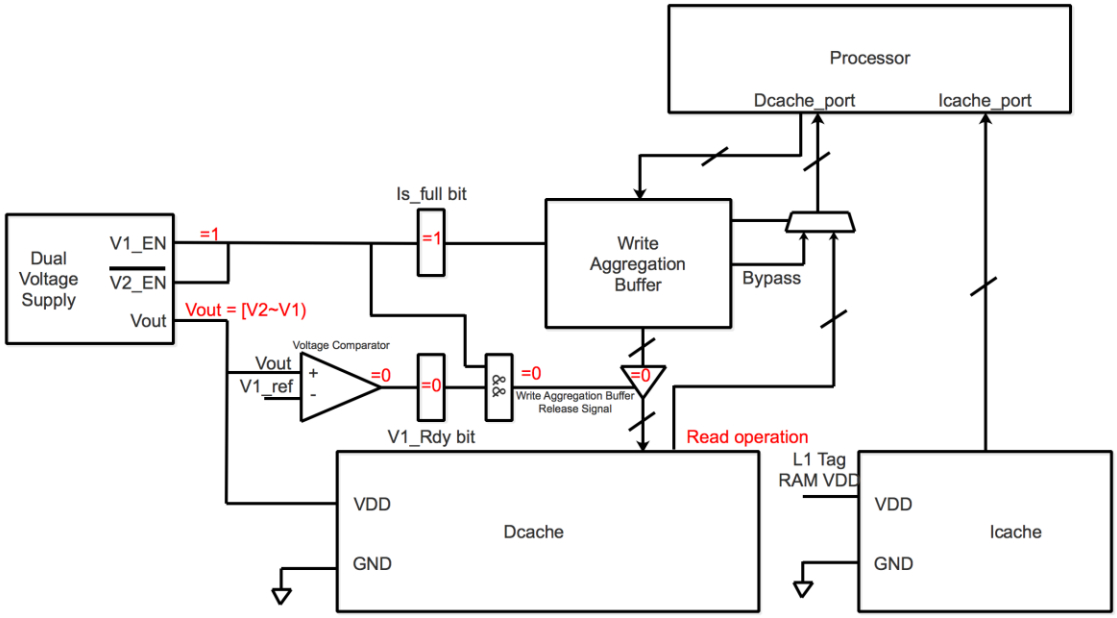

Figure 4.3 Values/status of each register or component of write aggregation buffer structure in red zone time slot

For the purple zone time slot, the values/status of each register and component in write aggregation buffer structure are shown in Figure 4.5.

When write aggregation buffer is full up and Vout reach to V1, the Is_full bit and V1_Rdy bit are set 1 simultaneously. Then Is_full bit && V1_Rdy bit equals to 1, which indicates that write aggregation buffer release signal will be set to 1. Followed by these operations, all the write operations will be released from write aggregation buffer to L1-Dcache in this time slot.

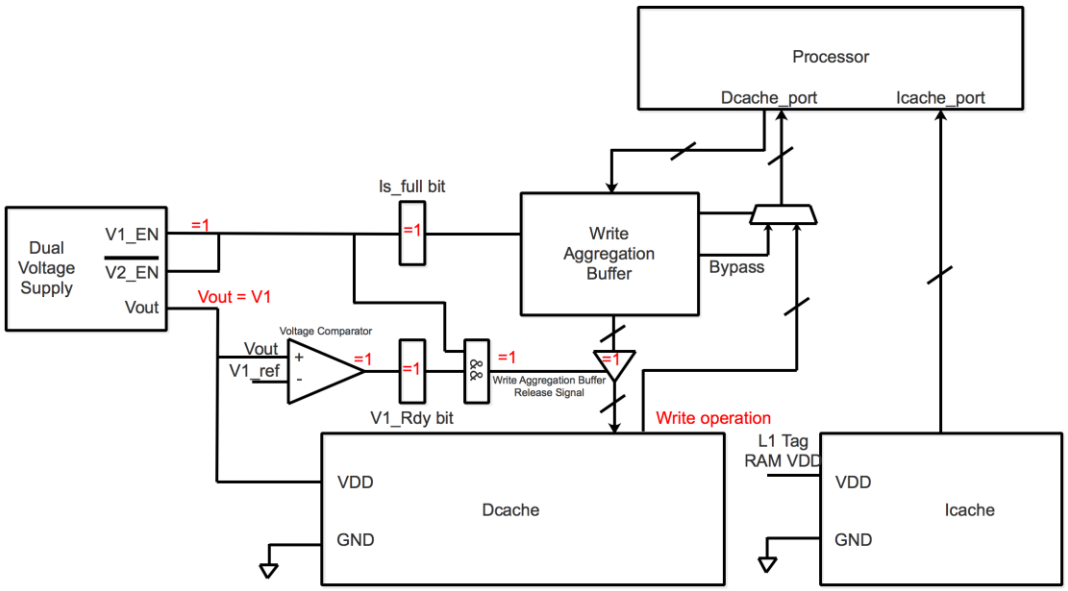

Figure 4.4 Values/status of each register or component of write aggregation buffer structure int blue zone time slot

