國 立 交 通 大 學

電機與控制工程學系

碩士論文

功率感知資料匯流排編碼解碼器設計

Design of Power Aware Data Bus Codec

研究生：黃德瑋

指導教授：林進燈 教授

陳右穎 教授

中華民國九十六年七月

功率感知資料匯流排編碼解碼器設計

Design of Power Aware Data Bus Codec

研 究 生：黃德瑋　　Student：De-Wei Huang

指導教授：林進燈 教授　　Advisor：Dr. Chin-Teng Lin

陳右穎 教授　　Dr. You-Yeng Chen

國立交通大學

電機與控制工程學系

碩士論文

A Thesis
Submitted to Institute of Electrical and Control Engineering
College of Electrical and Computer Engineering
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in
Electrical and Control Engineering

July 2007

Hsinchu, Taiwan, Republic of China

中華民國九十六年七月

功率感知資料匯流排編碼解碼器設計

Design of Power Aware Data Bus Codec

學生：黃德瑋　　指導教授：林進燈 博士

陳右穎 博士

國立交通大學電機與控制工程研究所

中文摘要

本論文提出在匯流排傳輸上面，設計一個功率感知資料匯流排編碼解碼器，來降低 transition activity，進而達到降低功耗輸出的效果。在 8 位元寬度以及外部負載電容 50 pF 環境下模擬結果，分別與編碼前及 RSH 方法相比較可降低 23%和 6%功率消耗，其設計特色在於：(1)編碼解碼端不需要花費龐大硬體成本以及處理時間，便可達到迅速傳輸資料以及有效率降低功耗的目的；(2)針對不同應用能自動挑選來做最合適的編碼處理。經由測試結果，只需要額外增加 6%硬體成本，在多媒體資料傳輸，平均可降低 20%左右動態功率；在 DCT、FIR 程式中，平均可降低 50~60%左右動態功率。

再者，我們將此低功耗匯流排整合至在已開發的嵌入式RISC/DSP單核心處理器內，針對處理器系統架構上面，加入數位低功耗設計，有效率的降低功率消耗，期望能在效能以及功耗上達到一個平衡點。此設計採用TSMC 0.18μm 製程，晶片製作面積約 2.11x2.11 mm^2，預估最大操作頻率在 100MHz，功率消耗約 16mW。

Design of Power Aware Data Bus Codec

Student：De-Wei Huang　　Advisor：Dr. Chin-Teng Lin

Dr. You-Yeng Chen

Department of Electrical and Control Engineering

National Chiao-Tung University

Abstract

In this thesis, we propose a power-aware codec scheme to reduce transition activity for data bus design. The low power data bus codec consisting of transparent, inverter, XOR, and XNOR module can lead to 23 % & 6 % power reduction compared with the un-coding and R-S-H's methods under the 8-bit width and the 50 pF capacitance loading. The main features of this codec design are: (1) codec can save 68% area overhead compared with R-S-H's design and (2) codec can adaptively choose the optimal encoding scheme for different kinds of data types due to versatile applications. From the FIR and DCT benchmark simulations, the power can be reduced to 50%~60% on average.

Furthermore, we integrate this data bus codec into a RISC/DSP unit-core processor with the tradeoff between cost and power. The chip fabricated in TSMC 0.18μm CMOS technology process with the total area of $2.11\times2.11mm^2$ and has power consumption of 16mW at 100MHz with 1.8V supply voltage.

誌謝

兩年的研究所生涯隨著論文的完成劃上了句號，這兩年間，要感謝許多人的鼓勵和幫忙，使我獲得充實的專業能力並順利完成研究所的學業。

首先要感謝的是我的指導教授-林進燈老師。林老師是國內十分傑出的一位教授，在不同領域內都有相當好的研究成果。感謝老師提供了很理想的研究環境及正確的引導，使我在研究上非常順利。在老師悉心的指導下，讓我學習到解決問題的能力及做研究應有的態度。

另外，最感謝資工系范倫達教授及鐘仁峰學長的教導，尤其是面臨畢業主題方向模糊的壓力時，范教授在這上面給予我相當大的助力，而教授親切的態度及學識的也讓我在討論論文時感到輕鬆而無壓力，獲益良多。此外在實驗室中，不管大小疑難雜症，常常去請教仁峰學長，非常感謝學長不厭其煩地教導，使我增進了對積體電路設計上的專業知識，開拓了我的視野。也感謝實驗室所有的夥伴，經翔學長、紹航學長、峻谷學長、家昇學長、庭緯學長、有德學長、笑容甜美頭髮捲捲的美女靜瑩、酷酷運動全能的智文、開朗運動全能的俊傑、可愛熱情的正妹林玫、翰林大學士肇廷以及讓我論文不能早點寫完的學弟妹們等，感謝大家在研究上的互相扶持及鼓勵。

也感謝我的爸爸、媽媽、奶奶、哥哥，你們的支持一直是我最溫暖的後盾；你們的鼓勵是我信心的來源。

Table of Contents

List of Figures

List of Table

Chapter 1
Introduction

1.1 Brief Introduction

In 3C integration era, the mobile phone does not only communicate with people but also has various functions like digital camera, MP3 player, games, and etc. Therefore, the multi-functions mobile phone just can acquire favor of consumers in the information market.

However, when the demand of performance and functions of the mobile phone increases, the power consumption would be an important design issue. Most of companies not only seek for high performance and low cost, but also focus on low power design.

In other words, low power is a primary consideration to System on Chip (SOC) design, especially for handheld devices due to the limited battery life. In order to accomplish such challenging tasks, many design techniques such as multi-Vth design techniques [1][2], dynamic voltage scaling [3][4], gated clock [5], and low-power on-chip memory architecture [6] have been proposed to reduce both dynamic power and leakage power However, those design techniques require advanced design process to reach the low power goal.

In the processor, it becomes increasingly limited by memory performance and system power consumption [7]. The power associated with off-chip accesses can dominate the overall power budget. The memory power problem is even more acute for processors that possess memory intensive access patterns and require streaming serial memory access that tends to exhibit low temporal locality.

In terms of reducing memory power, one approach is to consider how optimally to schedule off-chip accesses. The capacitance associated with the external bus is much larger than the internal node capacitance inside a microprocessor. [7] For example, a low-power embedded microprocessor system like an Analog Devices ADSP-BF533 running at 500 MHz consumes about 374 mW on average during normal execution. Assuming a 3.65 V supply voltage and 133 MHz bus frequency, the average external power consumed is around 170 mW, which accounts for approximately 30% of the overall system power dissipation. One factor affecting the capacitance on external bus power is the bus width. For example, the power dissipation on 16-bit bus is larger than 30% on 8-bit bus. As a consequence, the design target like MP3 player, PDA and mobile phone always use low bit width bus instead of the high bit width bus.

Recently, R-S-H proposed codec scheme to reduce power consumption for data and address buses. However, the table size is proportional to bit width in [16]. That means that while data width is larger, more power consumption certainly be induced. In this thesis, we are motivated to design a power-aware data bus codec which can reduce dynamic power for data transmission. This power-aware codec is composed of transparent, inverter, XOR, and XNOR modules. We use the audio, image, EEG,, random, and specific data to verify the codec characteristics via simulation results and compare with other encoding schemes. In terms of codec implementation, a RISC/DSP unit-core processor that integrates the proposed codec and low power cache controller design is used for verification. The chip has been fabricated in TSMC 0.18μm CMOS technology with the total area of $2.11 \times 2.11 mm^2$. The maximum clock frequency runs at 100MHz with a single 1.8V supply voltage.

The proposed codec design has following features:

(1)Low cost

Codec does not need large hardware cost (just have 5% gate counts of total processor) and one cycle processing time penalty.

(2)Low power

In the result of 8-bit simulation, our proposal has 23 % dynamic power reduction in average on bus. For DSP function such as DCT and FIR, our proposal has 50-60% dynamic power reduction on bus. For power estimation, the proposed encoder and decoder only have 0.8mW in PrimePower simulation.

(3)Awareness

The general encoder is usually suitable for several specific data stream or data property. For instance, Bus-Invert encoding scheme can only be used to acute data variability. Our proposed method can compare the result of all encoding functions in encoder and adaptively choose the optimal encoding scheme for different kinds of data types due to versatile applications.

1.2 Organization of the Thesis

In this thesis, the organization is as follows. In Chapter 1, we give a brief introduction for low power design. In Chapter 2, we propose a new power-aware codec design for data bus. The integrated processor including our proposed bus codec, and tool chains will be demonstrated in Chapter 3. The processor layout and simulated result are shown in Chapter 4. Finally, conclusions and future work are remarked in the last Chapter.

Chapter 2
Power-Aware Data Bus Codec

We would present an adaptive data bus codec including proposal, architecture, and performance comparison with the features of low power, low cost, and awareness.

2.1 Motivation

As we know, there are two major sources of power dissipation in digital CMOS circuits, which are summarized as follows[8][9]

$$P = a \times C \times V^2 \times f + I_{leakage} \times V, \quad (2\text{-}1)$$

Where *P, C,* α, *V, f* denote power consumption, capacitance, transition activity, supply voltage, and clock frequency, respectively. The first and second terms represent the dynamic power and leakage power, respectively. In the second term, leakage current that can be arisen from substrate injection and sub-threshold effects is primarily determined by the fabrication technology.

For the reduction of dynamic power, the main design principle is to minimize the values of *V, C, f* and α in Eq. (2-1) [10]. Among the four parameters, supply voltage *V* that has a quadratic effect and capacitance C are very efficient ways of decreasing the power dissipation. However, for CMOS circuits, the designers usually decrease *V* and *C* in layout level. For larger digital circuits and systems, decreasing *V* and *C* is an annoying problem in cell-based design. On the other hand, lowering the transition activity is a very promising way to reduce the power consumption in cell-based design.

Generally speaking, the percentage of power dissipation on bus is in the range of 10% and 80% for microprocessor. The category of bus is external bus and internal bus. External bus includes external memory data transmission and I/O data transmission. Internal bus includes internal memory, cache, and IP data transmission. The power dissipation in external busses usually is larger than that of internal busses by hundred times [8]. Thus, we are motivated to solve this critical power problem of data bus in architecture and logic level. In this paper, we propose a power-aware encoder and decoder to compress the data transition activity α, and thus the power can be saved.

There are four properties in bus stream [11] discussed as follows.

(1) Instruction address stream: Instructions addresses are often consecutive. As a result, instruction address stream is very predictable.

(2) Data address stream: Data access may be consecutive while accessing arrays; otherwise, the data address stream is random. Although data addresses are less predictable, they still follow the principles of spatial and temporal locality.

(3) Instruction stream: Most ISAs (Instruction Set Architecture) exhibit some regularity and instructions can be partitioned into fixed-location fields. As a result, Instruction stream is predictable by fixed-location fields.

(4) Data stream: The sequence is not predictable. The values vary irregularly with different kinds of applications and different kinds of algorithms.

The above properties in bus stream have been widely applied to three off-the-shelf computer architectures.

(a) Harvard architecture with four busses:

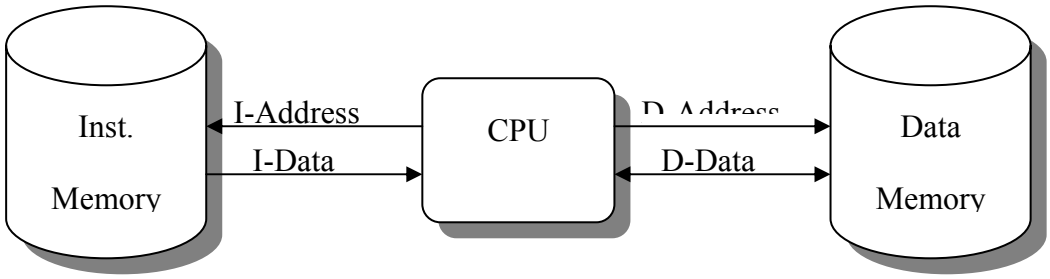


Fig. 2-1. Harvard architecture with four busses.

Harvard architecture is a computer architecture with physically separate storage and signal pathways for instructions and data. Each address bus and data bus is only for instruction memory or data memory. As a result, each stream has independent bus and been easily controlled.

(b) von Neumann architecture with two busses:

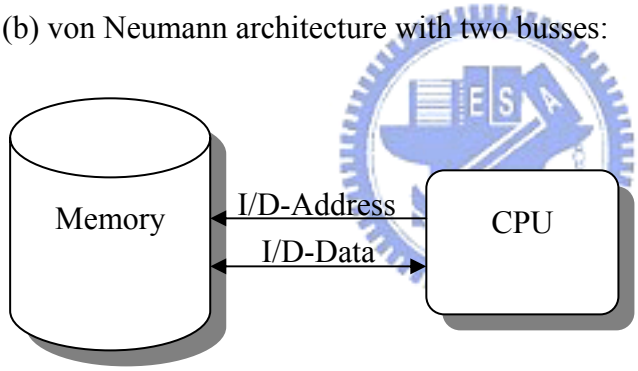


Fig. 2-2. von Neumann architecture with two busses.

The von Neumann architecture is a computer architecture that uses a single storage structure to hold both instructions and data. Instruction address stream and Data address stream are set on the same bus. Instruction stream and Data stream is so on.

(c) von Neumann architecture with one bus:

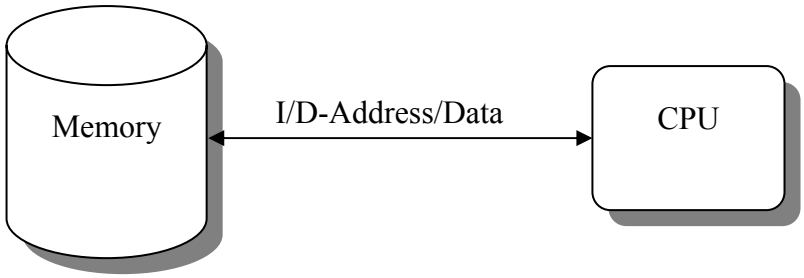


Fig. 2-3 von Neumann architecture with one bus.

All streams are running on the same bus. On this bus, it needs more signals to control stream operations.

2.2 Related Works

In this section, we would introduce the relative researches of low power bus encoding. From the beginning, we will have a brief subsection about Bus-Invert encoding. Bus-Invert encoding [12] is a traditional encoding at the early low power designs. It has the advantage of low cost hardware implementation. In Section 2.2.2, we will introduce Zero-Transition Activity encoding [15]. In Section 2.2.3, we will show a coding framework for low power address and data busses [16].

2.2.1 Bus-Invert Bus Encoding

We will consider the activity on a typical data bus to be characterized by a random uniformly distributed sequence of values [13][14]. The assumption of random uniformly distributed inputs is also conveniently made by most of the statistical power estimation methods. With this assumption for any given time-slot the data on an n-bit wide bus can be any of 2^n possible values with equal probability. The average number of transitions per time slot will be $n/2$. For example on an eight-bit bus there will be

an average of 4 transitions per time-slot or 0.5 transitions per bus-line per time-slot. When all the bus-lines toggle at the same time (the probability of this happening in any time-slot is $1/2^n$) there will be a maximum of n transitions in a time-slot and thus the worst power dissipation is proportional with n.

The Bus-Invert method [12] proposed here uses one extra control bit called invert. By convention then invert = 0 the bus value will equal the data value. When invert = 1 the bus value will be inverted. The worst power dissipation can then be decreased by half by coding the bus as follows (Bus-Invert method):

(1) Compute the Hamming distance (the number of bits in which they differ) between the present bus value and the last data value.

(2) If the Hamming distance is larger than $n/2$, set invert = 1 and make the present bus value equal to the inverted present data value.

(3) Otherwise let invert = 0 and let the present bus value equal to the present data value.

(4) At the decoder side, the contents of the bus must be conditionally inverted according to the invert line. In any case the value of invert must be transmitted over the bus (the method increases the number of bus lines from n to $n + 1$).

The Bus-Invert encoding has the advantage of that the maximum number of transitions per time-slot is reduced from n to $n/2$. Therefore the worst power dissipation for the bus is reduced by half. Fig. 2-4 shows the 16 bit data sequence using the Bus-Invert encoding in order to decrease the number of transitions.

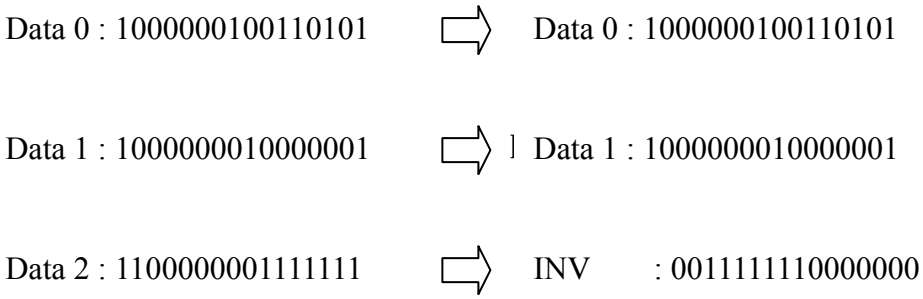


Fig. 2-4. Bus-Invert Encoding.

We can see the Hamming distance between the data 0 and data 1 is smaller than 8, so invert =0. However the Hamming distance between the data 1 and data 2 is bigger than 8, so invert =1 and data 2 is inverted.

2.2.2 Zero-Transition Activity Encoding

The scheme we propose is related to the Bus-Invert encoding, both Bus-Invert encoding [12] and Zero-Transition Activity encoding [15] rely on the addition of a redundant line to reduce the total number of transitions that may happen when streams of patterns are transmitted over the bus. For example, Bus-Invert encoding use a redundant line INV that control data encoding for power reduction.

In Zero-Transition Activity encoding scheme, called the T0 code, is that of avoiding the transfer of consecutive addresses on the bus by using a redundant line, INC, to transfer to the receiving sub-system the information on the sequentially of the addresses. When two addresses in the stream to be transmitted are consecutive, the INC line is set to 1, the address bus lines are frozen (to avoid unnecessary switch activities), and the new address is computed directly by the receiver. On the other hand, when two addresses are not consecutive, the INC line is driven to 0 and the bus lines operate normally.

If all addresses of the ideal stream are consecutive, the INC line is always high, and the bus lines always have no transition. Consequently, the switch activity of our code is zero transitions per emitted consecutive address.

More formally, our Zero-Transition Activity encoding (T0 code) scheme can be described as follows Eq. (2-2):

$$(B(t), INC(t)) = \begin{cases} (B(t-1), 1) & ;if\ t > 0\ and\ b(t) = b(t-1) + S \\ (b(t), 0); & otherwise \end{cases}, \qquad (2\text{-}2)$$

where *B(t)* is the value on the encoded bus lines at time t ,*INC(t)* is the additional bus line, *b(t)* is the address value at time t and S is a constant of increase, that we call stride. The corresponding decoding scheme can formally define as follows (2-3):

$$b(t) = \begin{cases} b(t-1) + S; & if\ INC = 1\ and\ t > 0 \\ B(t); & if\ INC = 0 \end{cases}, \qquad (2\text{-}3)$$

Notice that the T0 code retains its zero-transition property even if the addresses are incremented by a constant stride equal to a constant of two (as it is often the case for practical machines which are byte addressable, but that are able to access data or instructions aligned at word boundaries).

We take an example shows Zero-Transition Activity encoding following above equations (2-2) (2-3). Table 2-1 lists the switch activities with original data transfer, we can find the total transitions are 10 from cycle 0 to cycle 6. Table 2-2 lists the data transmission with Zero-Transition Activity encoding. At a given clock cycle *t* (*t* = [1,7] for table 2-2), the encoder computes the incremented address of cycle t and compares it to the address generated at cycle *t - 1*. If the incremented old *(t - 1)* address and the new (*t*) address are equal, the INC line is raised, and the old address is left on the bus. The encoder/decoder architecture is shown on Fig.2-5. The incrementer can be programmable, to be able to flexibly define the constant increment S. In Table 2-2, S is defined as 1.

The decoder architecture is simple. At any given clock cycle, the last cycle's

address is incremented. If the INC line is high, the old incremented value is used for addressing; otherwise, the value coming from the bus lines is selected. Finally, we can find the total transitions become 4. Zero-Transition Activity encoding make address value on bus be frozen when address is consecutive so that power dissipation will be reduced efficiently.

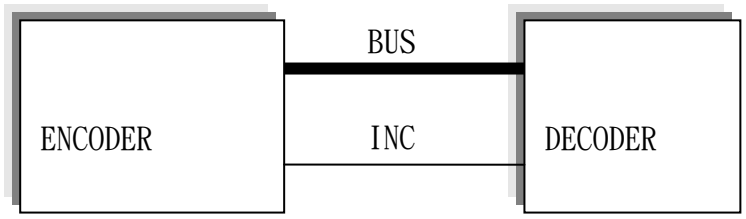


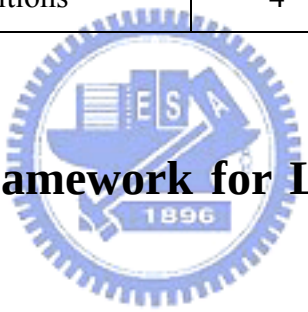
Fig. 2-5. Zero-Transition Activity encoder/decoder.

Table 2-1 Without Zero-Transition Activity Encoding

Continuous bus address transition		
cycle	Address to be transfer	Address on BUS
0	00000000	00000000
1	00000001	00000001
2	00000010	00000010
3	00000011	00000011
4	00001000	00001000
5	00001001	00001001
6	00001010	00001010
Total Transitions		10

Table 2-2 With Zero-Transition Activity Encoding

Continuous bus address transition			
cycle	Address to be transfer	Address on BUS	INC
0	00000000	00000000	0
1	00000001	frozen	1
2	00000010	frozen	1
3	00000011	frozen	1
4	00001000	00001000	0
5	00001001	frozen	1
6	00001010	frozen	1
Total Transitions		4	



2.2.3 A Coding Framework for Low Power Address and Data Busses

In this section, we present a source-coding framework for describing low power encoding schemes and then employ the framework to develop new encoding schemes [16]. In the framework proposed here, a data source is processed first by a decorrelating function f_1. Next, a variant of entropy coding function f_2 is employed, which reduces the transition activity.

Signal samples have higher probability of occurrence are assigned code words with fewer ON bits. This scheme is suited for the power dissipation depends on the number of ON bits. In VLSI systems, however, power dissipation depends on the number of transitions rather than thee number of ON bits.

A general communication system in Fig. 2-6 consists of a source coder, a channel coder, a noisy channel, a channel decoder, and a source decoder. The source coder (decoder) compresses (decompresses) the input data so that the number of bits required in the representation of the source is minimized. While the source coder removes redundancy, the channel coder adds just enough of it to combat errors that may arise due to the noise in the physical channel.

We consider the bus between two chips as the physical channel and the transmitter and receiver blocks to be a part of the pad circuitry, driving (in case of the transmitting chip) or detecting (in case of the receiving chip) the data signals. We will assume here that the signal levels are sufficiently high so that the channel can be considered as be noiseless. The noiseless channel assumption allows us to eliminate the channel coder resulting in the system shown in Fig. 2-7.

There have two functions f_1, f_2 in the source encoder shown in Fig. 2-8. The function f_1 decorrelates the input so that all linear dependencies can been removed. The function f_2 employs a variant of encoding whereby, instead of minimizing the average number of bits at the output, it reduces the average number of transitions.

Therefore, the function f_1 decorrelates the input and adjusts the input probability distribution so that function f_2 can reduce the transition activity by mapping encoding.

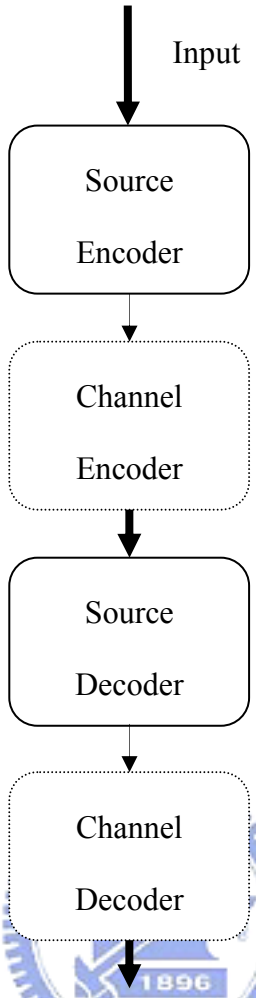


Fig. 2-6. A general communication system.

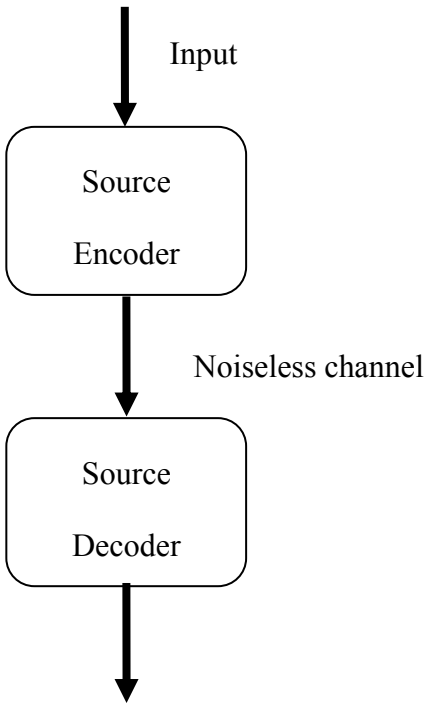


Fig. 2-7. A general communication system of noiseless channel.

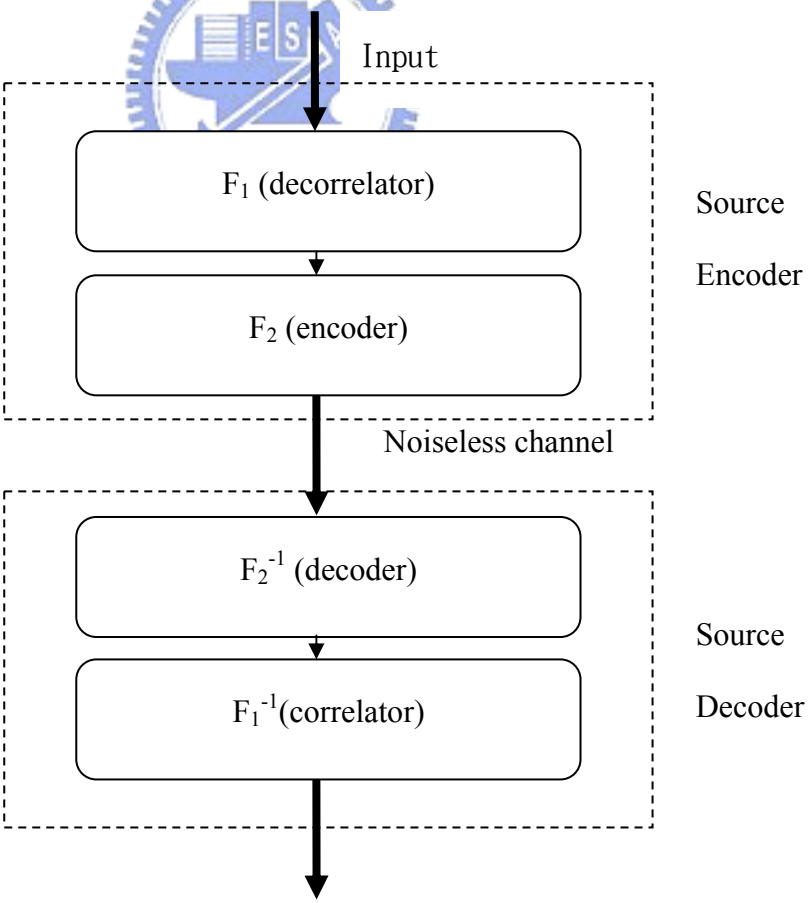


Fig. 2-8. A Practical communication system of noiseless channel.

In this thesis, we choose the Difference-Based Mapping as the function f_1, the Probability-Based Mapping as the function f_2. In the later chapter, we will use this encoding method to compare with other encoding schemes including Bus-Invert, XOR, XNOR, proposed scheme.

The method of Difference-Based Mapping (dbm) is shown as follows Eq. 2-4. The *x(n)* is the input data, The prediction $\hat{x}(n)$, is a function of the past value of *x(n)*. The dbm function returns the difference between *x(n)* and $\hat{x}(n)$ properly adjusted so that the output fits in the available *B* bits.

$$
\begin{aligned}
&\textit{if } (x(n) \geq \hat{x}(n) \;\&\&\; 2\hat{x}(n) \geq x(n)) \\
&\quad dbm = 2x(n) - 2\hat{x}(n); \\
&\textit{else if } (x(n) < \hat{x}(n) \;\&\&\; 2\hat{x}(n) - x(n) < 2^{B}) \\
&\quad dbm = 2x(n) - 2\hat{x}(n) - 1; \\
&\textit{else if } (\hat{x}(n) < 2^{B-1}) \\
&\quad dbm = x(n); \\
&\textit{else} \\
&\quad dbm = 2^{B} - 1 - x(n);
\end{aligned}
\tag{2-4}
$$

In the Difference-Based Mapping (dbm), we define four ranges for mapping, $\{\hat{x}(n) < 2^{B-1}\}$, $\{2\hat{x}(n) - 2^{B} \leq x(n) \leq \hat{x}(n)\}$, $\{\hat{x}(n) < x(n) < 2\hat{x}(n)\}$, and others. We can choose proper calculation according to four mapping ranges. For an example is listed in Table 2.3, we see that the dbm output is 0 when the current *x(n)* is equal to the previous $\hat{x}(n)$ and the output value increases as the distance between the current *x(n)* and previous $\hat{x}(n)$ increases. The goal of dbm is convert the total data distribution to close to 0 so that the number of transitions would be reduced. We see the occurrence distribution at the output of dbm for EEG 8 bits data is shown in

Fig. 2-9 and Fig. 2-10.The dbm skew the original distribution for most of the data sets and hence enable function *f2* ,Probability-Based Mapping (pbm) to reduce the number of transitions even more.

Table 2-3 Example of Difference-Based Mapping (dbm)

$\hat{x}(n)$	X(n)	Dbm(x(n), $\hat{x}(n)$)
011	000	101
011	001	011
011	010	001
011	011	000
011	100	010
011	101	100
011	110	101
011	111	111

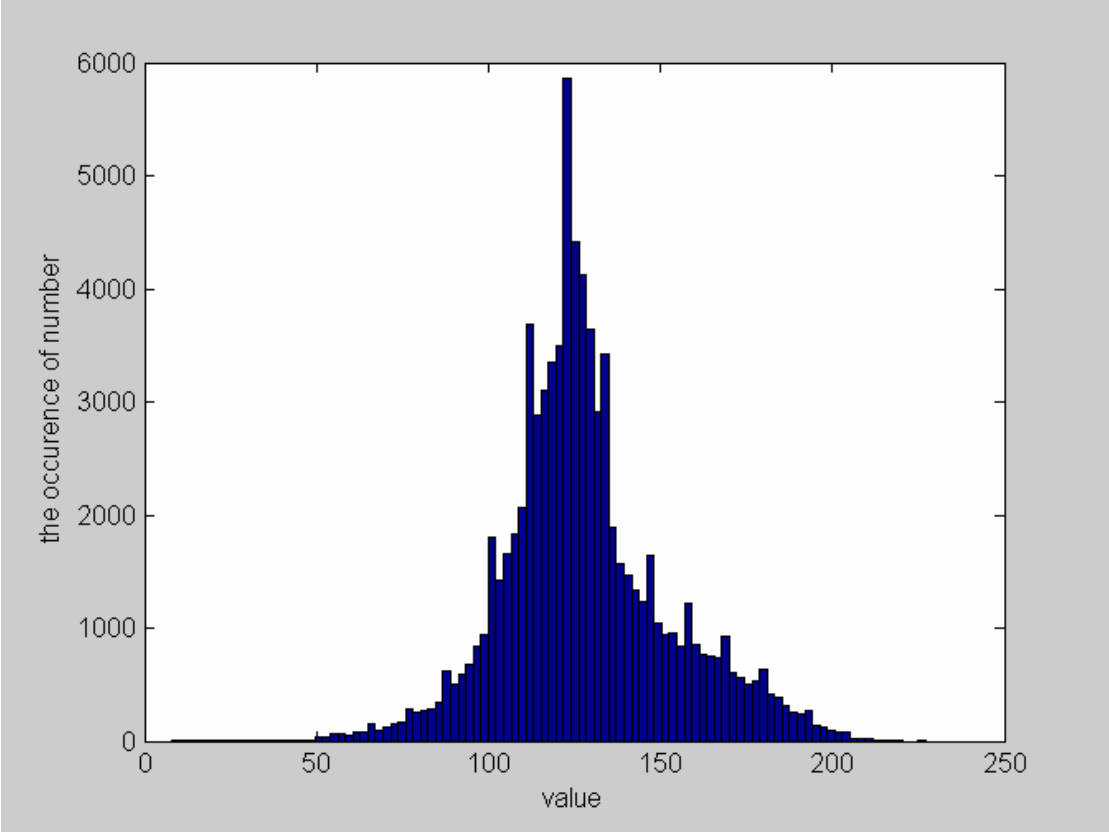


Fig. 2-9. Occurrence distribution for EEG data before dbm.

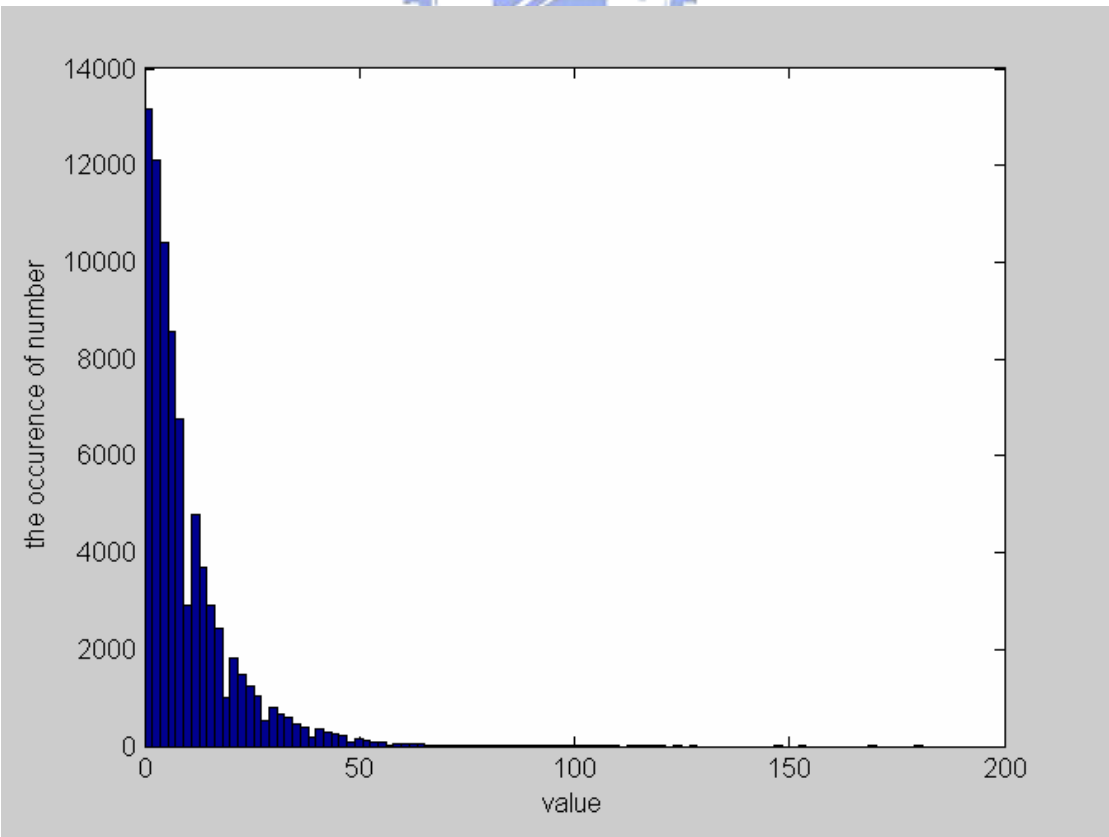


Fig. 2-10. Occurrence distribution for EEG data after dbm.

The Probability-Based Mapping (pbm) is a method of sorting for reducing the number of '1'.It satisfies given below.

$$if \ \Pr(i) > \Pr(j) \ then \ pbm(i) \le pbm(j) \ \forall(a,b) \qquad (2\text{-}5)$$

The probabilities in (2-6) can be computed using a representative data sequence. If the most probable value is i, then *pbm(i) = 0*.Then the second most probable value is j, *pbm(j) =1* and so on. Therefore all value are mapped to value in 2^i (i=0…B-1) by pbm. We can make a sorting table according to probability. An example of pbm is listed in Table 2-4

Table 2-4 Example of Probability-Based Mapping (pbm)

i	Pr(*i*)	Pbm(*i*)
000	0.37	000
001	0.14	010
010	0.22	001
011	0.11	011
100	0.05	101
101	0.03	110
110	0.06	100
111	0.02	111

In summary, we can reduce transition activity by combining with dbm and pbm encoding schemes. It can make the value having higher probability of occurrence to be assigned code words with fewer ON bits. In VLSI circuits, power dissipation depends on the number of transitions occurring at the capacitive nodes of the circuit. But unfortunately, the dbm + pbm require more hardware for build the input

probability distribution table and more execution time for encoding.

2.3 Power Aware Data Bus Codec

According to different kinds of data properties and correlations, the various encoding schemes can be generated. Zero-Transition Activity encoding [15] that needs high correlation and tardy variation in data type is suitable for instruction memory. Bus-Invert encoding method [12] that needs low correlation and rapid variation in data type is suitable for data memory. Dbm and Pbm encoding schemes [16] have an advantage of that it can change correlation of data and choose proper value by probability mapping. Dbm and Pbm encoding scheme is suitable for specific data value range, but Dbm and Pbm encoding scheme pays a heavy penalty on hardware implementation cost.

On the other hand, in general, although data width is constant, the variation of the most significant bit group (MSBG) is different from the variation of least significant bit group (LSBG). We define the MSBG is from 4^{th} bit to 7^{th} bit, the LSBG is from 0^{th} bit to 3^{rd} bit for 8 bits data bit width. For example, we choose the first ten decimal data sequences in Fig. 2-11 and the corresponding binary representation for observation in Table. 2-5. In Table 2-5, the data value ranges at between 115 and 150 and the variation of the MSBG is smoother than that of LSBG. Fig. 2-12 shows the variation curve.

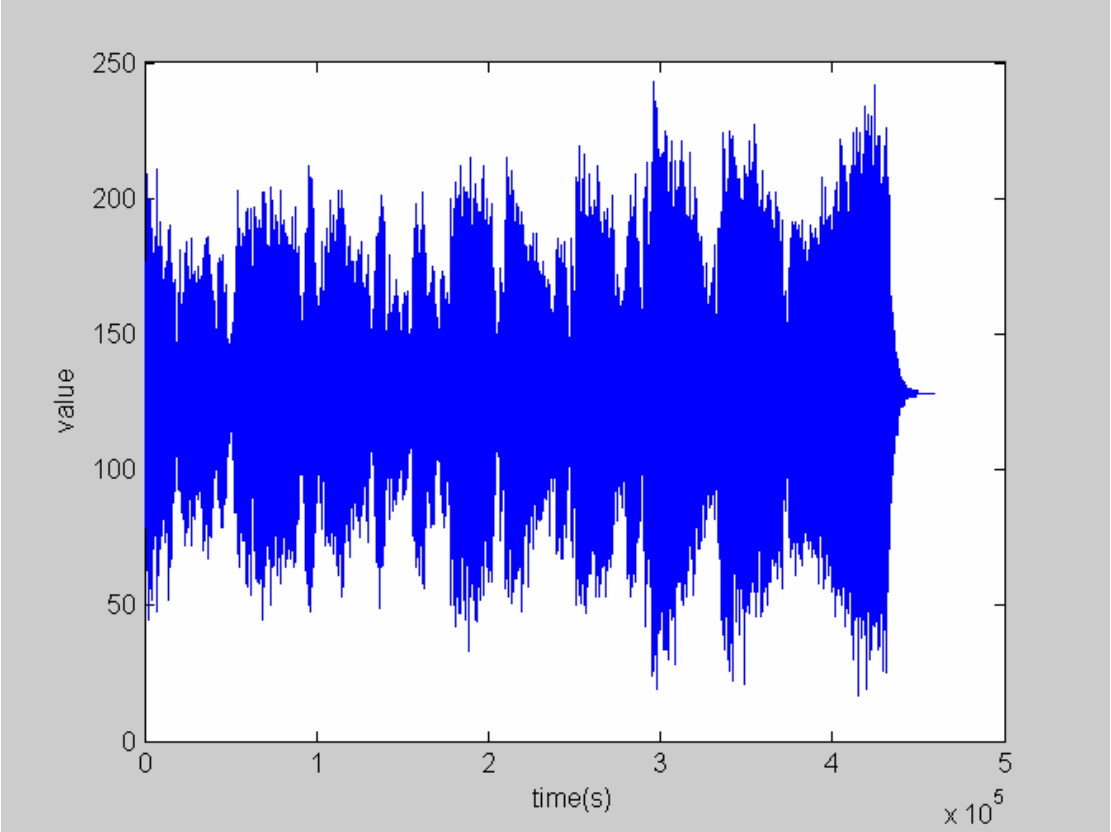


Fig. 2-11. Waveform of the classic music.

Table 2-5 First Ten Data Sequences of Classic Music

	Value(decimal)	Value(binary)
1	140	1000_1100
2	131	1000_0011
3	146	1001_0010
4	151	1001_0111
5	136	1000_1000
6	125	0101_1101
7	115	0101_0011
8	130	1000_0010
9	145	1001_0001
10	139	1000_1011

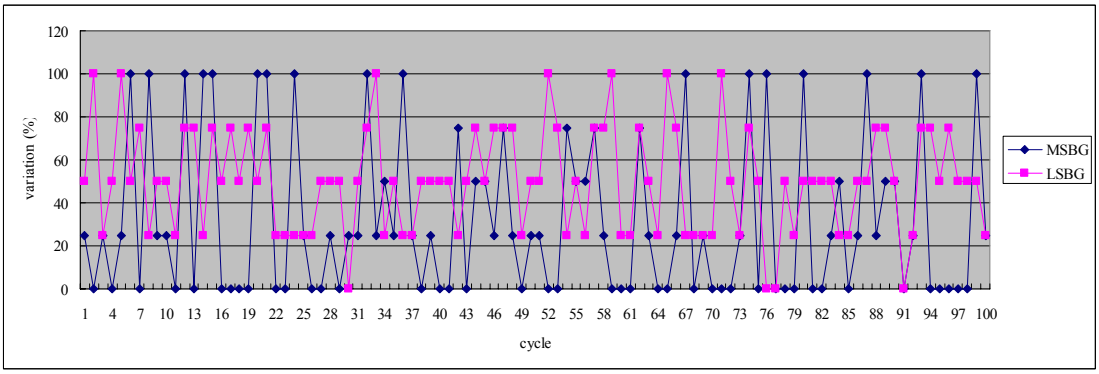


Fig.2-12. Data variation.

Table 2-6 Data Variation

	MSBG	LSBG
Total Hamming distance	126	200
Average of variation	31.5%	50%

We can find the difference obviously between MSBG and LSBG in Fig. 2-12. Therefore, unlike in [19], we can separate specific blocks from data bit width such that the proper encoding can be applied to each block. The transition activity of data transmission can be reduced by encoding.

2.3.1 Proposed Data Bus Codec

The architecture of encoder have four kinds of encoding schemes, Invert, XOR [17][18], XNOR [17][18], original, and then we will introduce each encoding algorithm and proper data type for each a algorithm.