For the green zone time slot, the values/status of each register and component in write aggregation buffer structure are shown in Figure 4.5.

When the write aggregation buffer release is done, Is_full bit is reset to 0, supply voltage switch turns Vout from V1 to V2, then the V1_Rdy bit is reset to 0. So that Is_full bit && V1_Rdy bit equals to 0, and the data access type should be read operation in this time slot. Though there is also about 10ns delay when Vout goes down from V1 to V2, the voltage range V1->V2 is still reliable for read operation. Therefore, there is no delay penalty as well in green zone time slot.

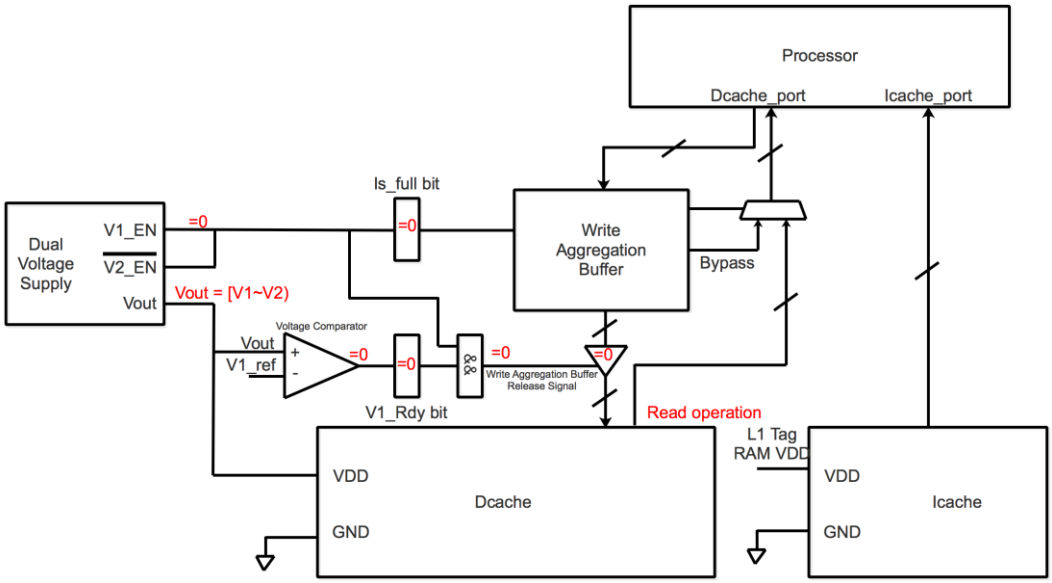

Figure 4.5 Values/status of each register or component of write aggregation buffer structure in green zone time slot

4.3 Write Aggregation Buffer Working Mechanism

In this section, we will introduce the working mechanism of write aggregation buffer.

And Table 4-2 shows the entire algorithm of how this buffer aggregate write operation.

Table 4-2 Dual supply voltages switch algorithm

Algorithm Write Aggregation Buffer

```
if (read_request) then
  //Compare read_addr with write aggregation buffer addr array: wb_addr_array
  if (read_addr == wb_addr_array[i]) then
    Bypass <-- wb_data_array[i]
    CPU <-- Bypass
  end if
  if (read_addr != wb_addr_array[i]) then
    CPU <-- R/W Dcache
  end if
end if
else if (write_request) then
  //Compare write_addr with write aggregation buffer addr array: wb_addr_array
  if (write_addr == wb_addr_array[i]) then
    wb_data_array[i] <-- write_data
  end if
  if (write_addr != wb_addr_array) then
    Fill the wb_data_array
  end if
if (Write Aggregation Buffer full up) then
  Set Full_up bit to 1
end if
else if (Write Aggregation Buffer not full up) then
  Full_up bit reset
end if
```

In this write aggregation buffer structure, if there is a read or write request, the write aggregation buffer will compare the new data address with all data address saved in buffer. For new read request, if there is a read request that has the same address as the data saved in write aggregation buffer, then this data will be read by Read Bypass. The Read Bypass will send the data that has the same address to the mux, and the mux will send this data to CPU Dcache read port. This design will prevent the Read After Write (RAW) data hazard. And for new write request, if there is a write request that has the same address as the previous data saved in write aggregation buffer, then the previous data with the same address will be updated by this new write request data. This design will prevent the Write After Write (WAW) data hazard. Figure 4.6 and Figure 4.7 shows an example 4-word size write aggregation buffer working process.