The Invert function is given in Eq. 2-6, where *Hamming*($x(n)$, $\hat{x}(n)$) returns the Hamming distance between the current data $x(n)$ and the previous data $\hat{x}(n)$. If the Hamming distance exceeds half the number of bus lines, and then the input is inverted

and the inversion is signaled using an extra bit. An example of classic music before using Invert is listed in Table 2-7, and an example of classic music after using Invert is listed in Table 2-8.

$$\begin{aligned} &if\ (\ Hamming(x(n)\ ,\hat{x}(n)\ )\ > \frac{Bitwidth}{2} \\ &\quad y(n)\ =\ inv(\ x(n)\ ); \\ &else \\ &\quad y(n)\ =\ x(n); \end{aligned} \tag{2-6}$$

Table 2-7 Example of Classic Music before Using Invert

cycle	$\hat{x}(n)$	X(n)	transitions
1	00000000	10001100	3
2	10001100	10000011	4
3	10000011	10010010	2
4	10010010	10010111	2
5	10010111	10001000	5
6	01110111	01011101	3
7	01011101	01010011	3
8	01010011	10000010	4
9	10000010	10010001	3
10	10010001	10101010	5
Total transitions			34

Table 2-8 Example of Classic Music after Using Invert

cycle	$\hat{x}(n)$	X(n)	$Hamming(x(n), \hat{x}(n))$	Y(n)	Inv	transitions
1	00000000	10001100	3	10001100	off	3
2	10001100	10000011	4	10000011	off	4
3	10000011	10010010	2	10010010	off	2
4	10010010	10010111	2	10010111	off	2
5	10010111	10001000	5	01110111	on	(*)3
6	01110111	01011101	3	01011101	off	3
7	01011101	01010011	3	01010011	off	3
8	01010011	10000010	4	10000010	off	4
9	10000010	10010001	3	10010001	off	3
10	10010001	10101010	5	01010101	on	(*)3
Total transitions						30

The block diagram of Invert encoding is sketched in Fig. 2-13, where *Hamming* function is composed of 8 exclusive-OR gates and adders for 8-bit length input.

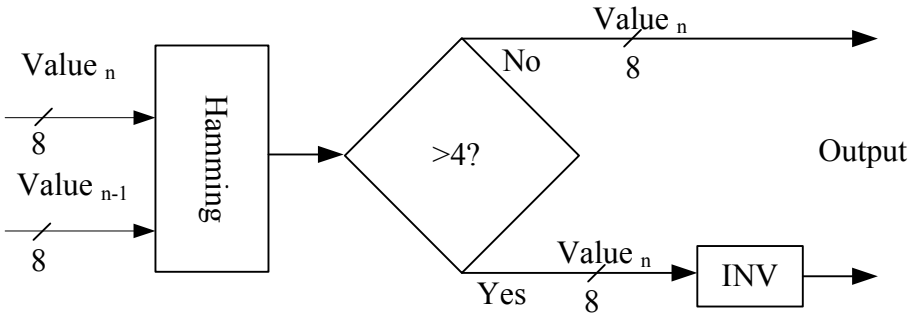


Fig. 2-13. Block diagram of Invert coding.

The XOR function is given in Eq. 2-7, where $XOR(x(n), \hat{x}(n))$ returns the value of the current data $x(n)$ exclusive-or the previous data $\hat{x}(n)$. If the value of $Hamming(x(n), \hat{x}(n))$ is smaller than $Hamming(XOR(x(n), \hat{x}(n)), \hat{x}(n))$, and then the output for transmission equals to $XOR(x(n), \hat{x}(n))$ .Otherwise, the output for transmission will be unchanged.

For example, classic music coding results using transparent and XOR coding schemes are listed in Table 2-9 and Table 2-10.

$$\begin{aligned} &if\ (\ Hamming(x(n), \hat{x}(n)\ ) > Hamming(XOR(\ x(n),\ \hat{x}(n)\ ),\ \hat{x}(n)\ ) \\ &\quad y(n) = XOR(\ x(n),\ \hat{x}(n)\ ); \\ &else \\ &\quad y(n) = x(n); \end{aligned} \tag{2-7}$$

$$XOR(\ x(n),\ \hat{x}(n)\ ) = x(n) \otimes \hat{x}(n);$$

Table 2-9 Example of Classic Music before Using XOR

cycle	$\hat{x}(n)$	X(n)	Transitions
1	00000000	10001100	3

2	10001100	10000011	4
3	10000011	10010010	2
4	10010010	10010111	2
5	10010111	10001000	5
6	01110111	01011101	3
7	01011101	01010011	3
8	01010011	10000010	4
9	10000010	10010001	3
10	10010001	10101010	5
Total transitions			34

Table 2-10 Example of Classic Music after Using XOR

cycle	$\hat{x}(n)$	X(n)	$Hamming(x(n), \hat{x}(n))$	$Hamming(XOR(x(n)), \hat{x}(n))$	Y(n)	XOR	transitions
1	00000000	10001100	3	3	10001100	off	3
2	10001100	10000011	4	3	00001111	on	(*)3
3	00001111	10010010	5	2	10011101	on	(*)2
4	10011101	10010111	2	5	10010111	off	2
5	10010111	10001000	5	3	00011111	on	(*)3
6	00011111	01011101	2	5	01011101	off	3
7	01011101	01010011	3	4	01010011	off	3
8	01010011	10000010	4	2	11010001	on	(*)2
9	11010001	10010001	1	3	10010001	off	3
10	10010001	10101010	5	4	00111011	on	(*)4
Total transitions							28

The block diagram of XOR encoding is sketched in Fig. 2-14. The conditional block will select optimal result which the function *Hamming ()* has smallest value.

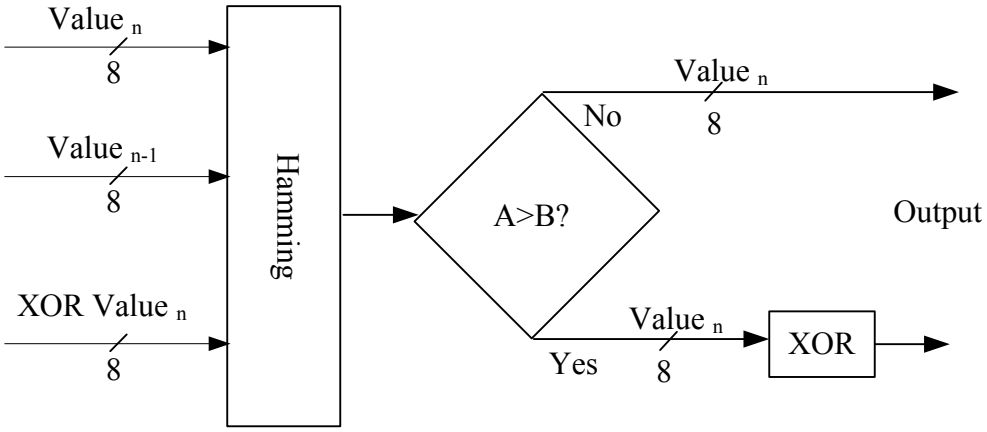


Fig. 2-14. Block diagram of XOR coding.

The XNOR function is given in Eq. 2-8, where *XNOR(x(n,),* $\hat{x}(n)$*)* returns the value of the current data *x(n)* exclusive-nor the previous data $\hat{x}(n)$. If the value of *Hamming*($x(n)$,$\hat{x}(n)$) is smaller than *Hamming*(*XNOR*($x(n),\hat{x}(n)$) ,$\hat{x}(n)$) ,and then the output for transmission equals to *XNOR(x(n),* $\hat{x}(n)$*)* .Otherwise, the output for transmission will be unchanged. The inversion is signaled using an extra bit.

$$
\begin{aligned}
&\textit{if}\ (\ \textit{Hamming}(x(n),\hat{x}(n)\ ) > \textit{Hamming}(\textit{XNOR}(\ x(n),\hat{x}(n)\ ),\ \hat{x}(n)\ )\\
&\quad y(n) = \textit{XNOR}(\ x(n),\hat{x}(n)\ );\\
&\textit{else}\\
&\quad y(n) = x(n);
\end{aligned}
\qquad (2\text{-}8)
$$

$$
\textit{XNOR}(\ x(n)\ ) = \sim (x(n) \otimes \hat{x}(n));
$$

The logic diagram is shown in Fig. 2-15. The conditional block will select optimal result which the function *Hamming* has smaller value.

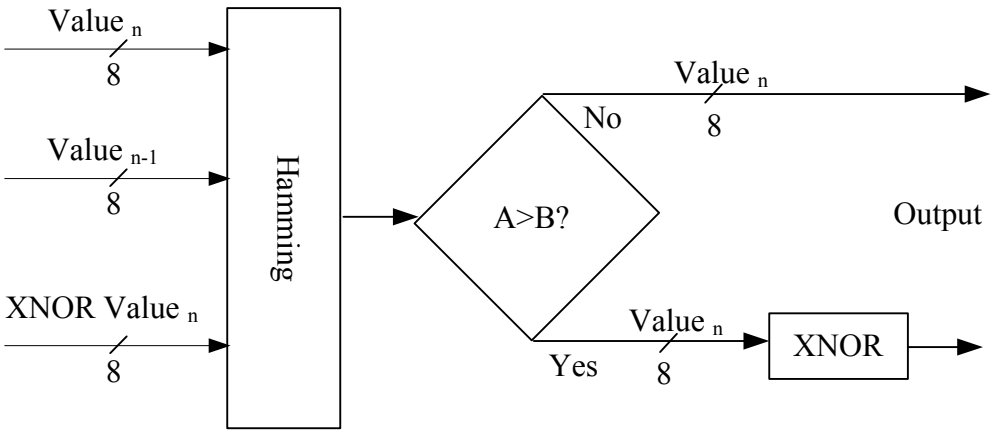


Fig. 2-15. Block diagram of XNOR coding.

2.3.2 Architecture of Codec

The total codec system overview is shown in Fig. 2-16. The proposed codec architecture is placed between I/O, external memory interface and I/O, and external memory module. The extra bit line on bus is used for notify which function to decoding in decoder.

The proposed encoder architecture is composed of four encoding functions. It targets at different kinds of data types and adaptively choose the optimal encoding way for transmission. According to the property that different bit group location has different kinds of variation, the transmission data would be separated into several blocks for encoding.

The encoder architecture diagram is sketched in Fig. 2-17.

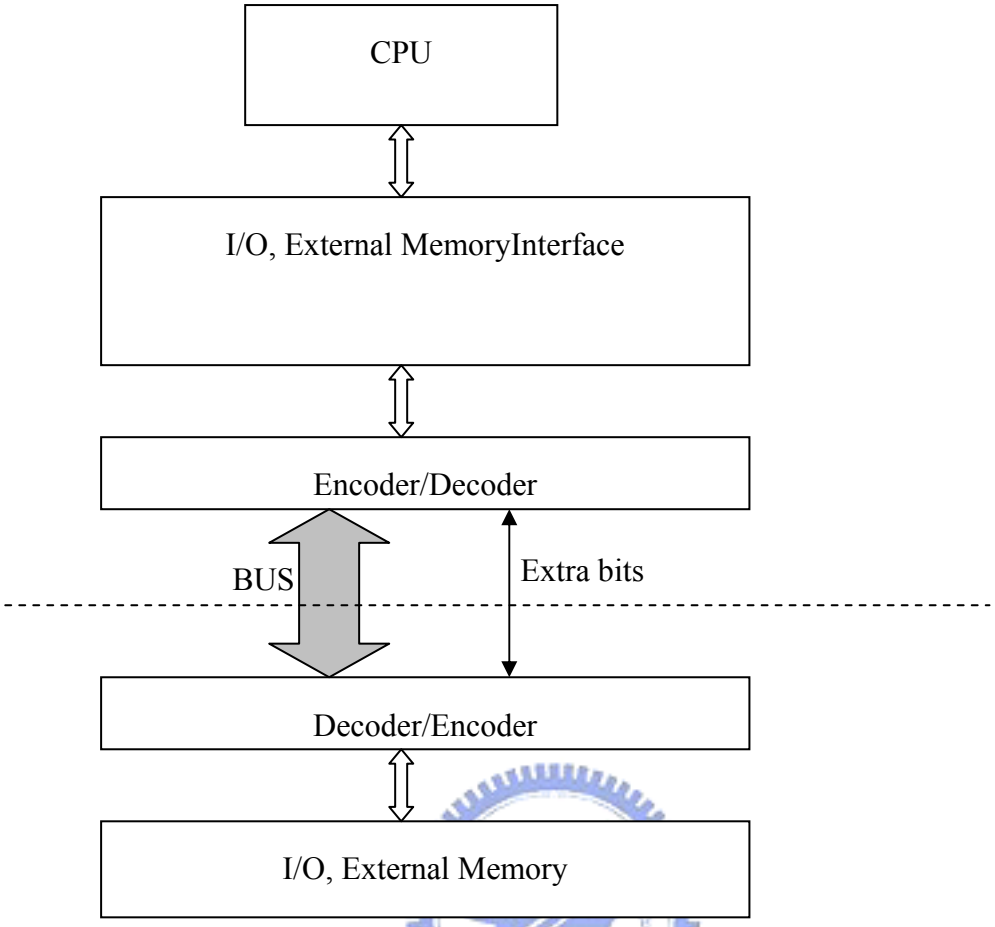


Fig. 2-16. System architecture.

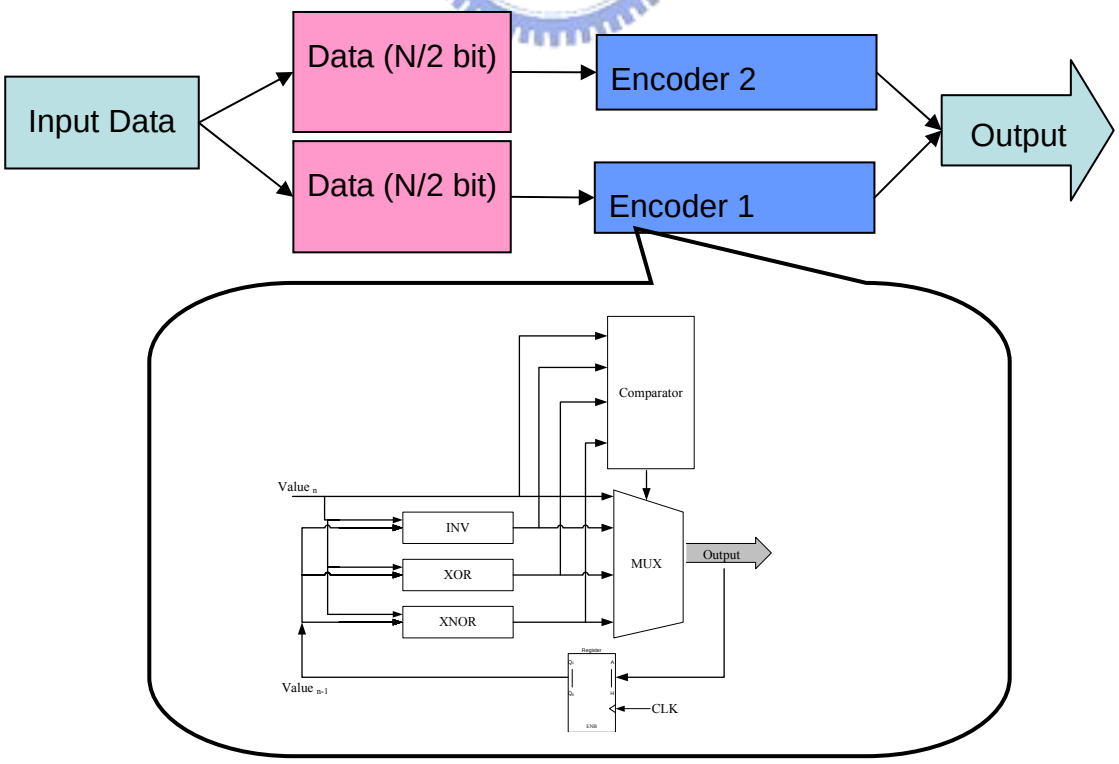


Fig. 2-17. Block diagram of encoder.

The input data for transmission is separated into two or more bit groups. Each bit group has individual encoder for encoding its separated data.

The current data $Value_n$ and the previous data $Value_{n-1}$ enter the *INV, XOR, XNOR* functions, and then the comparator chooses the optimal encoding way that has the minimum Hamming distance and sends the encoded data for transmission on bus.

Our architecture of decoder is similar as architecture of encoder.

It has two input source, transmission data on bus and extra bits. It depends on the extra bits from encoder to decode the data for transmission .Extra bits mean four decoding functions, *INV, XOR, XNOR*, and transparent. After decoding data by MUX, the data will be return to original form by suitable decoding functions. Fig. 2-18 shows the decoder architecture diagram.

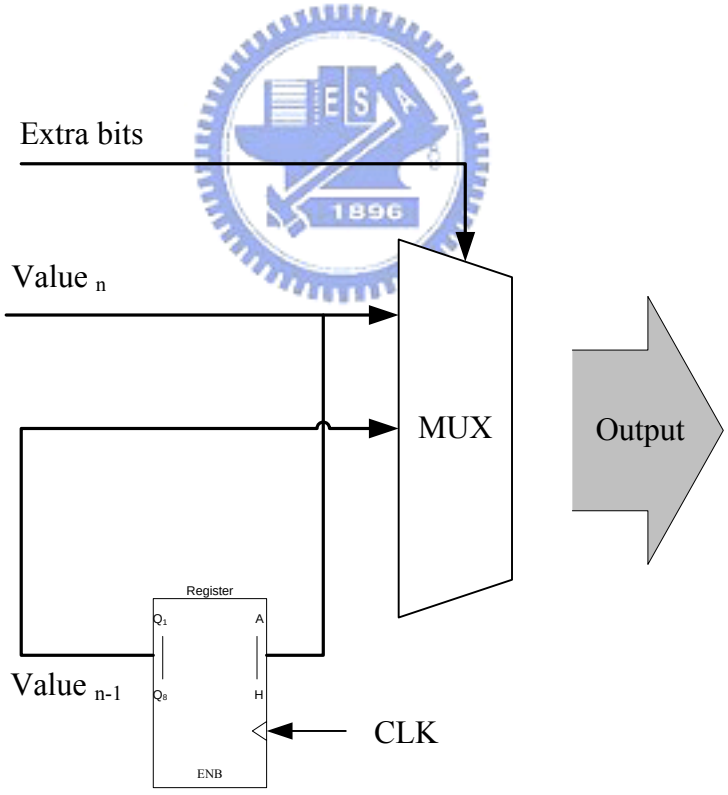


Fig. 2-18. Block diagram of decoder.

2.4 Power Aware Data Bus Codec Simulator

To verify Power Aware Data Bus Codec and compare the performance with other encoding schemes like Bus-Invert, XOR, XNOR, Dbm (different based mapping) plus Pbm (probability based mapping). Our thesis has not only RTL model design but also a simulator by C++ language. The simulator can help us know the switch activity effect in different kinds of data variation.

2.4.1 8 bits Power Aware Data Bus Codec Simulator

In 8 bits Power Aware Data Bus Codec Simulator, we define our proposed codec which is separated into two 4-bit groups for 8-bit length data encoding. And we configure some variability parameters for simulation.

Most significant bit group variability: the variability of the value 0 to 1 or 1 to 0 from 4^{th} to 7^{th} bits

Least significant bit group variability: the variability of the value 0 to 1 or 1 to 0 from $0^{th}$to 3^{th} bits

When the simulator executes a test pattern, it would record the results about bit transitions below:

switch_act switch activity before encoding

switch_act_BI_total switch activity after Bus-Invert encoding

switch_act_XOR _total switch activity after XOR encoding

switch_act_XNOR_total switch activity after XNOR encoding

switch_act_dbm switch activity after dbm encoding

switch_act_dbm+pbm switch activity after dbm + pbm encoding

switch_act_1block_total switch activity after 1 block encoding

switch_act_2blocks_total switch activity after 2 blocks encoding

switch_act_BI_ctrl switch activity on extra bits after Bus-Invert encoding

switch_act_XOR_ctrl switch activity on extra bits after XOR encoding

switch_act_XNOR_ctrl switch activity on extra bits after XNOR encoding

switch_act_1block_ctrl_high switch activity on extra bits in most significant bit group after 1 block encoding

switch_act_1block _ctrl_low switch activity on extra bits in least significant bit group after 1 block encoding

switch_act_2blocks_ctrl_high switch activity on extra bits in most significant bit group after 2 blocks encoding

switch_act_proposal2_ctrl_low switch activity on extra bits in least significant bit group after 2 blocks encoding

In Section 2.1, we know the dynamic power depends on transition activity α. Therefore, we can use switch activity reduction (SAR) as a measurement metric to signify the power reduction.

$$\text{SAR}(\%) = \frac{SA_{before\ encoding} - SA_{after\ encoding} + SA_{control\ extra\ bit}}{SA_{before\ encoding}}, \qquad (2\text{-}9)$$

Where SA denotes switch activity.

We employee four encoding schemes, proposal and configure different variability parameters to simulate 100,000 data by in Fig. 2-19, Fig. 2-20, Fig. 2-21 and Fig. 2-22. We define 1 block is proposed coding scheme with one 8 bits group; 2 blocks is proposed coding scheme with two 4 bits groups;

The x-axis shows the group bits variability and the y-axis shows the switch activity reduction. For example, 25/50 means high level group bits have 25% variability and low level group bits have 50% variability.

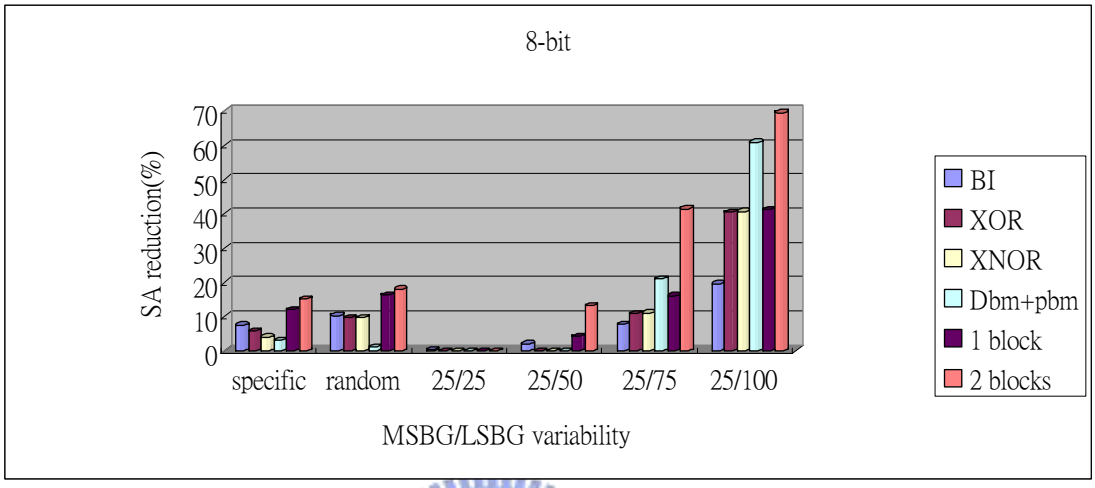


Fig.2-19. Switch activity reduction for 8-bit data.

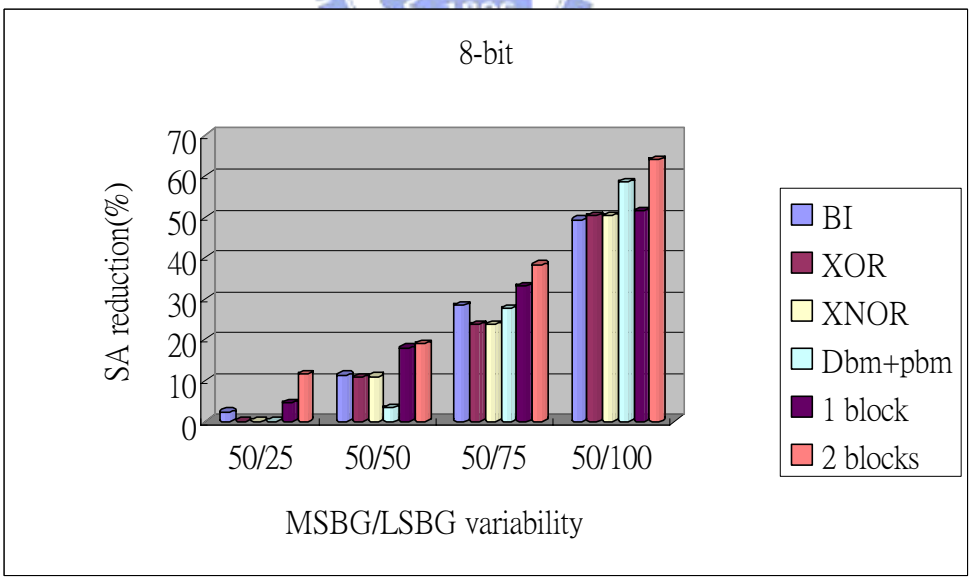


Fig.2-20. Switch activity reduction for 8-bit data.

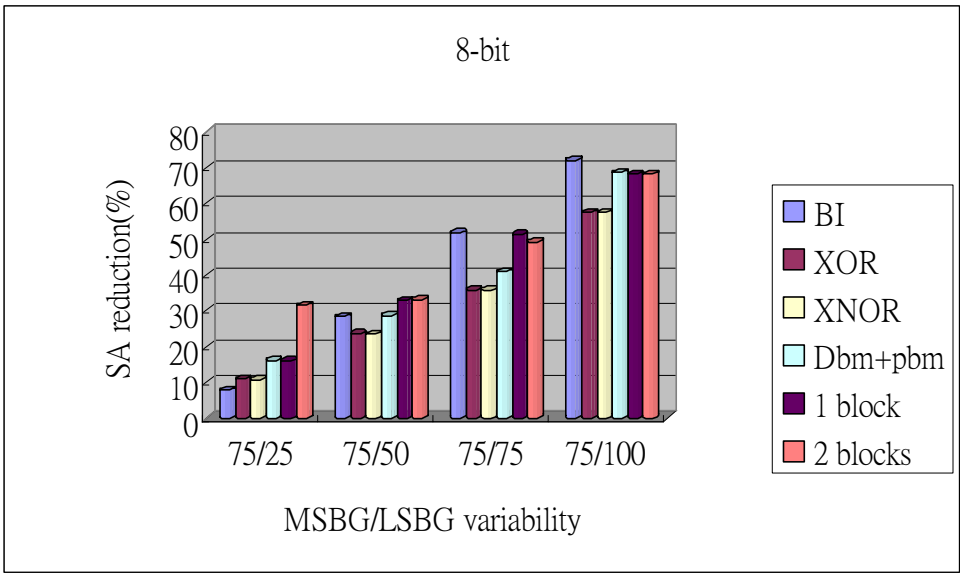


Fig.2-21. Switch activity reduction for 8-bit data.

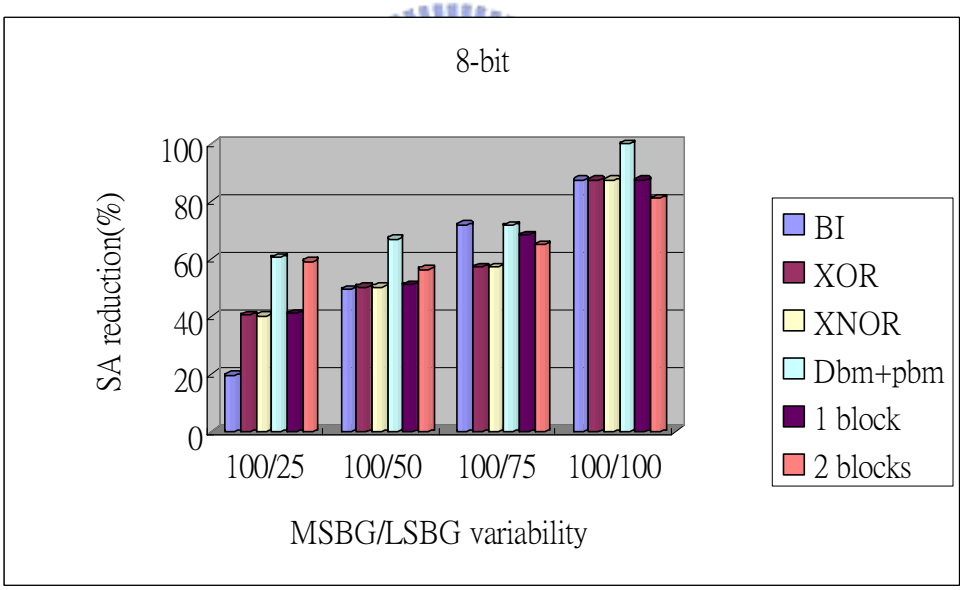


Fig.2-22. Switch activity reduction for 8-bit data.

We add the specific data and random data for simulation except above data. In the specific data, It has high probability in specific range.

The data distribution is shown in Fig. 2-23. The most of general video and audio data distribution are like this figure.

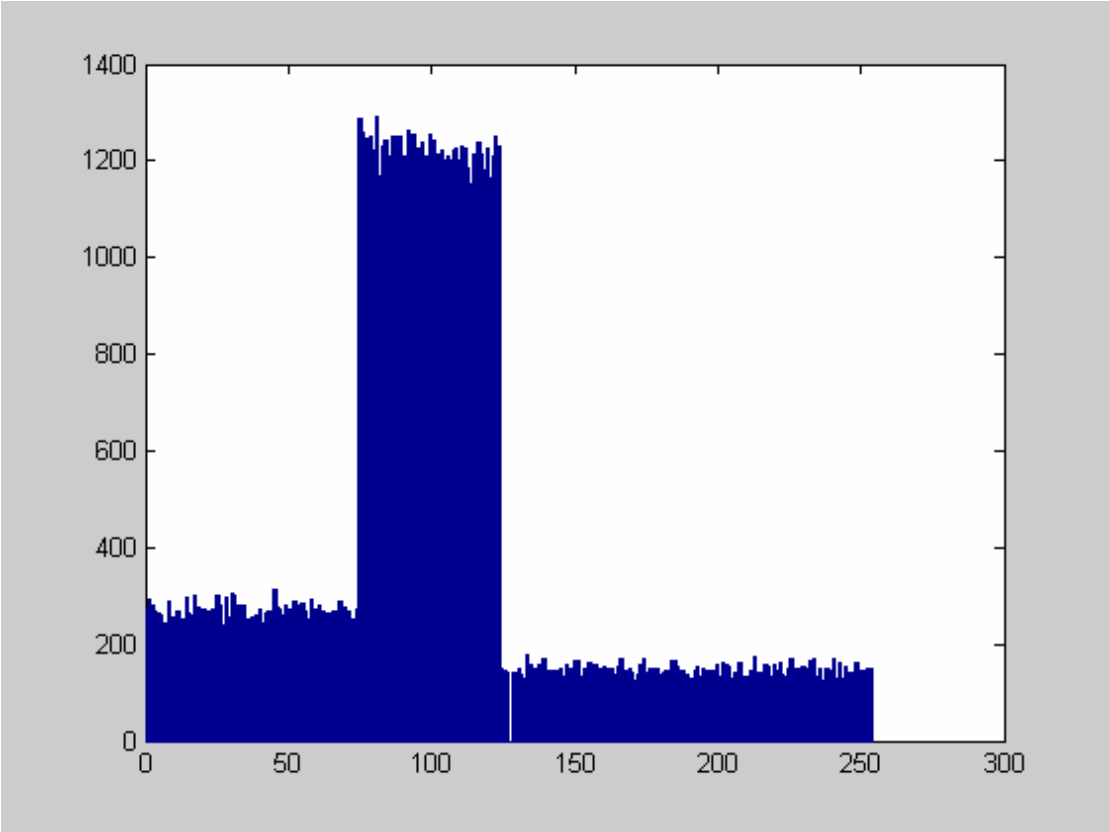


Fig. 2-23. The Data Distribution.

Either in the specific data or in the random data, the switch activity reduction has 15 ~ 18 percentages by proposal scheme. With the increasing of variability parameters, our proposal has more reduction in switch activity. In the variability parameters 25/75 and 75/25, we have 20% in switch activity reduction and our proposal has greater improvement than other encoding schemes.

2.4.2 16 bits Power Aware Data Bus Codec Simulator

In 16 bits Power Aware Data Bus Codec Simulator, we define our proposed codec which is separated into two 8-bit groups and four 4-bit groups for 16-bit length data encoding. And we configure some variability parameters for simulation.

Most significant bit group variability: the variability of the value 0 to 1 or 1 to 0 from 8^{th} to 15^{th} bits

Least significant bit group variability: the variability of the value 0 to 1 or 1 to 0 from $0^{th}$to 7^{th} bits

When the simulator executes a test pattern, it would record the results about bit transitions below:

switch_act switch activity before encoding

switch_act_BI_total switch activity after Bus-Invert encoding

switch_act_XOR _total switch activity after XOR encoding

switch_act_XNOR_total switch activity after XNOR encoding

switch_act_dbm+pbm switch activity after dbm + pbm encoding

switch_act_1block_total switch activity after proposal encoding by 1 blocks.

switch_act_ 2blocks_total switch activity after proposal encoding by 2 blocks.

switch_act_4blocks_total switch activity after proposal encoding by 4 blocks.

switch_act_BI_ctrl switch activity on extra bits after Bus-Invert encoding

switch_act_XOR_ctrl switch activity on extra bits after XOR encoding

switch_act_XNOR_ctrl switch activity on extra bits after XNOR encoding

switch_act_1block_ctrl_high switch activity on extra bits in high level group after proposal encoding by 1 block.

switch_act_1block_ctrl_low switch activity on extra bits in low level group after proposal encoding by 1 block.

switch_act_ 2blocks_ctrl_high switch activity on extra bits in high level group after proposal encoding by 2 blocks.

switch_act_2blocks_ctrl_low switch activity on extra bits in low level group after

proposal encoding by 2 blocks.

switch_act_4blocks_ctrl_high switch activity on extra bits in high level group after proposal encoding by 4 blocks.

switch_act_4blocks_ctrl_low switch activity on extra bits in low level group after proposal encoding by 4 blocks.

We run 100,000 data by four encoding schemes, proposal and configure different variability parameters in Fig.2-24, Fig.2-25, Fig.2-26 and Fig.2-27. Encoding schemes:

Bus-Invert; XOR; XNOR; Dbm+Pbm;

1 block: Proposed coding scheme with one 16 bits group;

2 blocks: Proposed coding scheme with two 8 bits groups;

4 blocks: Proposed coding scheme with four 4 bits groups;

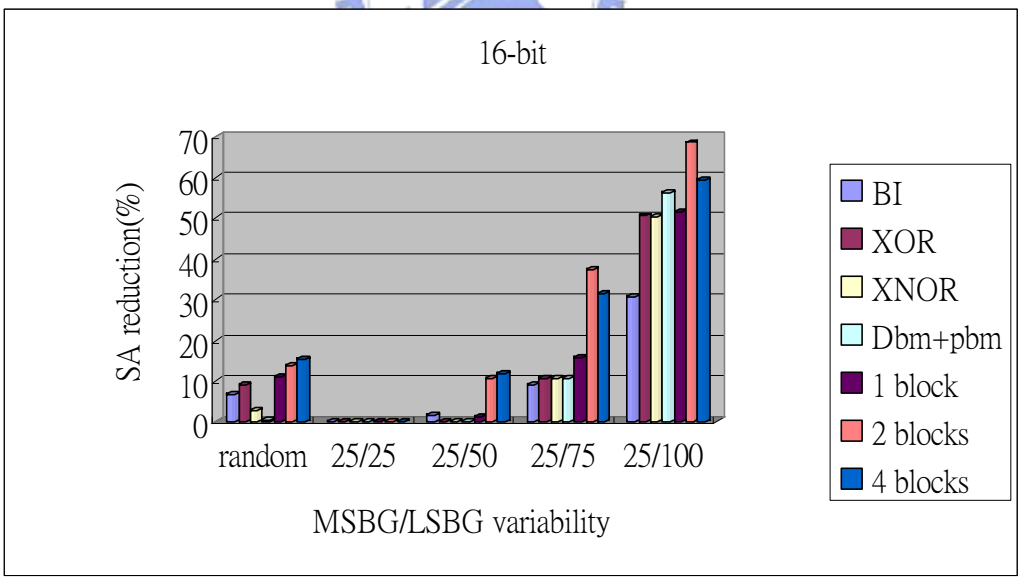


Fig.2-24. Switch activity reduction for 16 bits data.

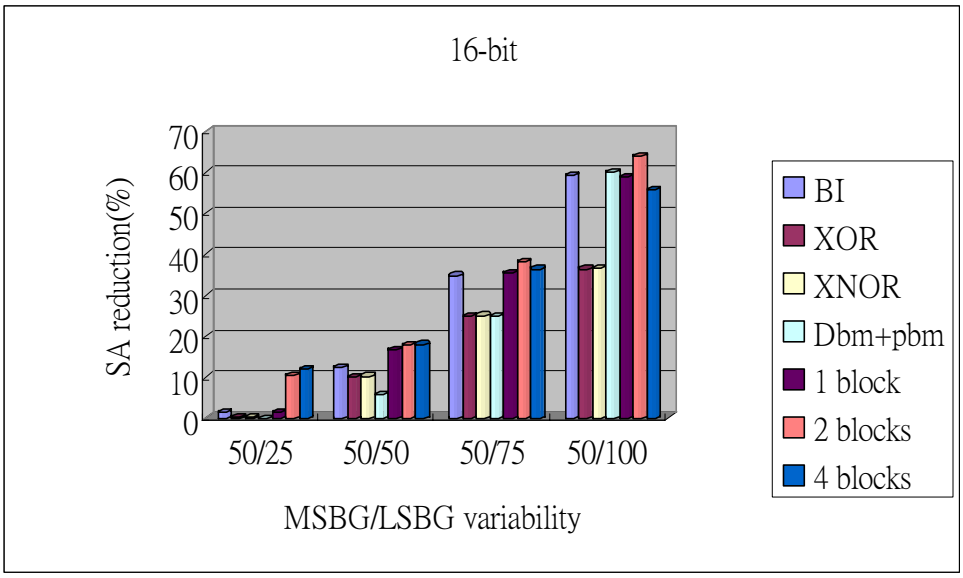


Fig.2-25. Switch activity reduction for 16 bits data.

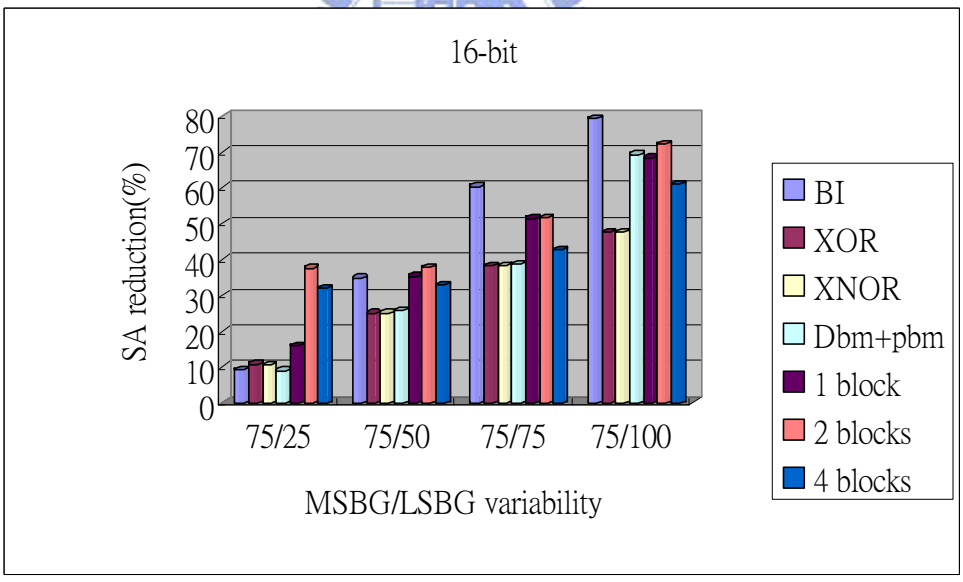


Fig.2-26. Switch activity reduction for 16 bits data.

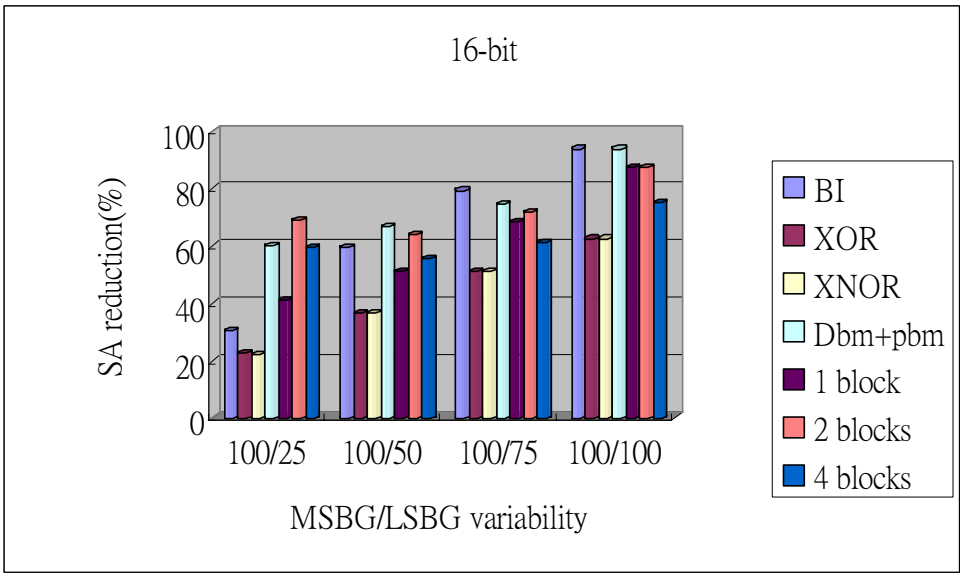


Fig.2-27. Switch activity reduction for 16 bits data.

Either in the random data, the switch activity reduction has 20 percentages by proposed scheme. With the increasing of variability parameters, our proposed method has more reduction in switch activity and has greater improvement than other encoding schemes.

2.5 Result and Analysis

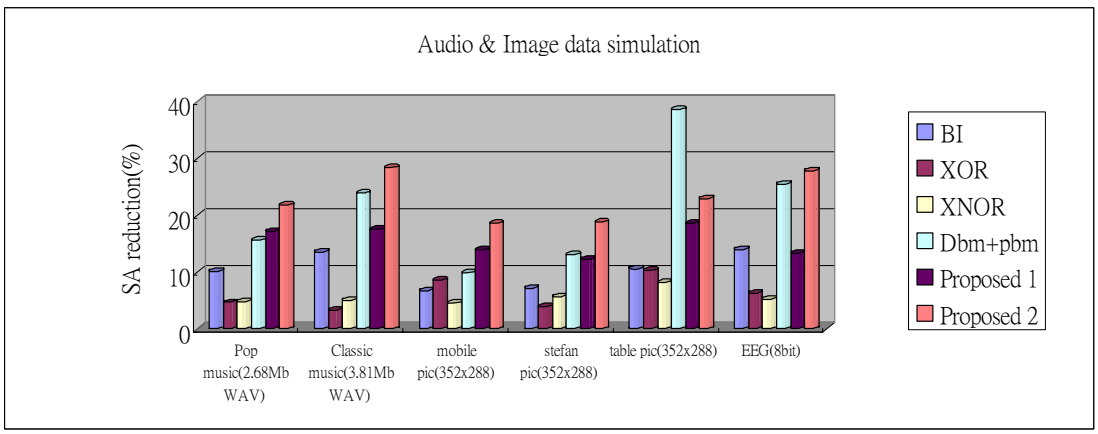


Fig. 2-28. Simulation for Multi-Media data.

Proposed 1: Proposed coding scheme with one 8 bits group;

Proposed 2: Proposed coding scheme with two 4 bits groups;

The simulation for 8 bits multi-media data in Fig. 2-28 shows our proposal has 20 % dynamic power reduction in average. In image data, we choose three 352x288 pictures including mobile, table tennis, and Stefan for encoding. We can find that Table image has low data variability so that is suited for Dbm + Pbm encoding scheme. In other images, the high data variability is well for our proposed encoder can select optimal encoding scheme.

Chapter 3
Low Power Embedded Processor Design

In order to verify the codec function [20], we have performed the codec combine with a 32 bits embedded processor [21]. We will introduce the properties of processor, instruction set, tool chains and other specific designs in this Chapter.

3.1 Architecture of the Low Power Embedded Processor

3.1.1 Low Power Embedded Processor Core

Our processor applies RISC architecture including low power designs, which are Master-Slave cache, low power phased cache controller, and power aware data bus codec.

The low power embedded processor has seven-pipeline architecture. All instructions start by using the program counter (PC) to supply the instruction address to the instruction memory. After the instruction is fetched, ID stage decodes the instruction and specifies register operands. Once the operands have been fetched in ALU, they can be operated to compute a memory address, to compute an arithmetic result, or to compare. If the instruction is an arithmetic-logical instruction, the result from ALU must be written to a register. If the operation is a load or store, the result from ALU is used as an address to either store or load a value .The result from the ALU or memory is written back into the REG stage. Cache controller controls the Load/Store operation in the memory peripheral device. Fig. 3-1 shows the architecture

of processor.