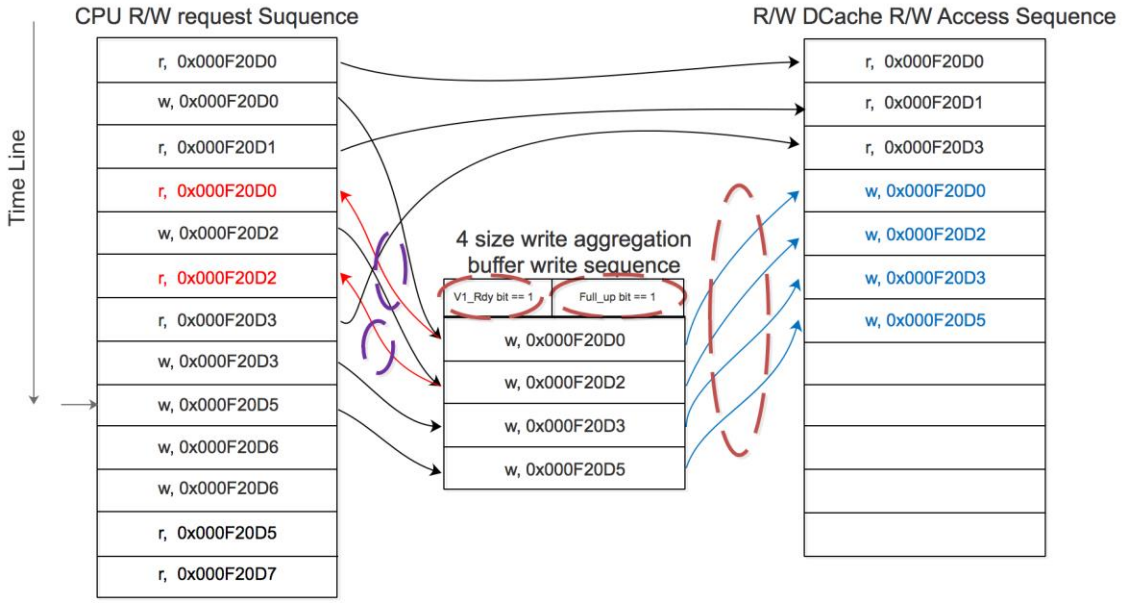

Figure 4.6 Write aggregation buffer working mechanism I

In Figure 4.6, the write aggregation buffer is just full up. In this write operation aggregate process we can see that the read request in red are two RAW data hazard. But it is solved

by Read Bypass. The data are going to be written to address 0x000F20D0 and 0x00F20D2 are read to CPU directly from write aggregation buffer instead of L1-Dcache.

And in this process, when the write aggregation buffer is full up. Is_full bit will be set to 1, and then this buffer wait for V1_Rdy bit turning to 1. Once V1_Rdy bit and Is_full bit set to 1 at the same time. The write aggregation buffer releases all write operations that have been aggregated to L1-Dcache R/W access sequence.