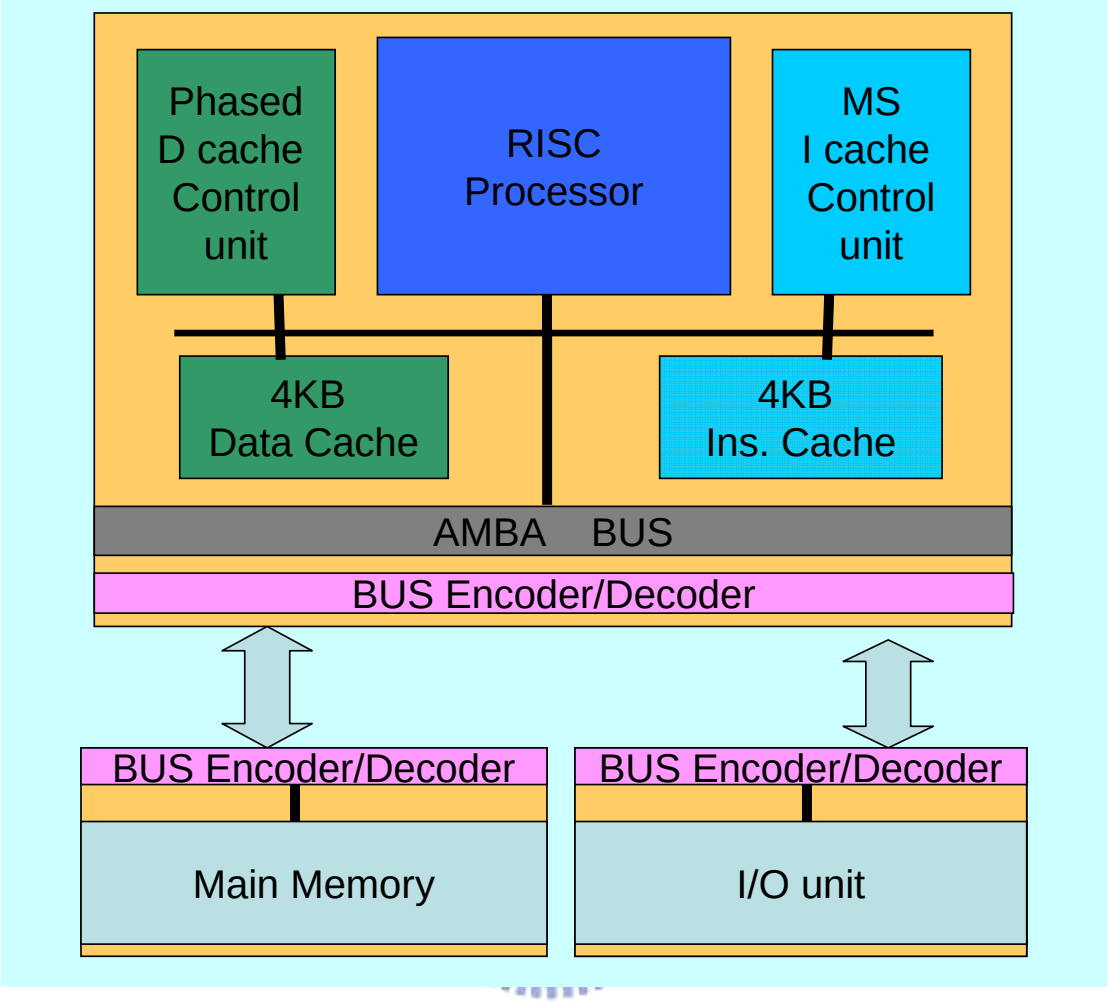


Fig. 3-1. The architecture of processor.

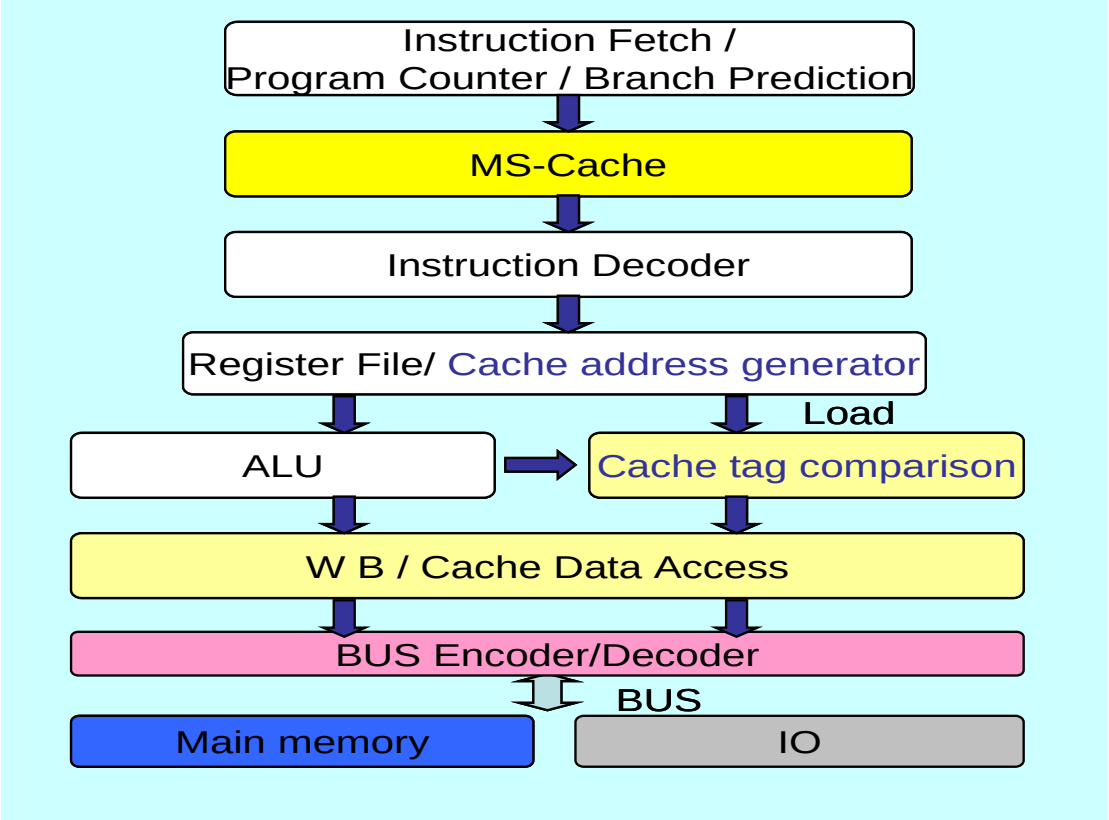


Fig. 3-2. Pipeline processing flow.

The processor has 7 pipeline architecture including Instruction fetch/Program Counter/Branch Prediction, MS cache(2 stages), Instruction decoder, Register file, ALU/Cache tag comparison, and Write back/Cache data access.

The seven stages are the following:

PC Counter/Branch Predict/ Instruction Fetch： In the top portion of hardware architecture, Program counter handles branch instructions and generates the PC address. The instruction is read from memory using the address in the PC and then is placed in the ID pipeline register. Due to some instructions need PC address to be computed in ALU stage, the PC address would be saved stage by stage. Therefore, PC address is saved in the next stage register. In order to avoid an instruction be fetched after branch instruction occurs, we set two flags to handle branch instructions. These

flags can show whether the pipeline is in stall state and decide the stage process.

MS-cache (2 stages): The second portion of Fig. 3-2 shows the operation of instructions. If data miss occurs, it will replace data from main memory. The MS-cache design is based on phased cache. The phased cache compares Tag value in first cycle, and reads Hit data to ID stage in second cycle. By the way, MS-cache also enhances the hit rate for branch/jump instructions.

Instruction Decoder：In ID stage, the instruction separates into two-source registers location, one destination register location. These locations can get source operands for the Register stage and provide destination operand for ALU stage.

Register File: It provides 16 general-purpose registers, 16 interrupt registers for external interrupt, internal interrupt and other configuration.

ALU/Cache Tag access：All operands computation from Register File are executed in ALU stage. Data forwarding is supported in ALU stage to eliminate RAW hazard. Meanwhile, the value in Tag cache is compared with memory address and is verified whether it is a cache hit or miss when Load/Store instructions are executed.

Write-Back/Cache data access： The ALU writes data back to the Register file, cache or memory in this stage. In case of Load/Store instructions execution, it would access memory data according to a cache hit.

Five specific hardware designs is supported for DSP:

SIMD(Single Issue Multi Data) support: 8/16 bits SIMD instruction set is supported to improve multi-media processing, such as 8 bits image processing or 16 bits speech processing.

Bit Reverse：A memory addressing mode is designed for FFT. For example, address 01101 can be transformed to 10110.

MAC can be finished in one cycle.

Effective Data forwarding [22].

Conditional Branch：Prediction – untaken method.

3.1.2 Low Power Embedded Processor Instruction Set

The instruction set has four categories: Data moving instructions, Arithmetic & Logic instructions, Branch/Jump instructions, SIMD instructions and others.

6 addressing modes are supported: Direct, Reg to Reg, Indirect, Displacement (base add), Index and Bit-Reverse addressing modes.

Table 3-1 Data Moving Instructions List

Instruction	Opcode	Example	Mode
MOVRC	000001	MOV rd,data	Direct
MOVRR	000010	MOV rd,rs	Reg-Reg
MOVRM	000011	MOV rd,address	Direct
MOVMR	000100	MOV address,rs	Direct
MOVMRR	000101	MOV @rs2,rs	Indirect
MOVRRM	000110	MOV rd,@rs	Indirect
MOVARR	100010	MOV rd(a),rs(b)	Reg-Reg
MOVB	101111	MOVB rd,base(rs)	Displacement
MOVI	110000	MOVI rd,rs1(rs2)	Index
MOVREVRM	101010	MOV rd,address	Bit Reverse
MOVREVMR	101011	MOV address,rs	Bit Reverse
MOVREVMRR	101100	MOV @rs2,rs	Bit Reverse

MOVREVRRM	101101	MOV rd,@rs	Bit Reverse

Table 3-2 Arithmetic & Logic Instructions List

Instruction	Opcode	Example
ADDRR	001000	ADD rd,rs1,rs2
SUBRR	001010	SUB rd,rs1,rs2
MULRR	001100	MUL rd,rs1,rs2
ADDRC	000111	ADD rd,data
SUBRC	001001	SUB rd,data
MULRC	001011	MUL rd,data
MACR	100111	MAC rd,rs1,rs2
MACC	110001	MAC rd,rs1,data
ANDRR	001110	AND rd,rs1,rs2
ORRR	001111	OR rd,rs1,rs2
XORRR	010000	XOR rd,rs1,rs2
INVR	010001	INV rd,rs

Table 3-3 Branch/Jump Instructions List

Instruction	Opcode	Example
JMP	010010	JMP address
JMPR	010011	JMP @rs
JBE	010100	JBE rs1,address
JNE	010101	JNE rs1,address
JMB	010110	JMB rs1,address
JLB	010111	JLB rs1,address

JBER	011000	JBER rs1,rs2,address
JNER	011001	JNBR rs1,rs2,address
JMBR	011010	JMBR rs1,rs2,address
JLBR	011011	JLBR rs1,rs2,address
CALL	100011	CALL address
RET	011110	RET

Table 3-4 SIMD Instructions List

Instruction	Opcode	Example
MOVHLRC	110001	MOVHLRC rd,direct
MOVHURC	110010	MOVHURC rd,direct
ADDHRR	110011	ADDHRR rd,rs1,rs2
SUBHRR	110100	SUBHRR rd,rs1,rs2
MULHRR	110101	MULHRR rd,rs1,rs2
MACHR	100110	MACHR rd,rs1,rs2
ANDHRR	110110	ANDHRR rd,rs1,rs2
ORHRR	110111	ORHRR rd,rs1,rs2
XORHRR	111000	XORHRR rd,rs1,rs2
ADDBRR	111001	ADDBRR rd,rs1,rs2
SUBBRR	111010	SUBBRR rd,rs1,rs2
MULBRR	111011	MULBRR rd,rs1,rs2
ANDBRR	111100	ANDBRR rd,rs1,rs2
ORBRR	111101	ORBRR rd,rs1,rs2
XORBRR	111110	XORBRR rd,rs1,rs2

In case of SIMD instructions, the 32 bits data in the register file is divided into 8 bits or 16 bits blocks. Each block are parallel processed. Therefore, it can improve 8 bits or 16 bits calculation.

For example, the following is MACHR instruction,

$$R_d = ACC = ACC + A1 \times B1 + A2 \times B2$$

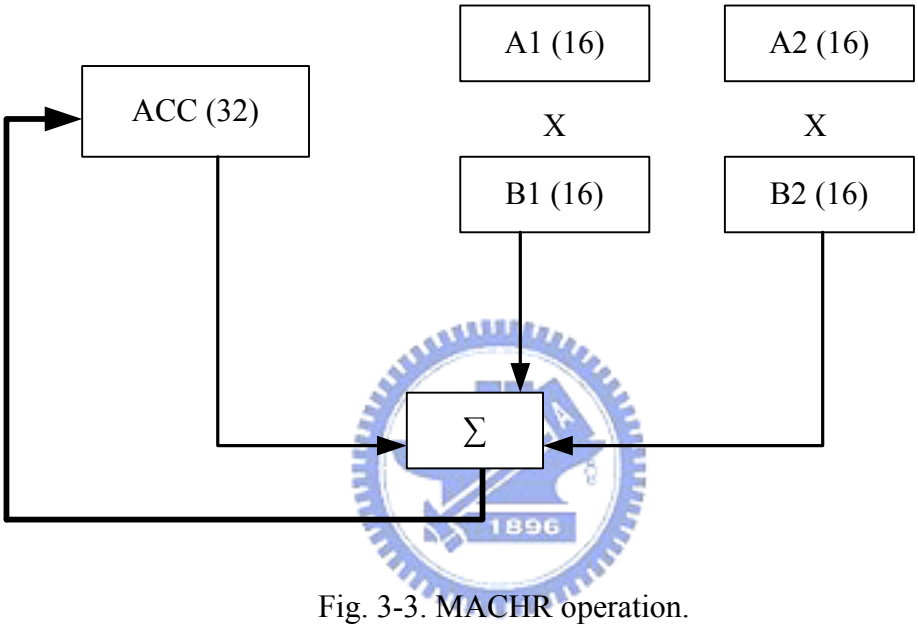


Fig. 3-3. MACHR operation.

Table 3-5 Other Instructions List

Instruction	Opcode	Example
SET	011100	SET A,rs
INTOK	011101	INTOK
SHR	100000	SHR rs
SHL	100001	SHL rs
ENDC	011111	ENDC

SET: It can sets two extra 16 bits I/O ports.

INTOK: Instructions for software interrupt.

SHR: It would right shift 1 bit from rs.

SHL: It would left shift 1 bit from rs.

3.2 Configurable Master-Slave I-Cache Controller

In general, 20%~30% of total power dissipation in the processor dissipated in instruction cache. Therefore, the configurable Master-Slave Instruction cache controller is designed for low power design.[24]

3.2.1 The Proposal of Configurable Master-Slave I-Cache Controller

The Configurable Master-Slave I-Cache controller is designed for increasing hit rate efficiently in large range of jump. The Configurable Master-Slave I-Cache controller algorithm is shown in Fig. 3-4.

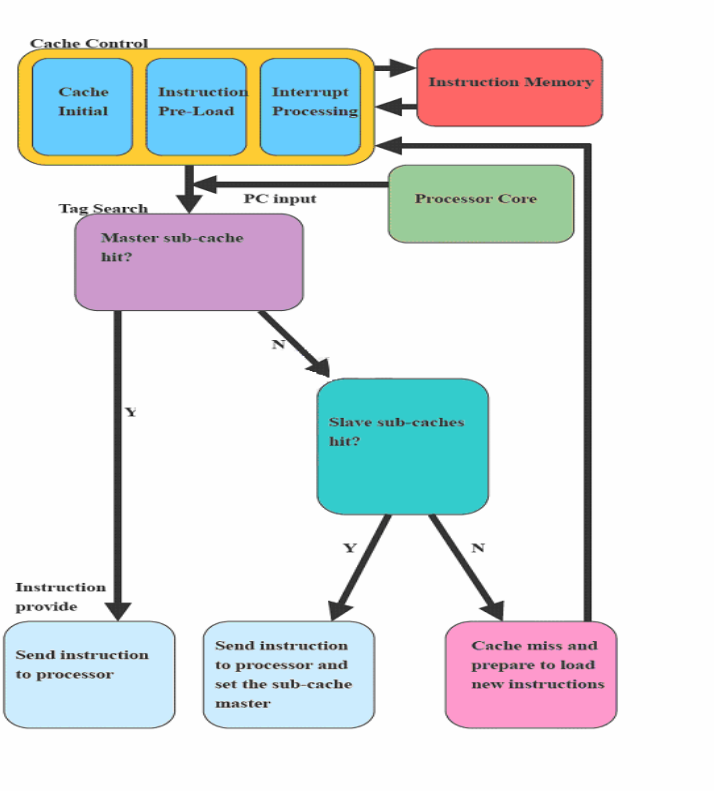


Fig. 3-4. The Configurable Master-Slave I-Cache controller algorithm.

3.2.2 The Performance of Configurable Master-Slave I-Cache

Fig. 3-5 shows the total performance improvement in different kinds of CR_Ratio.

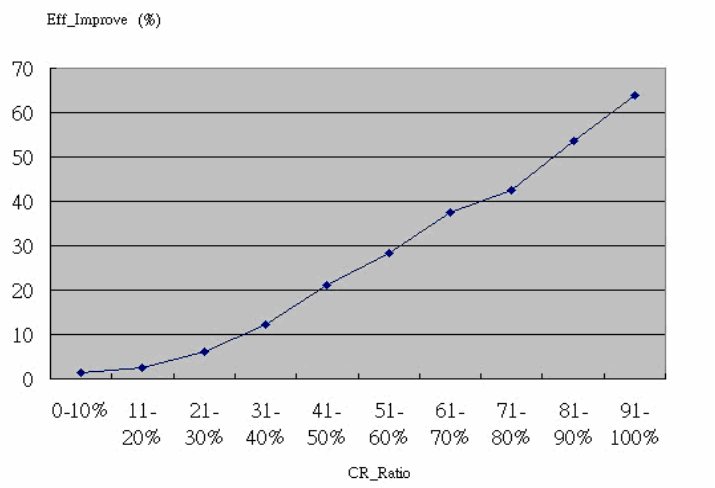


Fig. 3-5. The improvement of MS-cache.

CR_Ratio: The ratio of returnable jump in total jump instructions

Eff_Improve: A parameter of total performance improvement.

When CR_Ratio increases, the value of Eff_Improve increases obviously.

On the other hand, MS-cache uses the architecture of phased cache so that it can reduce 44% of power dissipation.

3.3 High performance pipeline design of low power phased cache

High performance pipeline design of low power phased cache is combined phased cache with specific pipeline. It takes advantages of that it can eliminate the set associate cache power and access the cache data one stage early by specific pipeline. Our approach can reduce 44%~70% (2 ~ 4way) cache power consumption without any time latency and only cost 6% total gate count in implementation.

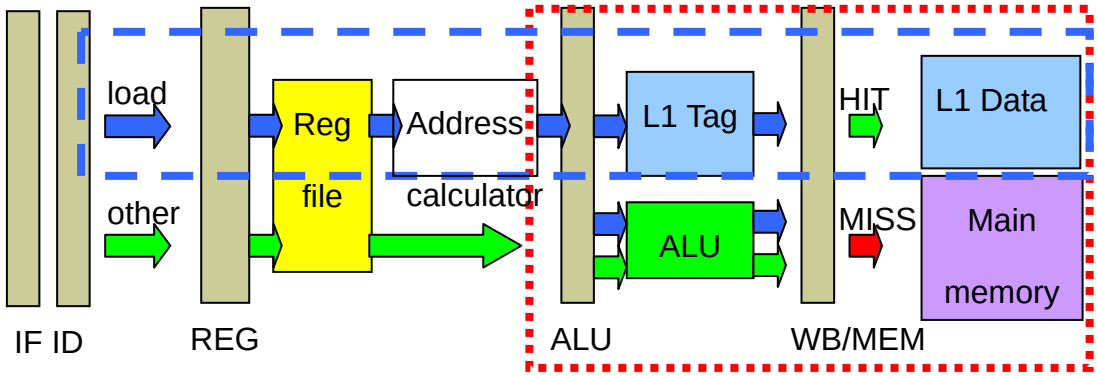


Fig. 3-6. The architecture of High performance pipeline design of low power phased cache.

Fig. 3-7 reports the results of cache access cycle and total performance by Simplescalar. The time consumption of cache access is reduced 38% and power

consumption is reduced 40% - 70%.

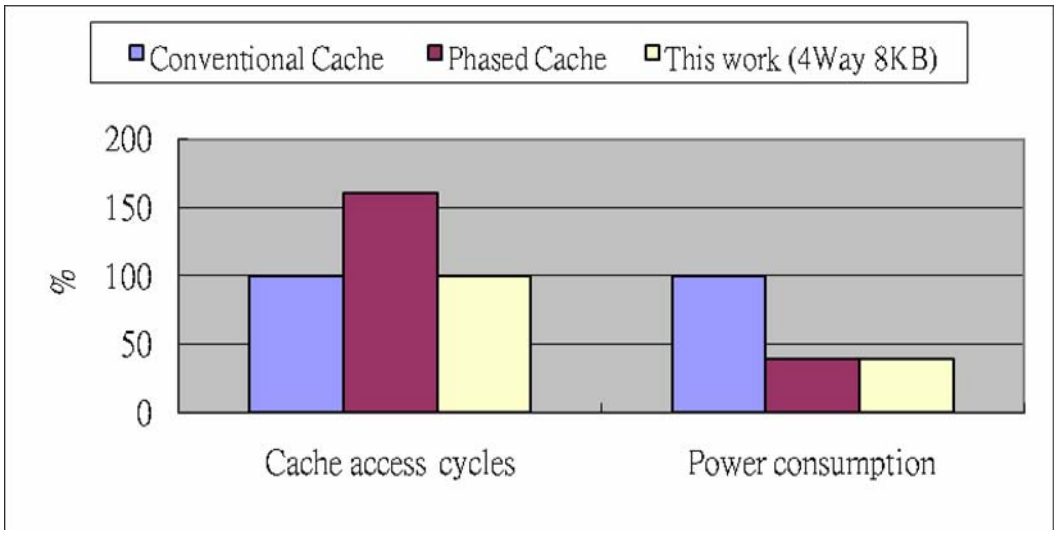


Fig. 3-7. Cache access cycles & Power consumption.