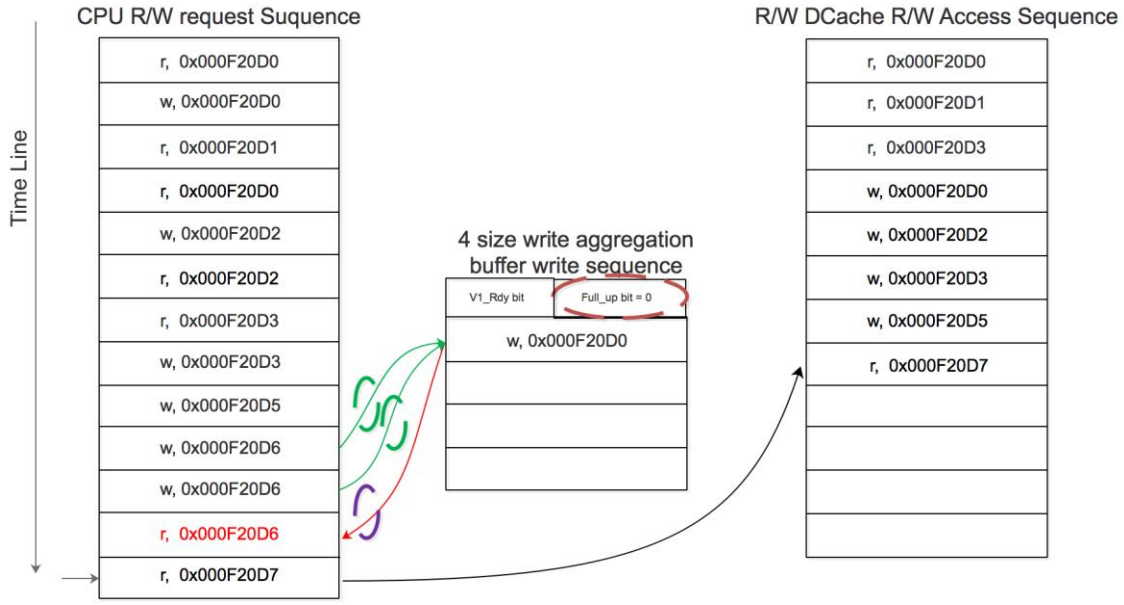

Figure 4.7 Write aggregation buffer working mechanism II

In figure 4.7, the write aggregation buffer is not full up. We see that there are two write request to write data to the same address and this address is also the same with write data already saved in write aggregation buffer (WAW data hazard). At this moment, we don't need to save these two new write data to write aggregation buffer and just let it update its previous data. When the write aggregation buffer is not full up, Is_full bit is set to 0. And we only let read request go through to L1-Dcache R/W access sequence but leave write

request to aggregation buffer. This buffer will help to generate more long consecutive read sequence.

4.3 Result Analysis

In this section, we implement the 32-word and 64-word size write aggregation buffer to parse the cache trace when we use Gem5 to simulate the cache behavior of SPEC2000 swim and applu benchmark and get the CDF of consecutive read sequence.

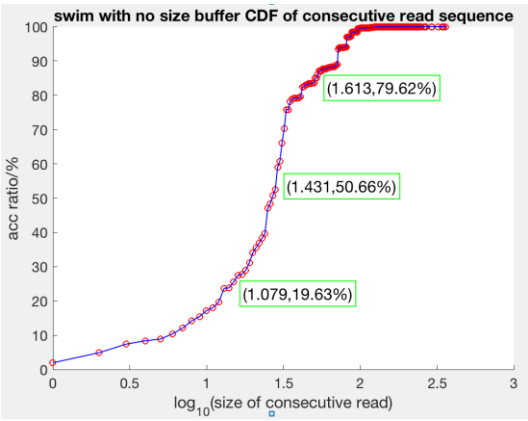

Figure 4.8 swim CDF of consecutive read sequence without write aggregation buffer

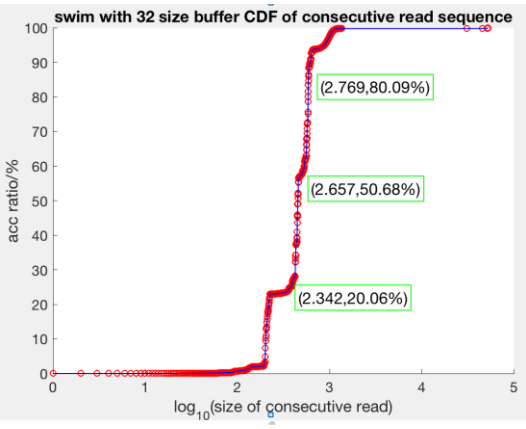

Figure 4.9 swim CDF of consecutive read sequence with 32-word size write aggregation buffer

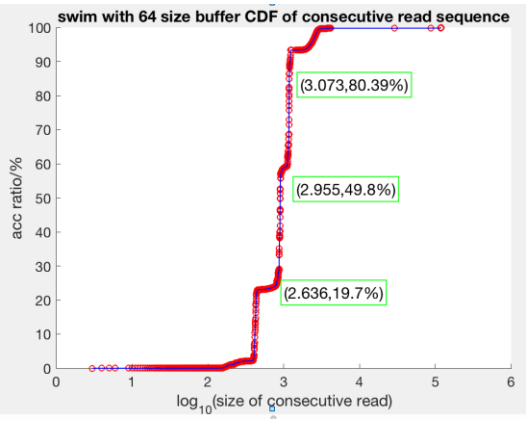

Figure 4.10 swim CDF of consecutive read sequence with 64-word size write aggregation buffer