3.4 Tool Chain

3.4.1 Assembler

The GUI assembler supports machine code translation, program ROM generation and debug information. User can debug and generate test bench by above information. The assembler figure is shown in Fig. 3-8.

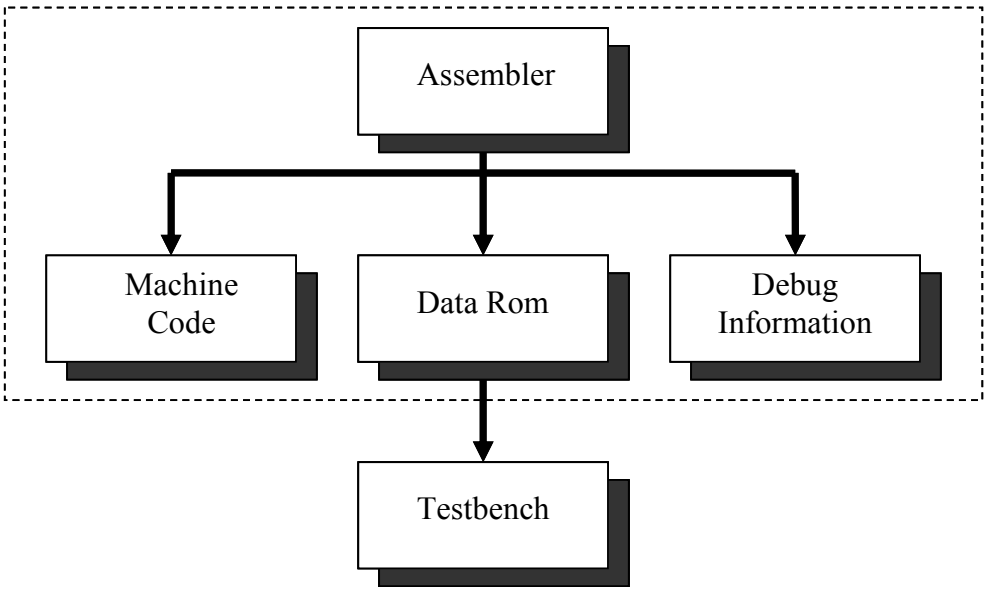


Fig. 3-8. The assembler Figure.

We implemented the tool based on Visual C++ language in Fig. 3-9. The assembler generates files:

Pop.txt : Hexadecimal program code for testing chip.

Bin.txt : Binary program code for simulation.

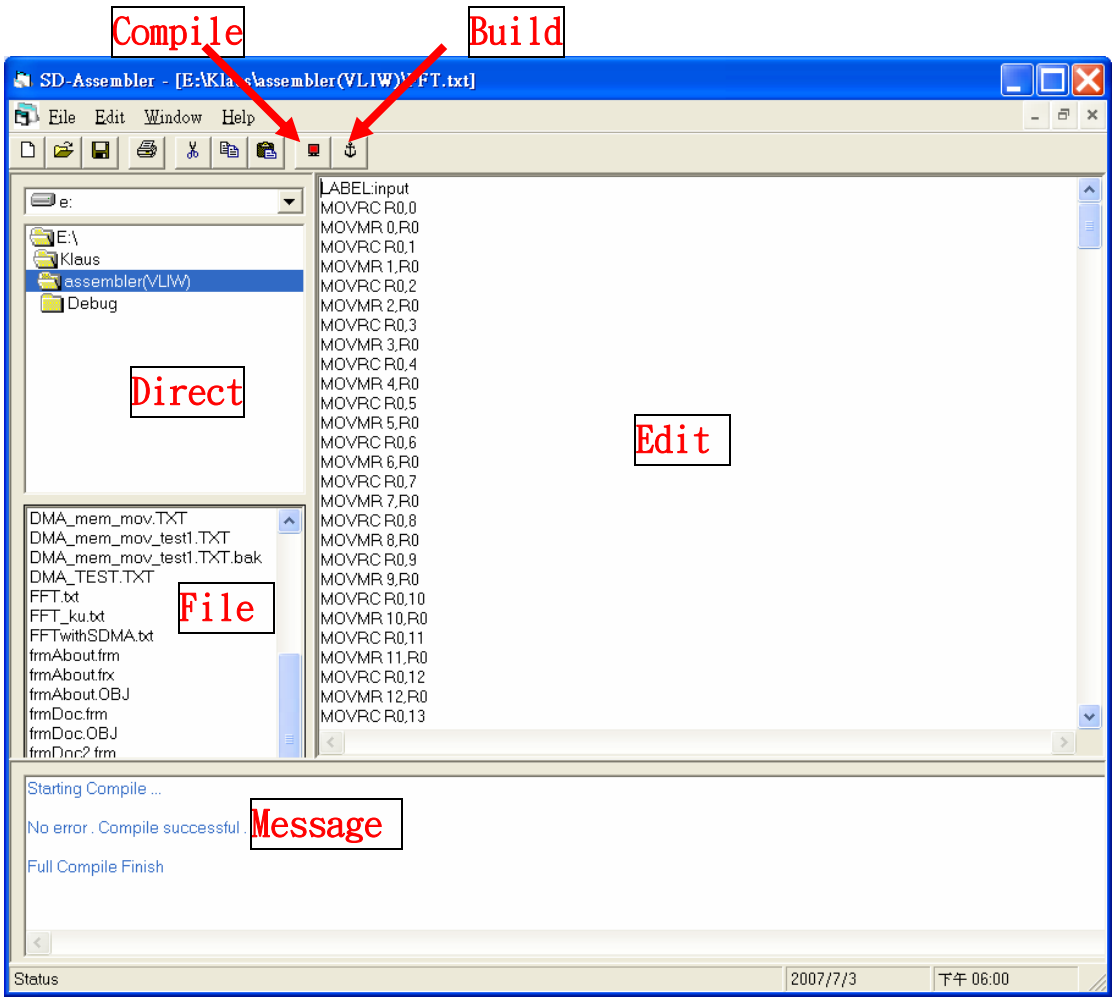


Fig. 3-9. Assembler Interface.

3.4.2 Simulator

Our thesis provides a simulator implemented by Visual C++ language for different kinds of test patterns. We apply a method like software pipeline [13] for simulator so that each iteration is arranged in inverse order. An example for five pipeline RISC architecture is in Fig. 3-10. All stages sort in inverse order.

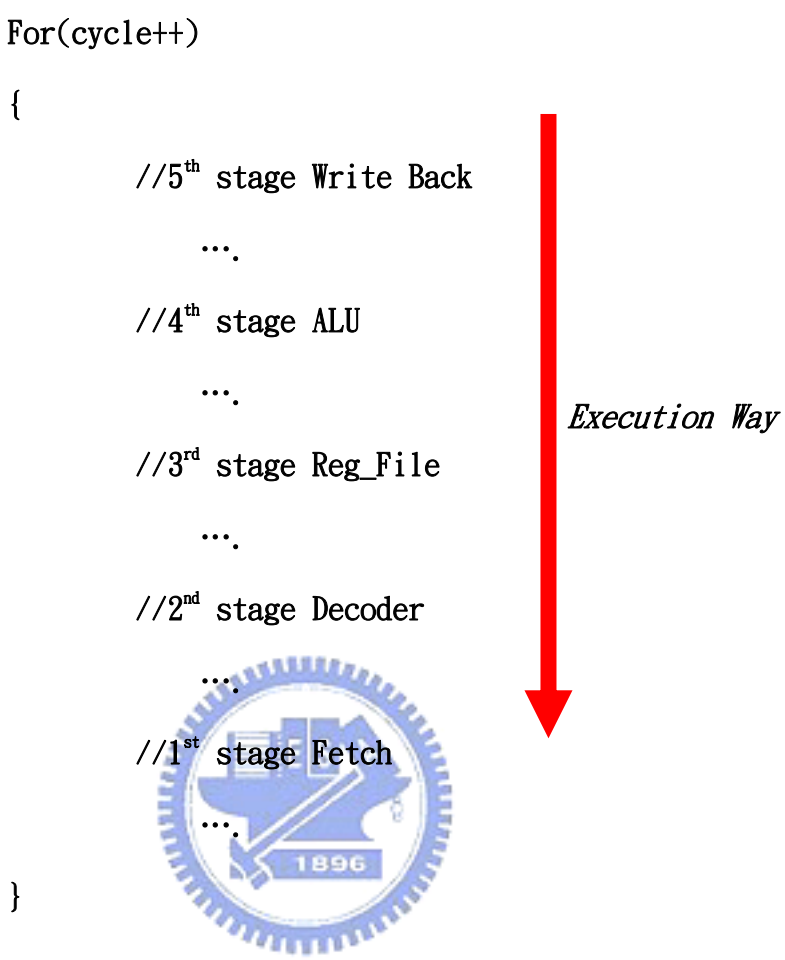


Fig. 3-10. Software pipeline design flow.

The simulator provides the ability to view register value and memory content and calculate the number of hazard and total penalty cycle.

These information can help programmer to analyze performance and debug easily. In Fig. 3-11, it shows assemble code, memory data, register value, total cycle count and total instruction count.

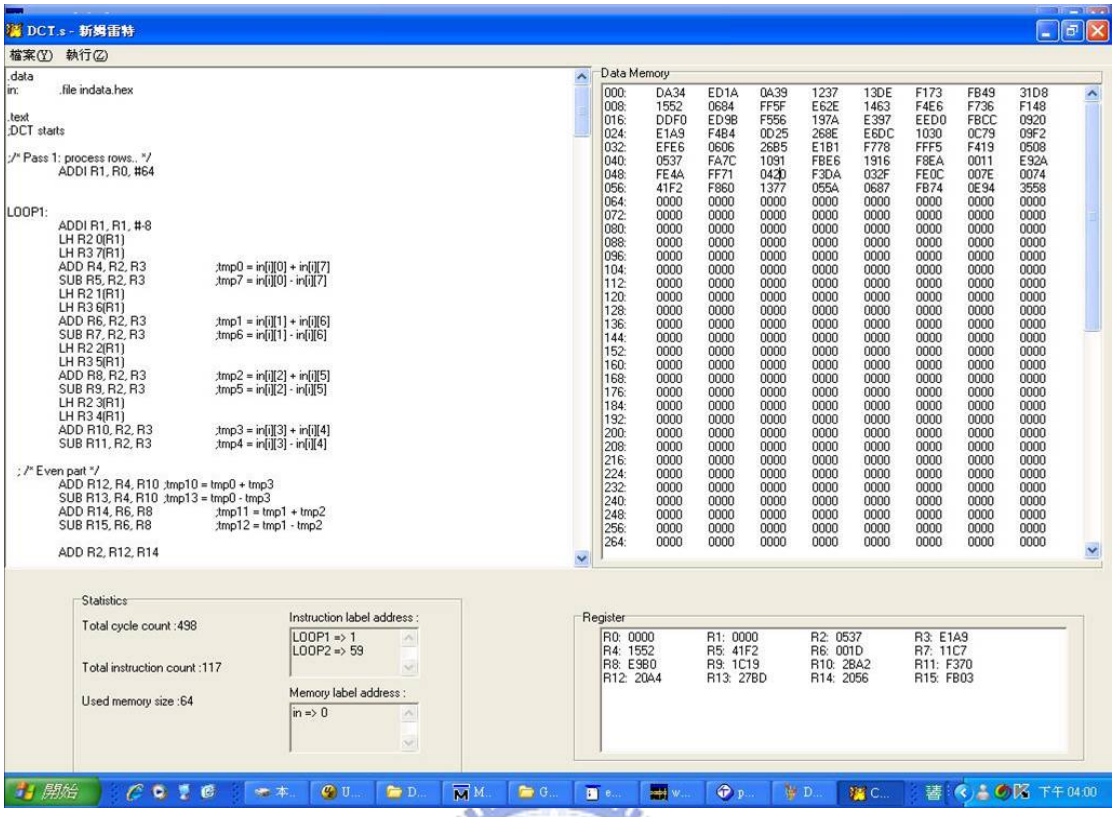


Fig. 3-11. The simulator interface.

3.5 Verification

In order to respond ISS(Information Systems and Sciences), our processor use some test pattern including F.I.R (Finite Impulse Response) ,D.C.T (Discrete Cosine Transform) and Sobel operator and the result of simulator to verify our processor's function. We will introduce three kinds of test pattern and these results in the following paragraph.

3.5.1 Finite Impulse Response

FIR filtering is a general application in communication and multi-media field. Fig. 3-12 shows the 16 tap impulse response FIR filter.

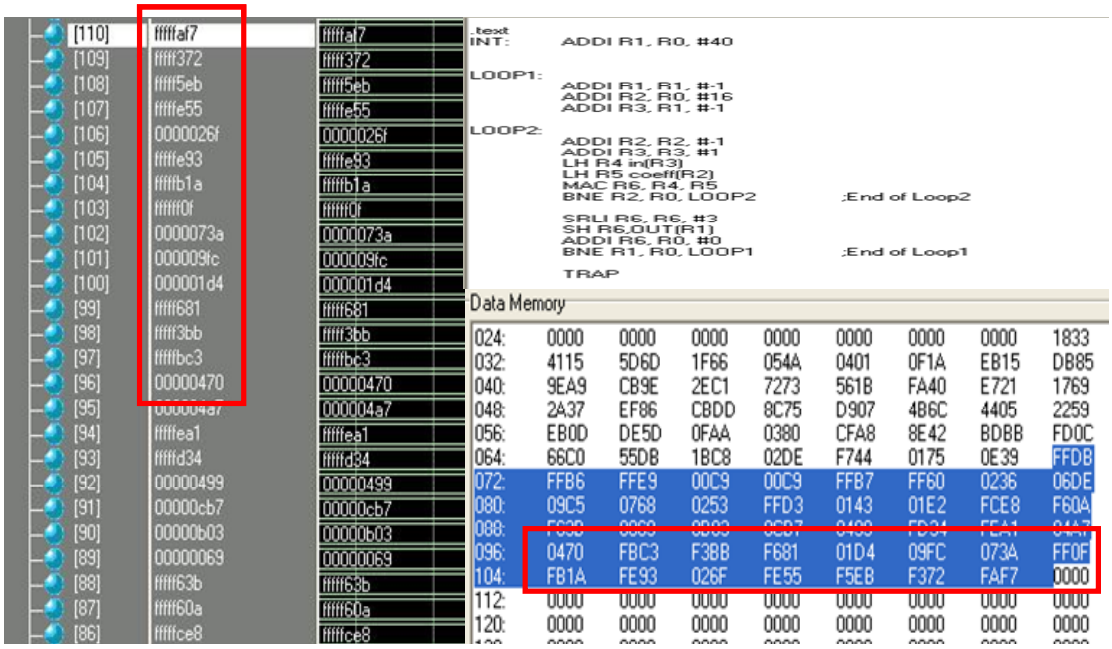


Fig. 3-12. FIR RTL simulation and simulator result.

For verify our proposed codec performance, we supports a module to calculate switch activity which data to external memory on bus. In Fig. 3-13, our proposed method can reduce 46.13 % of switch activity on data bus.

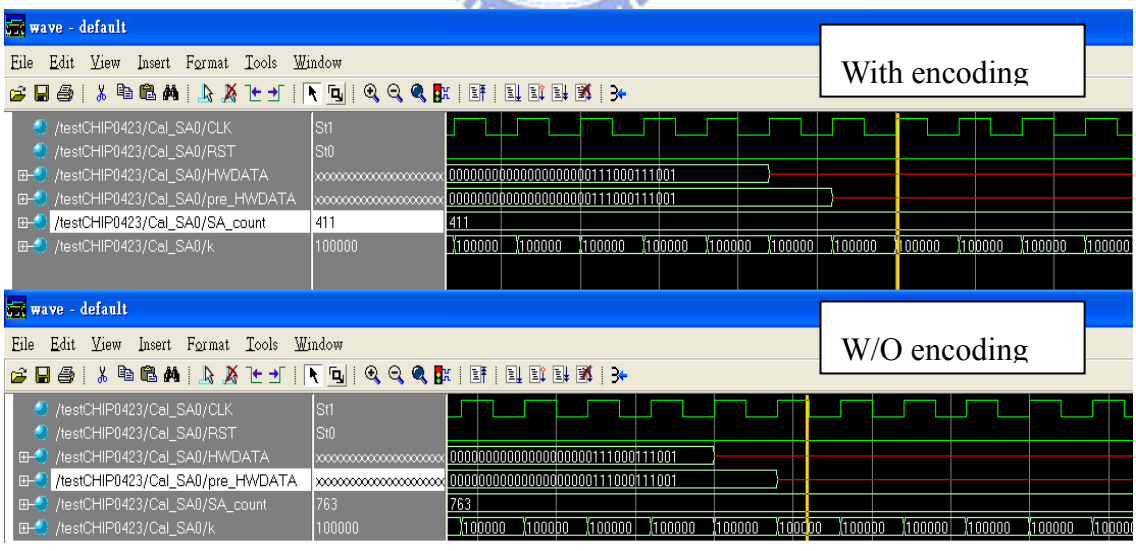


Fig. 3-13. Switch activity for FIR.

3.5.2 Discrete Cosine Transform

The 8 by 8 1-dimensional DCT algorithm is shown in Fig. 3-14. The 8 by 8 2-dimensional DCT is implemented by applying 1-dimension DCT row-by-row and column by column. The simulation result is shown in Fig. 3-15.

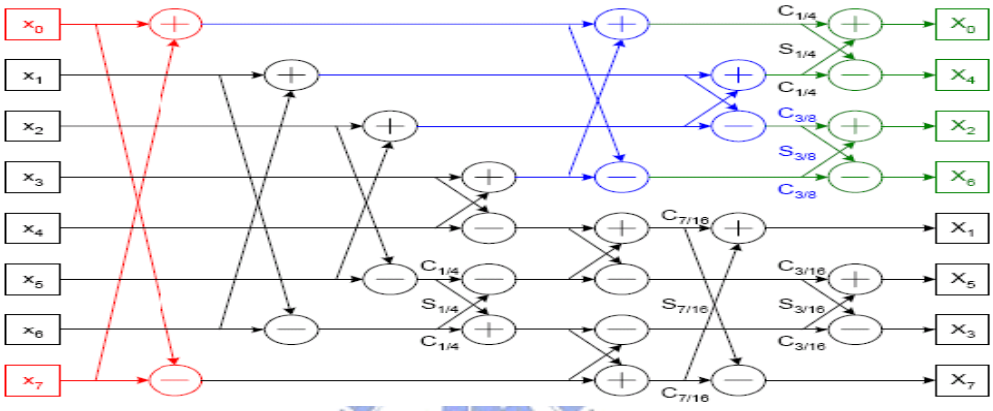


Fig. 3-14. 1 dimension 8 by 8 DCT.

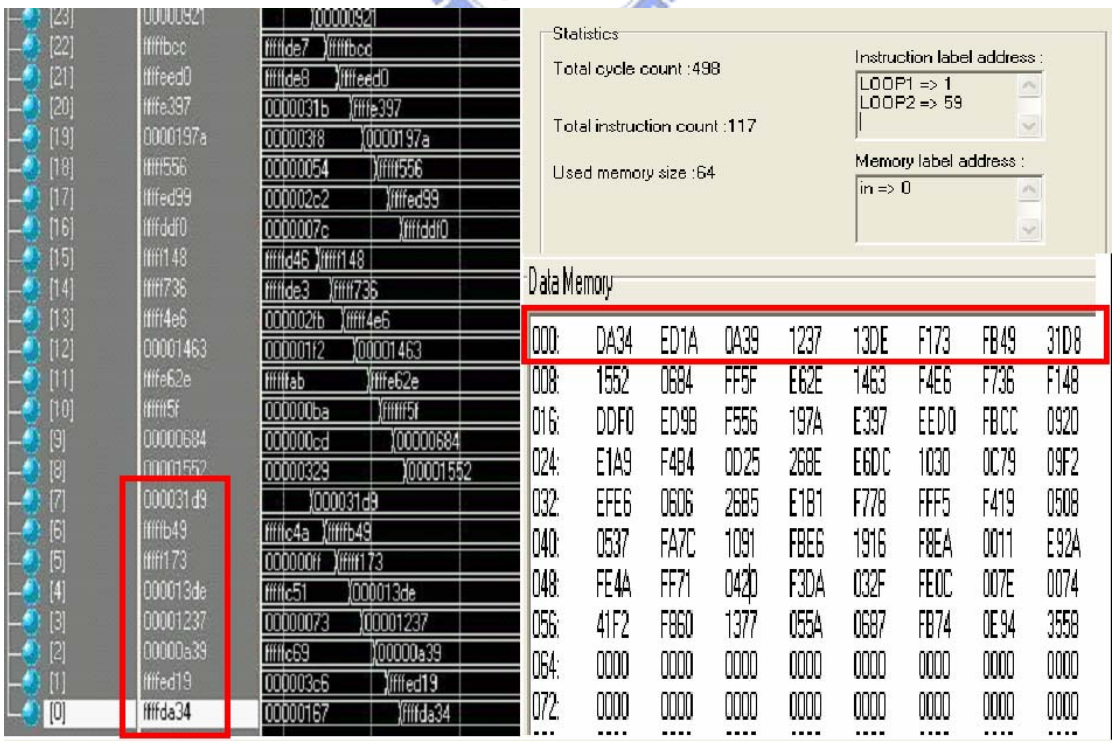


Fig. 3-15. 2 dimension 8-8 DCT RTL simulation and simulator result.