In Figure 4.8, the consecutive read sequence size corresponding to 20%, 50% and 80% are $10^{1.079}$, $10^{1.431}$ and $10^{1.613}$. In Figure 4.9, these three values are $10^{2.342}$,$10^{2.657}$ and $10^{2.769}$. And in Figure 4.10, these three values go up to $10^{2.638}$ $10^{2.955}$ and $10^{3.073}$. The consecutive read sequence size corresponding to these three percentage are increasing over twice. And in applu simulation results shown in Figure 4.11, Figure 4.12, and Figure 4.13. These sizes are approximately increasing four times. Therefore, we conclude that the write aggregation buffer indeed helps to generate long consecutive read sequence, which will make it have more opportunities to save power using dual supply voltages.

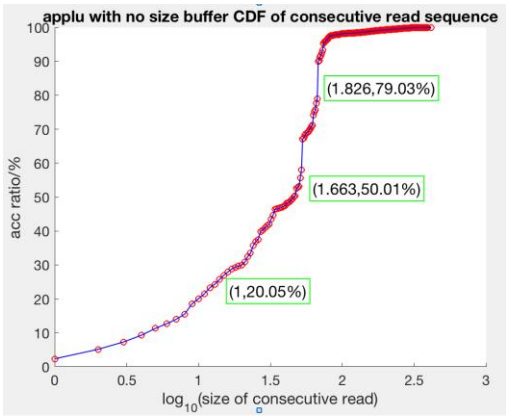

Figure 4.11 applu CDF of consecutive read sequence without write aggregation buffer

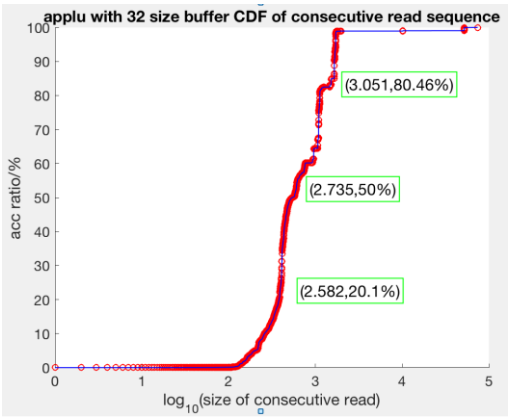

Figure 4.12 applu CDF of consecutive read sequence with 32-word size write aggregation buffer

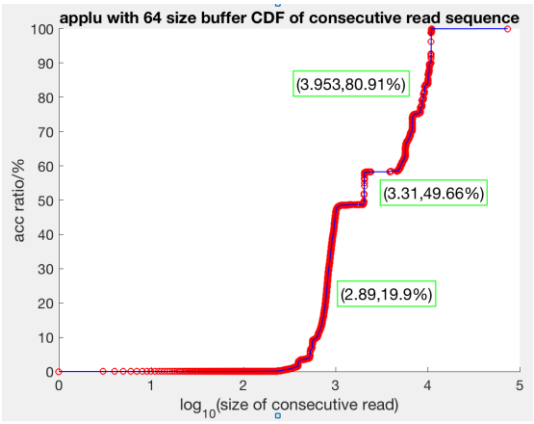

Figure 4.13 applu CDF of consecutive read sequence with 64-word size write aggregation buffer

Chapter 5: Case Study

In this chapter, we evaluate the power reduction opportunity using dual supply voltages in ASIC chip especially for machine learning accelerator.

5.1 DianNao Machine Learning Accelerator Case

DianNao machine learning accelerator is a small-footprint high-throughput accelerator for ubiquitous machine learning. table shows the power breakdown of DianNao accelerator. From the table, we could notice that the Synaptic buffer (SB), input buffer for input neurons (NBin) and output buffer for output neurons (NBout) parts take a large percent of total power. And we already know these buffers are built by SRAM from previous reported [2].

Table 5-1 Characteristics of accelerator and breakdown by component type (fist 5 lines), and functional block (last 7 lines) [2]

Component or Block	Area in μm^2	(%)	Power in mW	(%)	Critical path in ns
ACCELERATOR	3,023,077		485		1.02
Combinational	608,842	(20.14%)	89	(18.41%)	
Memory	1,158,000	(38.31%)	177	(36.59%)	
Registers	375,882	(12.43%)	86	(17.84%)	
Clock network	68,721	(2.27%)	132	(27.16%)	
Filler cell	811,632	(26.85%)			
SB	1,153,814	(38.17%)	105	(22.65%)	
NBin	427,992	(14.16%)	91	(19.76%)	
NBout	433,906	(14.35%)	92	(19.97%)	
NFU	846,563	(28.00%)	132	(27.22%)	
CP	141,809	(5.69%)	31	(6.39%)	
AXIMUX	9,767	(0.32%)	8	(2.65%)	
Other	9,226	(0.31%)	26	(5.36%)	

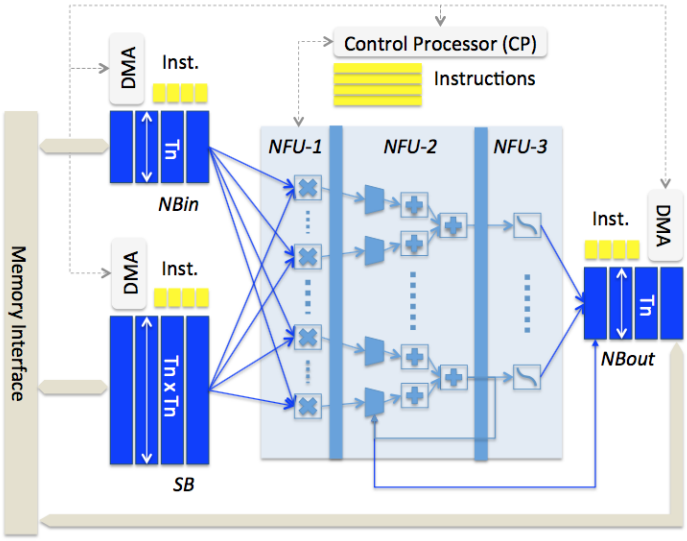

Figure 5.1 DianNao accelerator architecture [2]

Figure 5.1 shows the architecture of DianNao accelerator. Combine the architecture of this accelerator, the dataset they used for this machine learning accelerator and the control instruction table. We can approximate the power saving shown in Figure 5.2 of SB, NBin and NBout using dual supply voltages.

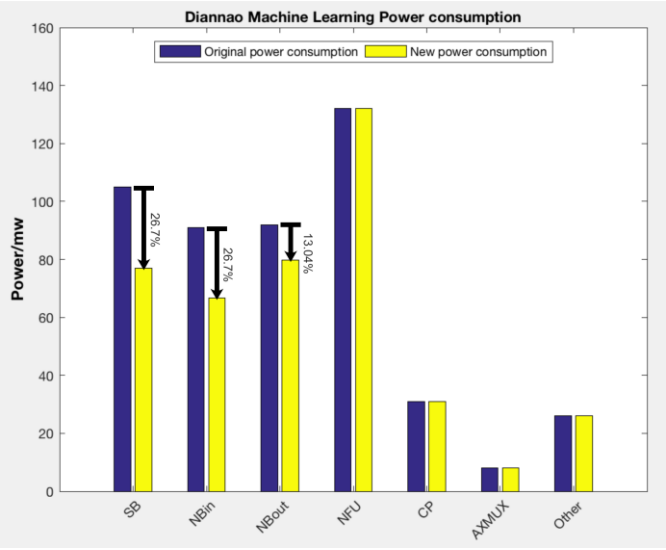

Figure 5.2 Approximate power saving using dual supply voltages in DianNao accelerator

SB save about 26.7 % power, NBin reduce the power about 26.7 and NBout slightly reduce the power by 13.04%. Because NBout are used for the buffer to save the intermediate result of this machine learning network.

5.2 Generous Purpose Machine Learning ASIC Chip Case

Investigating the architecture of different machine learning AISC chip, such as Eyeriss Accelerator [15] and Google TPU [16] in Figure 5.3 and Figure 5.4.