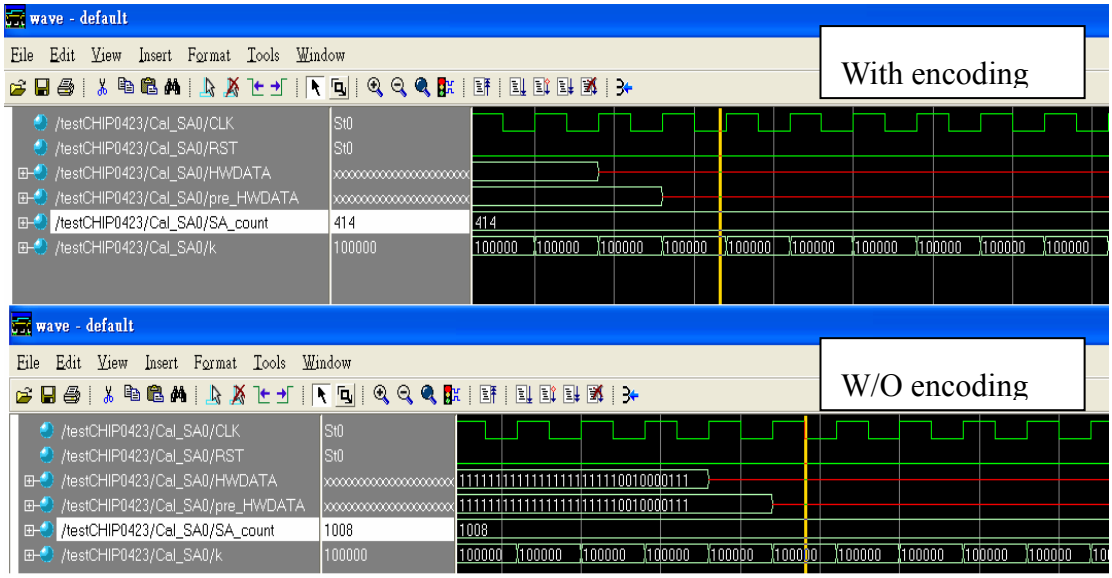


Fig. 3-16. Switch activity for DCT.

In Fig. 3-16, our proposal can reduce 58.92 % of switch activity on data bus.

3.5.3 Sobel Operator

We use Sobel operator to verify the large data moving in data cache. The Sobel operator is an edge detection algorithm in image processing. It is a discrete differentiation operator technically and gets the gradient of the image intensity function. At each point in the image, the result of the Sobel operator is either the corresponding gradient vector or the norm of this vector.

Sobel operator computes approximations of the derivatives for horizontal and vertical changes by using two 3x3 array which are convolved with the original image. We define A as the source image, G_x and G_y are two images which contain the horizontal and vertical derivative approximations. The equation is as:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * A \text{ and } G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A \quad (3\text{-}1)$$

And then we calculate root mean square value to get the resulting gradient approximations, using

$$G = \sqrt{G_x^{\,2} + G_y^{\,2}} \quad (3\text{-}2)$$

We implement edge detection for 64x64 pixels image in our processor and compare with MATLAB in Fig.3-17.

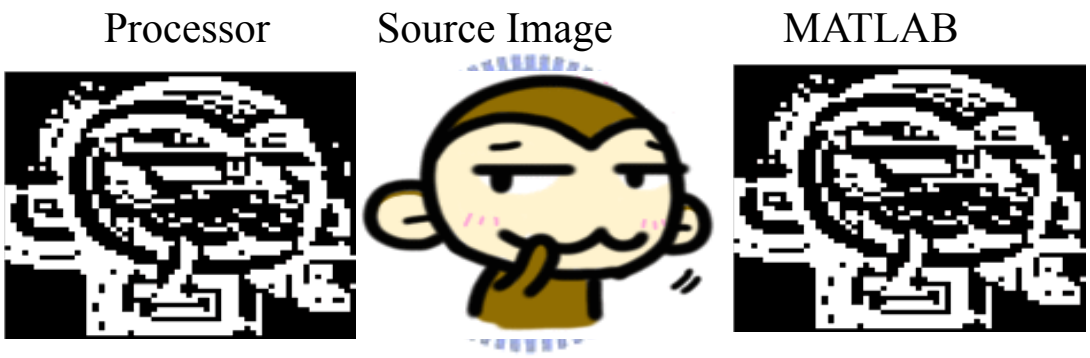


Fig. 3-17. Sobel Operator simulation.

3.6 Field-Programmable Gate Array (FPGA)

In the internet product, communication, industry system and electrical system field, they usually use FPGA to design their chip because FPGA has high flexibility so that user can make their logic function from Boolean function, register function, embedded memory and complex functional IP. In the other hand, it can easily place and fit in the platform for different kinds of application.

The processing time in FPGA is slower than ASIC. However, FPGA has the

advantages include a shorter time to market, ability to re-program in the field to fix bugs. The designs are developed on regular FPGAs and then migrated into a fixed version that more resembles an ASIC.

We use Altera APEX20KE EP20K1500EBC652-1X and Quartus Ⅱ to floorplan, place, and route.

The Sobel operator simulation in FPGA and Matlab simulation is shown in Fig. 3-15

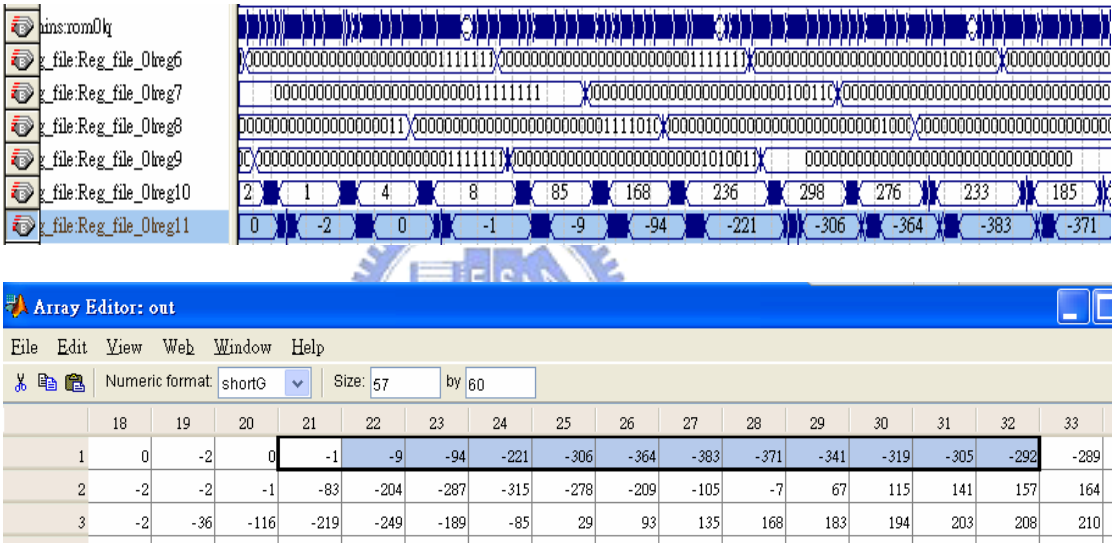


Fig. 3-18. The Sobel operator result in FPGA and Matlab.

3.7 Summary

In this chapter, we introduce a RISC embedded processor combined with configurable Master-Slave cache controller and High performance pipeline design of low power phased cache. The system not only prove the correctness of the algorithm but also take advantages of the ability of specific designs to reduce total power dissipation.

In the next chapter, we will integrate it into a SOC chip.

Chapter 4
Chip Implementation and Verification Results

4.1Chip Fabrication

In this chapter, we will debate the issues of the chip implementation and verification results.

4.1.1. Chip Design Flow

We refer to CIC cell-based design flow as shown in Fig. 4-1 for our chip implementation and verification.

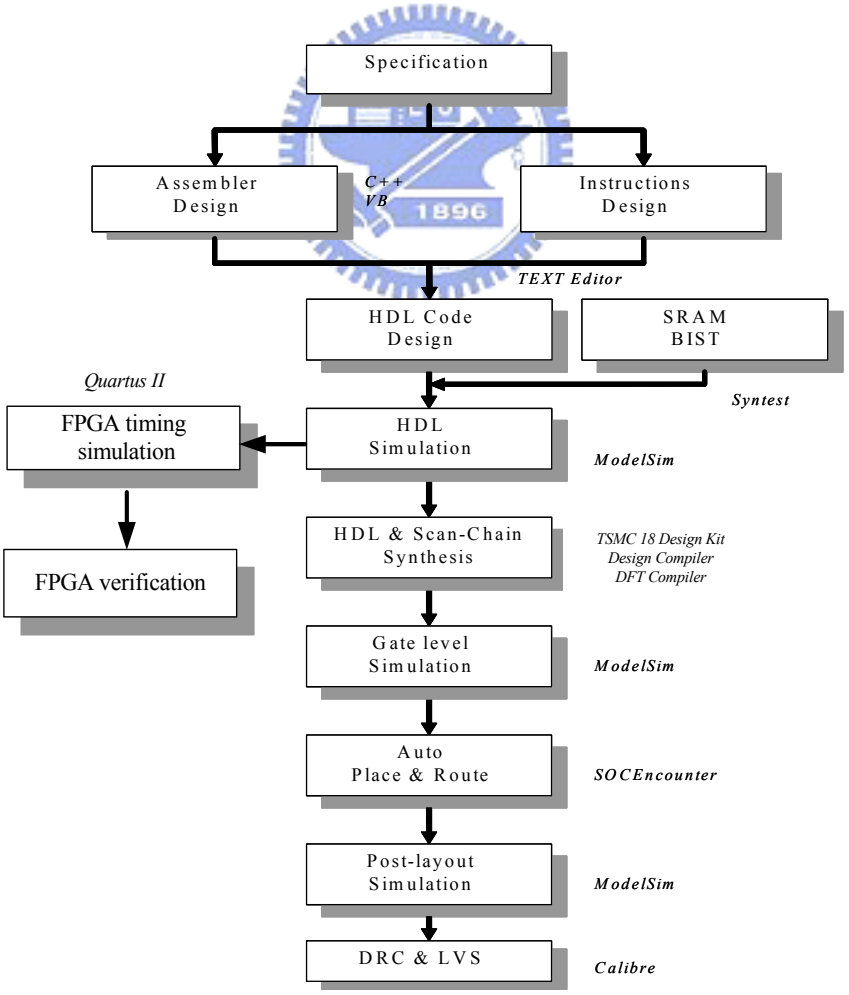


Fig. 4-1. Cell-based design flow.

4.1.2. Synthesis

The behavior level hardware description language (HDL) of the embedded processor has been pre-simulated by Mentor-ModelSim and synthesized by Cadence-Design Compiler, where the timing constraint is 10 ns. The synthesis report using TSMC 0.18um CMOS technology is summarized in Table 4-1, where the 98.25% fault coverage is measured.

Table 4-1 Synthesis Report

ITEM	Area (mm^2)	Timing	Fault coverage
Processor	497524.31	10 ns	98.25 %

4.1.3. Auto Placement and Routing (APR)

Next, we use Cadence-SOC Encounter to place and route the gate level code automatically.

Table 4-2 APR Report

Chip name	Bio-CAS Processor Version 1 (BP_v1)
Technology	TSMC 0.18um 1P6M CMOS
Package	160 CQFP
Chip Size	2.114× 2.114 mm^2
Gate Count	47K gate count (Bus codec: 2.014K)
Power Dissipation	~16mW
Max. Frequency	100MHz （10 ns）

We use test patterns include DCT, FIR and Sobel with PrimePower. The average

of power dissipation is 16mW. The chip layout is shown in Fig.4-2. The chip pin description and 160 pin-CQFP bounding diagrams are revealed in Figs. 4-3, 4-4, respectively.

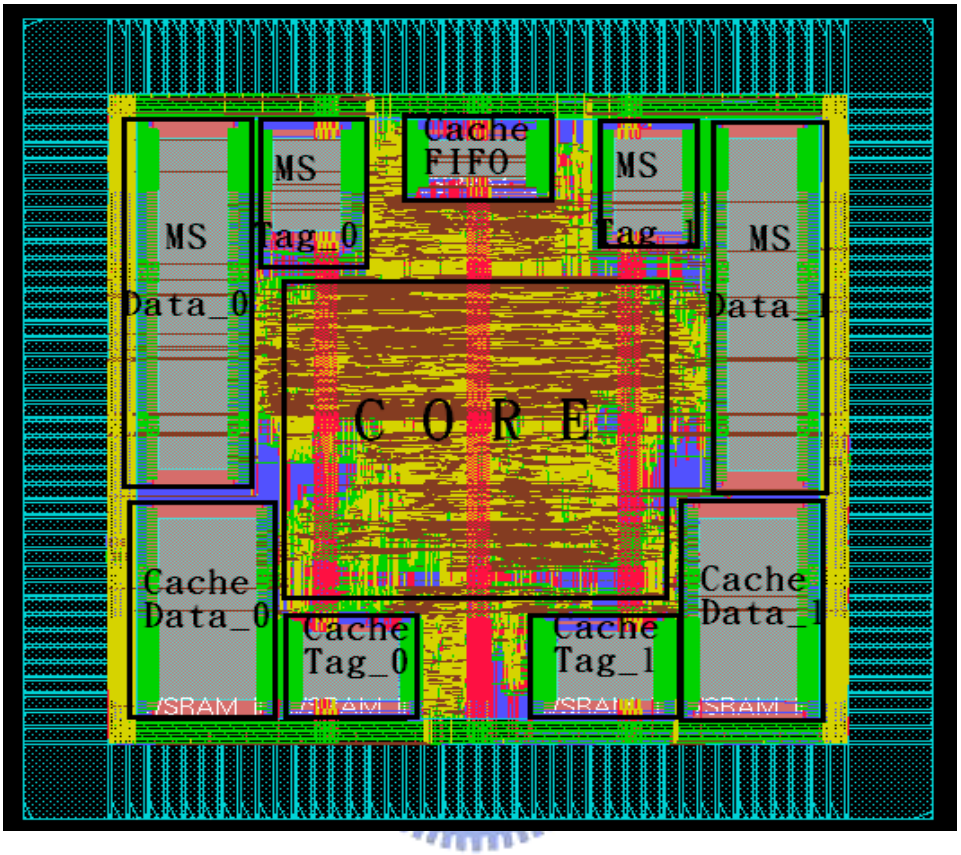


Fig. 4-2. Chip Layout.

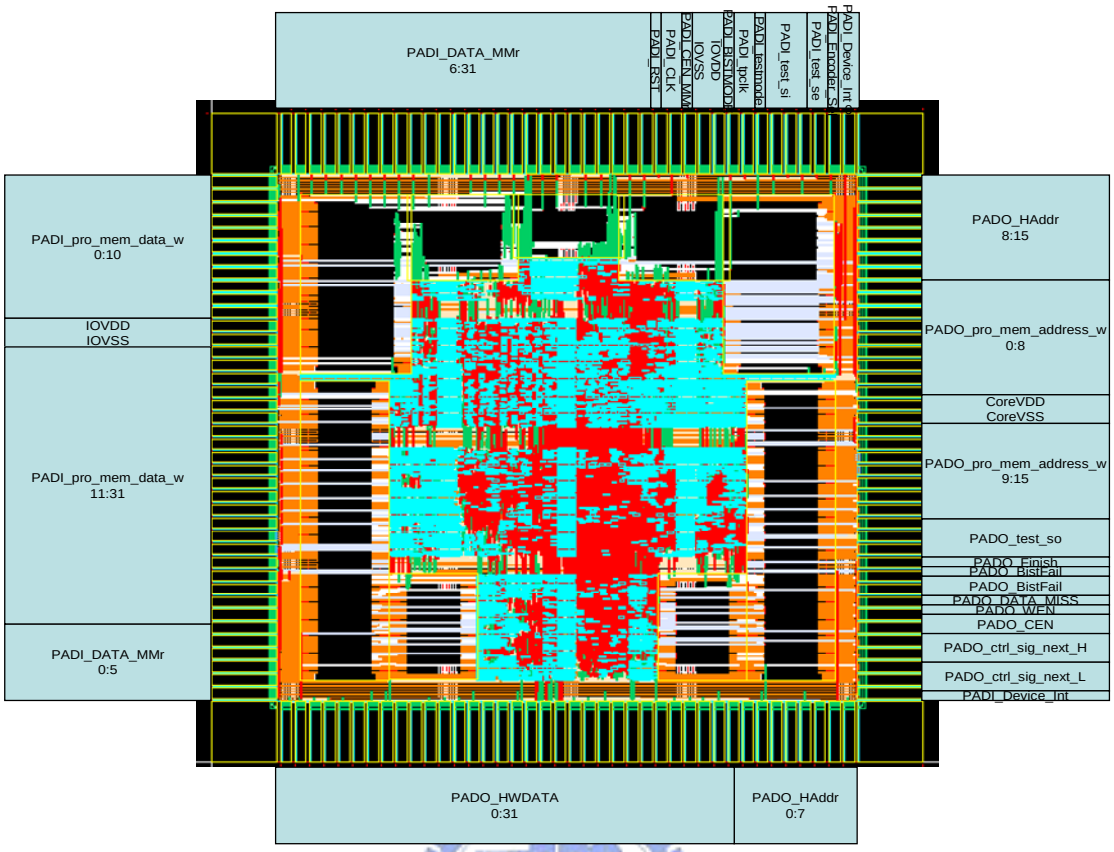


Fig. 4-3. Chip Pin Description Diagram.

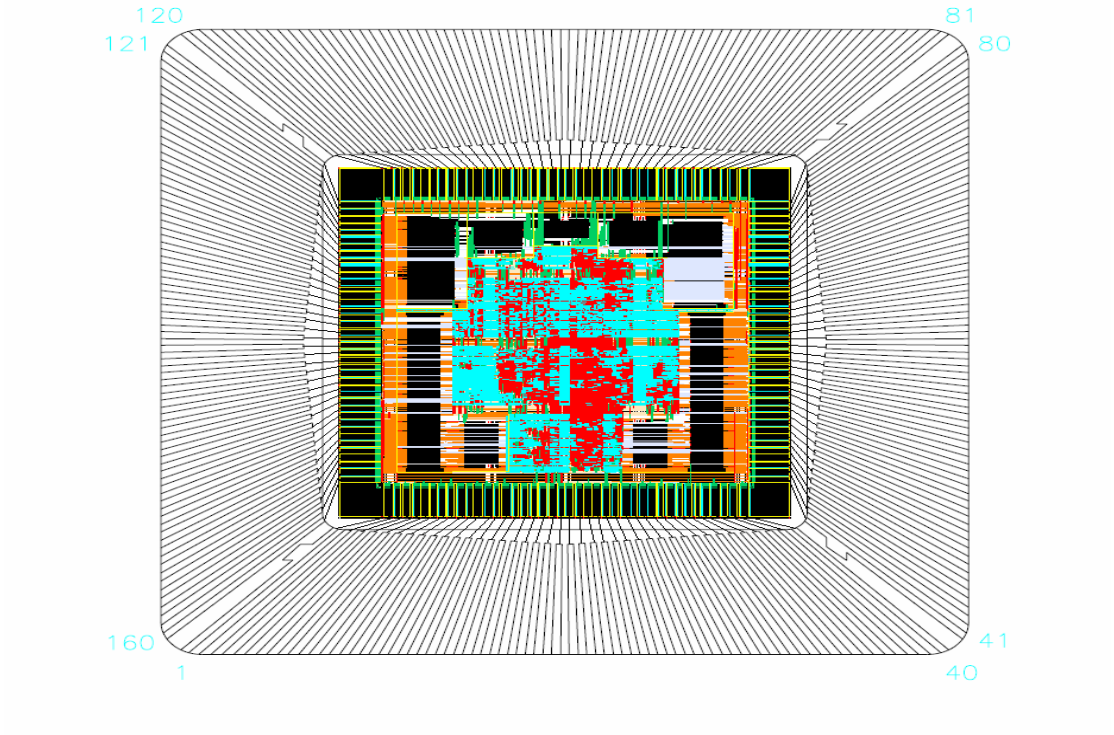


Fig. 4-4. 160 pin-CQFP Bounding Diagram.

Table 4-3 Chip Specification

Technology	Description
Process	TSMC 0.18μm 1P6M Mixed Signal
Architecture	7-stage pipeline
Synthesis	Synopsys Design Compiler
Gate Count	47K (Bus codec: 2.014K)
Embedded Memory	Cache_FIFO RAM(512x2)x1, Cache_DATA RAM(512x32)x2, Cache_TAG RAM(512x8)x2, MS_Cache RAM(512x32)x2, MS_Cache_TAG RAM(512x8)x2
Die size	2.114×2.114 mm^2
Supply	1.8V/3.3V ± 10%
Input Delay Time	Max 0.714ns/ Min 0.543ns
Power consumption	16 mW
Operating Frequency	100 MHz

The chip characteristics are listed in Table 4-2, where the power consumption is 16 mW.

Finally, we still use Calibre DRC（Design Rule Check）and LVS （Layout VS Schematic）for our final check.

In memory part, we generate 4 tag cache for address tag, 4 data caches for data by Cadence-Memory generator. The gate-level timing diagrams of DCT and Sobel benchmark are shown in Fig.4-5 and Fig. 4-6, respectively.

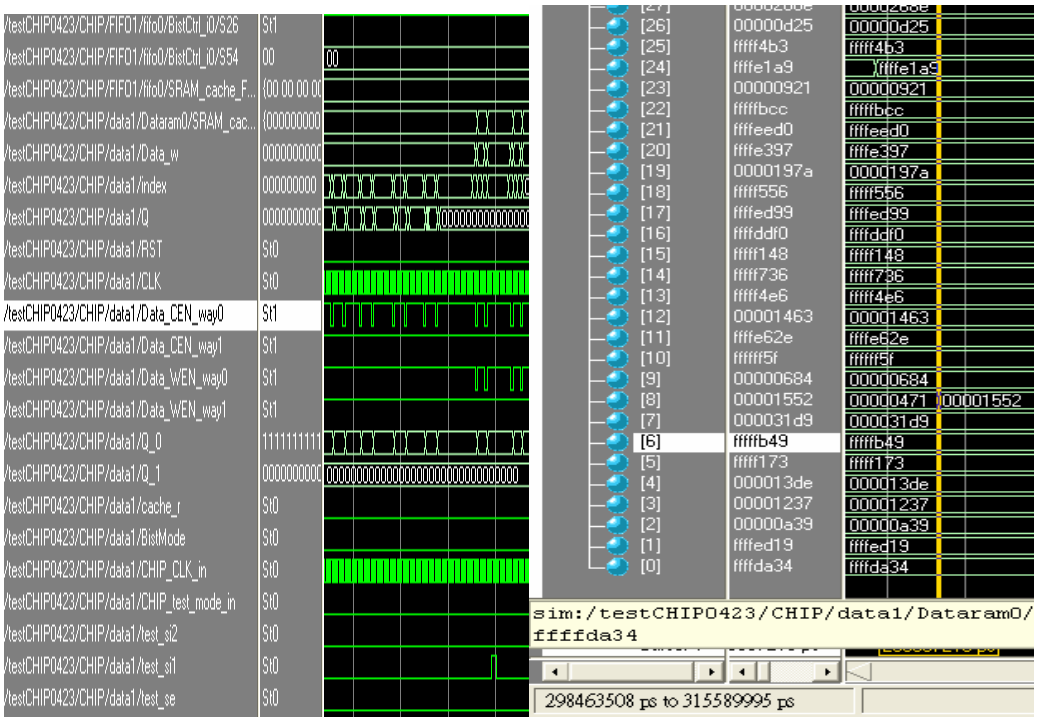


Fig. 4-5. DCT gate-level simulation.