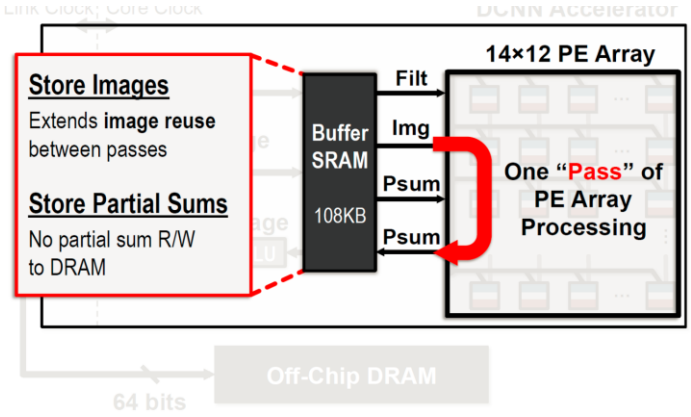

Figure 5.3 Eyeriss accelerator [15]

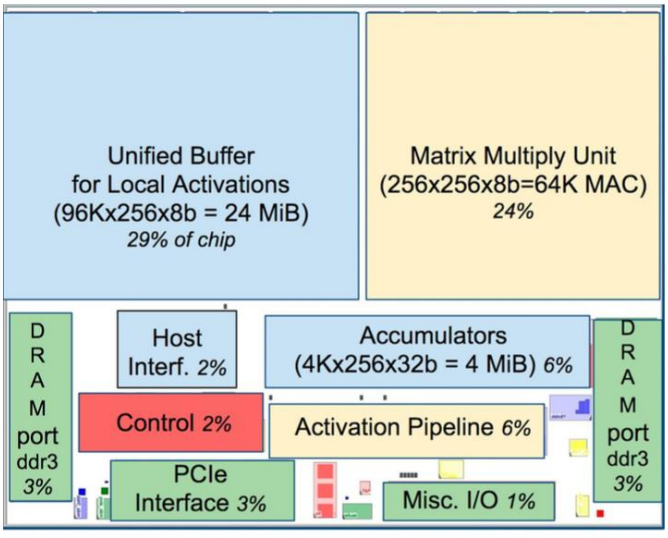

Figure 5.4 Google tensor processing unit [16]

In Eyeriss work, we get the data movement energy cost that is shown in Figure 5.5. If it uses Data movement energy cost of ALU as the reference, Buffer (SRAM structure) takes 10x power. Therefore, there is a large energy saving space in this buffer if we implement the dual supply voltages switch work in the Eyeriss ASIC chip. Since the SRAM buffer is written into 14×12 input pixels and the kernel size weight/bias data from DRAM at one time, and then PE array reads these data from SRAM buffer by several cycles. During this process, the partial sum results will be written to SRAM buffer when every row multiplication calculation is done. Also, these previous partial sum results will be read from SRAM buffer when it is feed another chunk of input pixels and weight/bias data.

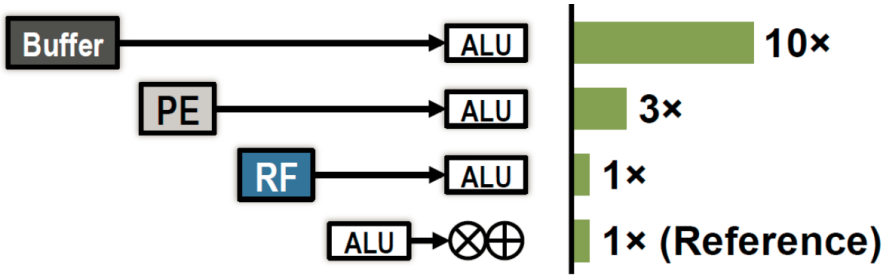

Figure 5.5 on chip data movement expense in Eyeriss [15]

From above analysis of data movement, we know that the read sequence or write sequence of SRAM buffer is already known and packaged together to several read operation chunks and write operation chunks. Therefore, there is a best opportunity to implement dual supply voltages switch SRAM to Eyeriss ASIC chip.

Meanwhile, from Norman et. al. [16], they have stated that the large SRAM uses much more power than arithmetic, the matrix unit uses systolic execution to save energy by reducing read and writes of the Unified buffer. Actually, the strategy they used is similar to what Eyeriss has done. They optimized the data movement between buffer (SRAM

structure) and arithmetic units, which means the read and write operation sequence is already settled down and grouped together to the large chunks.

Therefore, we conclude that in the ASIC Chip design, especially for accelerator design or some other machine learning processing unit design. The data movement of that design is usually already known and well optimized to long consecutive read and write chunks. Therefore, the dual supply voltages switches SRAM framework has good viability for the custom machine learning ASIC accelerator design.

Chapter 6: Future work

The write aggregation buffer is similar to store queue in CPU and is also very similar to store buffer in Intel Haswell Arch. So that we would like to merge them together to reduce the additional power consuming. Because build a new buffer architecture is a kind of trade off. When we build a new component, it will bring other power cost, such as the combinational logic power consuming. Therefore, merging design should be a best way to balance this problem.

Secondly, our write aggregation buffer will break the spatial locality in some way. But it is a kind of tradeoff between power and performance. We plan to replay the new cache trace with write aggregation buffer to gem5 replay interface and analyze how those parameters will be impacted if we use this kind of write aggregation.

Finally, we are still expecting that find some solid clue to predict read and write sequence. Maybe it is not an easy work. However, if we reach to this destination, we will have more opportunity to reduce the power using dual supply voltages.

Chapter 7: Conclusions

In this thesis, we reduce the Cache power consumption using dual power supply, which makes Caches works at lower voltages in read status and works at high voltage in write status.

We use gem5 and McPAT to analyze the leakage and dynamic power breakdown for CPU, and conclude that cache consume a large part of total power consumption.

We use gem5 to do the cache trace of L1Dcache based on different type benchmarks. Conclude that different benchmarks have different cache trace characterization. Machine learning benchmarks are usually memory-intensive application and SPEC2000 are the computation-intensive.

We propose write aggregation buffer to group the write operations together, in order to generate long consecutive read sequence, which will increase the power reduction opportunity using dual voltage supply.

We do case study to learn that dual voltage supply should reduce the power consumption of SRAM in ASIC chip more efficiently.

References

[1] http://rlpvlsi.ece.virginia.edu/node/368

[2] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam. DianNao: A small-footprint high-throughput accelerator for ubiquitous machine-learning. In Intl. Conf. on Architectural Support for Programming Languages and Operating Systems, ASPLOS '14, pages 269–284, 2014.

[3] http://news.softpedia.com/news/Nvidia-s-Project-Denver-Could-Pack-8-ARM-Cores-and-256-CUDA-Cores-212288.shtml

[4] http://thecodeartist.blogspot.com/2011/12/why-readmostly-does-not-work-as-it.html

[5] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen and N. P. Jouppi, "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures," *MICRO, New York, NY, 2009, pp. 469-480.*

[6] http://gem5.org/Main_Page

[7] https://www.spec.org/cpu2000/

[8] S. Thomas et al., "CortexSuite: A synthetic brain benchmark suite", Proc. Int. Symp. Workload Characterization, pp. 76-79, Oct. 2014.
[9] http://www.iue.tuwien.ac.at/phd/entner/node34.html

[10] D.Yan: Cache power using multiple voltage supplies to exploit cache Read/Write asymmetry, School of Engineering and Applied Science of Washington University in St. Louis.

[11] A. Carlson, Z. Guo, S. Balasubramanian, R. Zlatanovici, T. J. King Liu, and B. Nikolic, "SRAM read/write margin enhancements using FinFETs," IEEE Trans. VLSI, vol. 18, no. 6, pp. 887–900, Jun. 2010.

[12] Miller, Timothy N., et al. "Booster: Reactive core acceleration for mitigating the effects of process variation and application imbalance in low-voltage chips", in Proceedings of the 18th IEEE International Symposium on High Performance Computer Architecture, 2012.

[13] https://en.wikipedia.org/wiki/Cache_prefetching

[14] https://en.wikipedia.org/wiki/Branch_predictor

[15] Yu-Hsin Chen, "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks", IEEE Journal of Solid-State Circuits, Vol.52, no.1, Jan.2017

[16] Norman P. Jouppi, "In-Datacenter performance Analysis of a Tensor Processing Unit", ISCA, Tornoto, Canada, Jun26, 2017

[17] J. Schandy, L. Steinfeld, and F. Silveira, “Average Power Consumption Breakdown of Wireless Sensor Network Nodes Using IPv6 over LLNs,” in 2015 International Conference on Distributed Computing in Sensor Systems (DCOSS), Jun. 2015, pp. 242–247.

[18] Carroll, A. and Heiser, G. 2010. An analysis of power consumption in a smartphone. In Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference (Boston, MA, June 23 - 25, 2010). USENIX Association, Berkeley, CA, 21-21.

[19] Siming, “6-T SRAM Read and Write Failure Rate Characterization for Low Voltage Operations”, report.

[20] Nahid Rahman and B.P.Singh, “Static-Noise-Margin Analysis of Conventional 6T SRAM Cell at 45nm Technology”, International Journal of Computer Application (0975-8887), vol. 66, no. 20, March 2013.