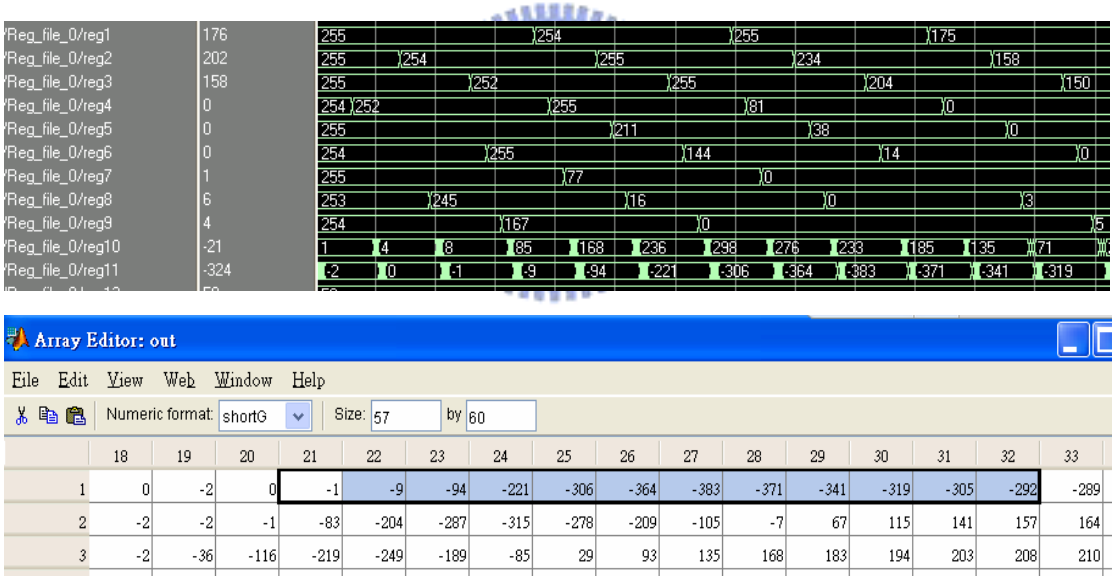


Fig. 4-6. Sobel gate-level simulation.

4.2 Power Analysis

The transparent un-coding power dissipation at the bus is given by (4-1) where P is power consumption, C_L is the bus capacitance per bit, α is transition activity, V is supply voltage, f is clock frequency.

$$P_{D,uncoded} = \alpha_{uncoded} C_L V^2 f \tag{4-1}$$

For our proposed encoding and decoding scheme, the power dissipation at the encoder, the bus, and the decoder is given by (4-2).

$$P_{D,coded} = P_{D,encoder} + P_{D,bus} + P_{D,decoder} \tag{4-2}$$

If α_{coded} is the reduced transition activity at he bus after encoding then the total power dissipation is given by (4-3).

$$P_{D,coded} = P_{D,encoder} + \alpha_{coded} C_L V^2 f + P_{D,decoder} \tag{4-3}$$

We estimate the power dissipation in the encoder and the decoder are 0.8mW by DCT function.

In Fig. 4-7 we plot the power dissipation for different bus capacitances, C_L. The power dissipation was estimated using 75 samples of DCT data to memory. We see that for small bus capacitances (<0.3415 pF/bit) in Fig. 4-8, it is best to not encode the data at all. For capacitances above 0.3415 pF/bit, our proposed scheme provides a well reduction in power dissipation. The slope of the curve is determined only by the reduced transition activity, α_{coded}, and is independent of the power dissipation in the encoder and decoder.

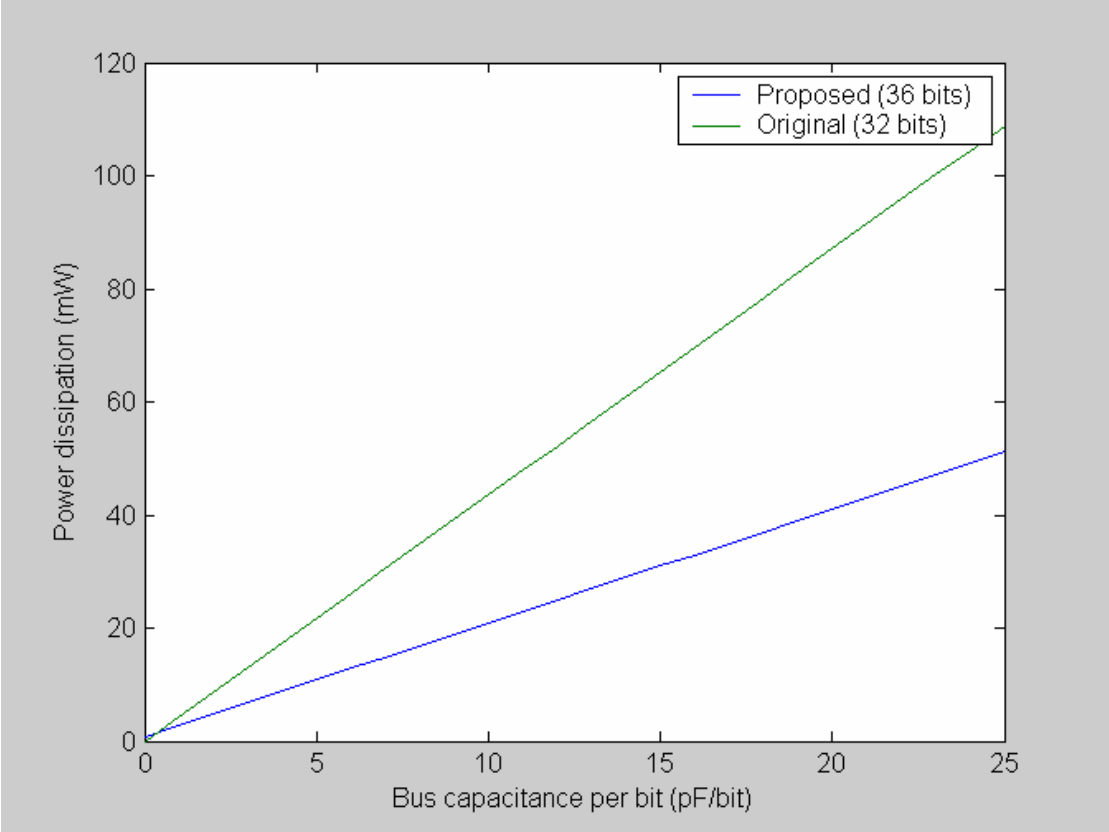


Fig. 4-7 Power dissipation for Proposed and Original

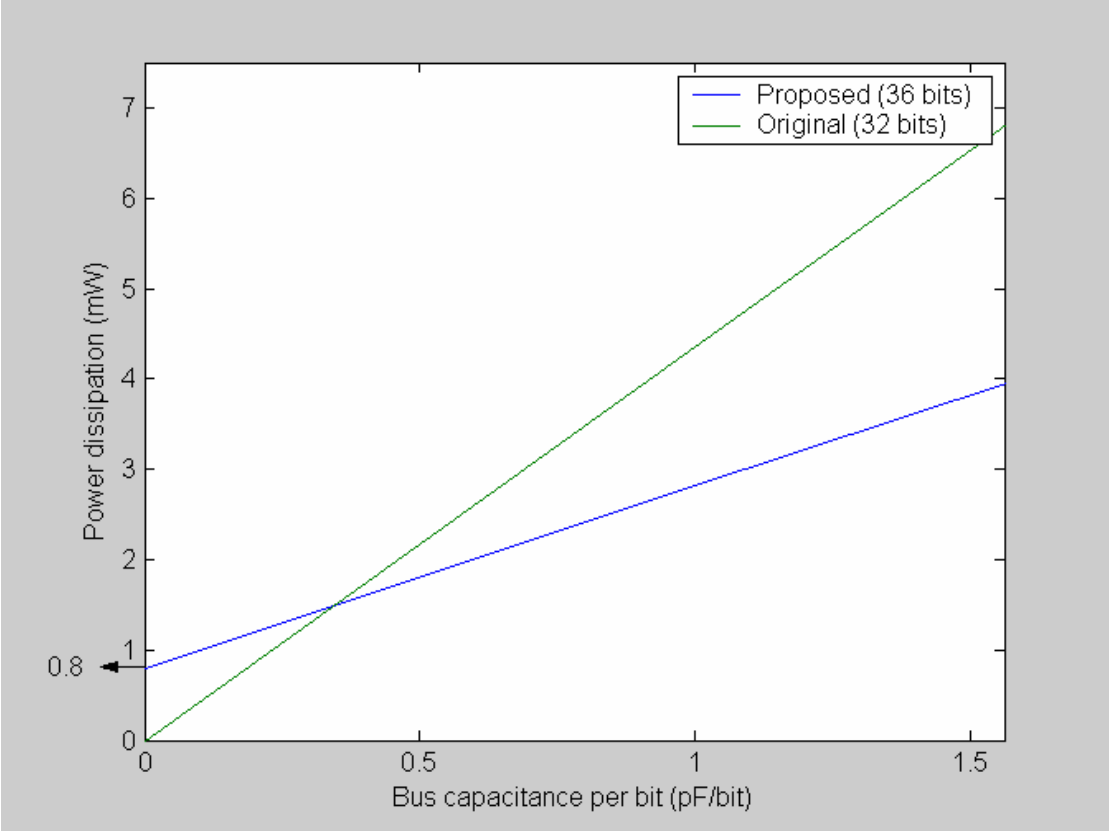


Fig. 4-8 Power dissipation for Proposed and Original

Chapter 5
Conclusions and Future Works

In this thesis, we proposed a power aware data bus codec design that can reduces switch activity by 23% on average since adaptively choosing the optimal encoding scheme for different data types. The advised codec can save 68% area overhead compared with R-S-H's design. From the FIR and DCT benchmark simulations, on average, 50%~60% power reductions can be guaranteed. In near future, the processor design adopting low power bus codec and phase-cache pipeline schemes will be tape out for completeness. At that time, we will integrate the proposed scheme with the operation amplifier, filter, and analog-to-digital converter for biomedical SOC design.

References

[1] S. Takahiro, S. AKUI, K. SENO, M. NAKAI, T. MEGURO, "Dynamic voltage and frequency management for a low-power embedded microprocessor," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 1, pp.28-35, January, 2005.

[2] A Khan, P Watson, G Kuo, D Le, T Nguyen, S Yang, P , "A 90-nm power optimization methodology with application to the ARM 1136JF-S microprocessor," *IEEE Journal of Solid-State Circuits*, vol. 41, issue 8, pp. 1707- 1717, August, 2006.

[3] Qing Wu, M. Pedram , Xunwei Wu, "Clock-gating and its application to low power design of sequential circuits," *IEEE Circuits and Systems* I, vol. 47, issue 3, pp. 415-420, Mar, 2000.

[4] L. Benini, P. Siegel, G. De Micheli, "Saving power by synthesizing gated clocks for sequential circuits," *IEEE Design & Test of Computers,* vol. 11, no.4, pp.32-41, October, 1994.

[5] P. Petrov, A. Orailoglu, "Low-power data memory communication for application-specific embedded processors," *Proceedings of the 15th international symposium on System Synthesis (ISSS '02)* , pp. 219-224, October, 2002.

[6] Michael J. Flynn, Patrick Hung, "Microprocessor design issues: thoughts on the road ahead," *IEEE Computer Society*, vol.25, no.3, pp. 16-31, MAY–JUNE 2005.

[7] Ke Ning, David Kaeli, "Power aware external bus arbitration for system-on-a-chip embedded systems," *Proceedings of High performance embedded architectures and compilers*, vol. 3793, pp. 87-101, November, 2005

[8] A.P. Chandrakasan , S. Sheng , R.W. Brodersen , "Low power CMOS digital design," *IEEE Journal of Solid-State Circuits*, vol. 27, issue 4, pp473-484, April, 1992.

[9] A.P. Chandrakasan and R.W. Brodersen, "Minimizing power consumption in digital CMOS circuits," *Proceedings of the IEEE*, vol. 83, issue 4, pp.498-523, April, 1995.

[10] S. Komatsu, M. Ikeda, K. Asada, "Bus data encoding with coupling-driven adaptive

code-book method for low power data transmission," *IEEE Solid-State Circuits Conferen*ce, pp. 297-300, September, 2001.

[11] 鍾崇斌, "Low-Power Computer Design :Slides,"國立交通大學，Sep, 2006.

[12] M.R. Stan, W.P. Burleson, "Bus-invert coding for low power I/O,"*IEEE Transactions on VLSI systems*, 1995.

[13] Tina Lindkvist, Jacob Lofvenberg, Oscar Gustafsson, "Deeo sub-micron bus invert coding," *NORSIG*, 2004.

[14] Yan Zhang, John Lach, Kevin Skadron, "Odd even bus invert with two phase transfer for buses with coupling," *Proceedings of the 2002 international symposium on Low power electronics and design*, pp. 80-83, 2002.

[15] L.Benini, G.DeMicheli, E.Macii, D.Sciuto, and C.Silvano, "Asymptotic zero-transition activity encoding for address busses in low-power microprocessor-based systems," *IEEE 7th Great Lakes Symposium*, 1997.

[16] S. Ramprasad, N. R. Shanbhag, I. N. Hajj, "A coding framework for low power address and data busses," *IEEE Transitions VLSI Systems,* vol. 7, no. 2, pp. 212-221, June 1999.

[17] J Yang, R Gupta, "Frequent value locality and its applications," *ACM Transactions on Embedded Computing Systems (TECS)*, vol.1, issue 1,pp.79- 105, November, 2002.

[18] J Yang, R Gupta, "Frequent value encoding for low power data buses," *ACM*

*Transactions on Design Automation of Electronic Systems (TODAES),*vol.9, issue 3,pp. 354 - 384, July, 2004.

[19] M. Muroyama, A. Hyodo, T. Okuma, H. Yasuura, "A power reduction scheme for data buses by dynamic detection of active bits," *Digital System Design Proceedings*, pp.408- 415, September, 2003.

[20] Wolfgang Nebel, "System-level power optimization," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol.5, issue 2, pp.115–192, April, 2000.

[21] Simon Segars, "Low power design techniques for microprocessors," *ISSCC*, Feb, 2001.

[22] M. Sami, D. Sciuto, C. Silvano, V. Zaccaria, R. Zafalon, "Low-power data forwarding for VLIW embedded architectures," *IEEE VLSI Systems,* vol. 10, issue 5, pp.614-622, October, 2002.

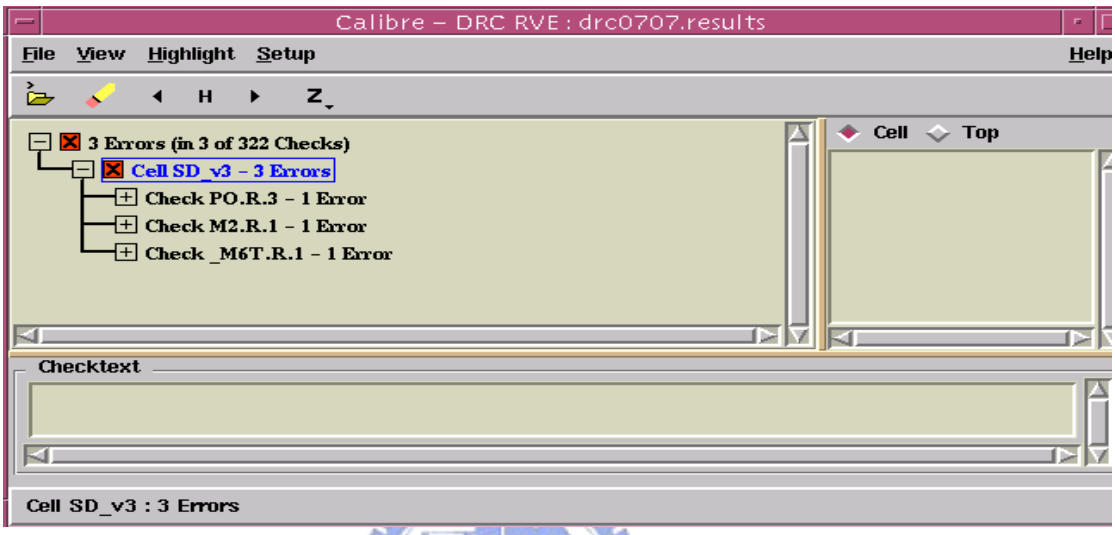
[23] Nian Shyang Chang, **Cell-Based IC Physical Design and Verification,** Chip Implementation Center, July, 2004.

[24]周經翔、林進燈, "具有使用者可調性主從式指令快取記憶體控制器: Design of Multi-Core Embedded Processor Using Configurable Master-Slave I-Cache Controller," 國立交通大學，碩士論文，中華民國 95 年

Appendix

A. DRC and LVS Verification

1. DRC



DRC 驗證無誤

2. LVS

```
                ######################################################
                ##                                                  ##
                ##        C A L I B R E   S Y S T E M               ##
                ##                                                  ##
                ##           L V S   R E P O R T                    ##
                ##                                                  ##
                ######################################################


REPORT FILE NAME:          lvs.rep
LAYOUT NAME:               SD_v3.spi (´SD_v3´)
SOURCE NAME:               SD_v3.spi (´SD_v3´)
RULE FILE:                 calibre.lvs
HCELL FILE:                (-automatch)
CREATION TIME:             Sun Jul  8 19:54:22 2007
CURRENT DIRECTORY:         /user/ku/cachetest/calibra/LVS
USER NAME:                 ku
CALIBRE VERSION:           v2004.2_5.19    Tue Jun 29 19:44:37 PDT 2004


                         OVERALL COMPARISON RESULTS


                      #       ####################       _   _
                    #  #      #                  #      *   *
                  #    #      #     CORRECT      #        |
                   # #        #                  #       \___/
                    #         ####################
```

LVS 驗證無誤

B. CIC Tapeout Review Form

1. 晶片概述：

1-1. 專題名稱：具有快取及匯流排低功耗設計之嵌入式處理器

1-2. Top Cell 名稱：SD_v3

1-3. 使用 library 名稱：

___ CIC_CBDK35

___ CIC_CBDK25

v CIC_CBDK18

版本: *v1.0*

1-4. 是否使用CIC提供之Memory？ *Yes*

1-5. 工作頻率：*100 MHz*

1-6. 功率消耗：*16mW*

1-7. 晶片面積：*2114* X *2114*

2. 設計合成：

2-1. 使用之合成軟體？*Synopsys design compiler*

2-2. 是否加入 boundary condition：

v input drive strength、*v* input delay、*v* output loading、*v* output delay

2-3. 是否加入 timing constraint：

v specify clock (sequential design)

___ max delay、___ min delay (combinational design)

2-4. 是否加入area constraint？ *Yes*

2-5. 合成後之report是否有timing violation？*No*

___ 有 setup time violation、___ 有 hold time violation

2-6. 合成後之verilog是否含有assign描述？ *No*

2-7. 合成後之verilog是否含有 *cell* 之instance name？ *No*

2-8. 合成後之verilog是否含有反斜線 \ 之instance name或net name？ *No*

3. 可測試性設計(前瞻性晶片必填)：

3-0. 使用之設計軟體？*DFT compiler*

3-2. 使用之ATPG軟體？*Tetramax*

3-3. 使用Embedded memory數量: SRAM *5* ，ROM *0*

Memory大小：*512x32 (Word x bit)x2 512x8 (Word x bit)x2 512x2 (Word x bit)x1*

測試方法: BIST *Yes* ，or 其他測試方法 *N/A*

若使用BIST,其Test Algorithm為何? *Moving Inversion (13N March)*

同時有多個memory，是否共用BIST controller *Yes* ，BIST controller數量 *1*

3-4. Scan Chain Information

Flip-Flop 共有多少個？ *2280*

Scan chain 的數量共有多少條？ *3*

Scan chain length (Max.) ？ *25561.840*

3-5. Uncollapsed fault coverage是否超過 90% ？ *Yes* ，為多少？ *98.24%*

ATPG pattern的數目為多少？ *272*

註：若使用 Synopsys TetraMAX 來產生 ATPG pattern，請使用 set faults -fault_coverage 指令指定 TetraMAX 產生 fault coverage information

若使用 SynTest TurboScan 之 asicgen 來產生 ATPG pattern，請以 atpg pessimistic fault coverage 的值為準

4. 佈局前模擬

4-1. gate level simulation是否有timing violation？ *No*

___ 有 setup time violation、___ 有 hold time violation

5. 實體佈局

5-1. 使用之P&R軟體？ ___ Apolo、*v* SE

5-2. power ring寬度？ *8* 是否已考量current density(1mA/1um)？ *Yes*

5-3. 是否考慮output loading？ *Yes*

5-4. 是否加上Clock Tree？ *Yes*

5-5. 是否加上Corner pad？ *Yes*

5-6. 是否加上 IO Filler？ *Yes*

5-7. 是否加上 Core Filler？ *Yes*

5-8. 是否上加 Bonding Pad？ *Yes*

以下(A-1)為使用 Apollo 者才須回答

A-1. 是否執行 Fill Notch and Gap 步驟？ _____

以下(S-1 至 S-2)為使用 SE 者才須回答

S-1. power ring上是否有overlap vias？ *No*

S-2. 是否確定IO Row和Corner Row互相貼齊？ *Yes*

6. 佈局後模擬

6-1. 是否做過post-layout gate-level simulation？ *Yes*

STA(static timing analysis) 軟體 ？ *Primetime / Modelsim*

6-2. 是否做過post-layout transistor-level simulation？ *No*

6-3. 已針對以下環境狀態模擬：___ SS、___ TT、___ FF

6-4. 晶片取得時將以何種方式進行測試？ *P600 of Agilent 93000*

6-5. 模擬時是否考量輸出負載影響？ *Yes*

7. DRC/LVS 驗證

7-1. 是否有DRC錯誤？ *No* 錯誤原因：

驗證DRC軟體？ *Calibre*

是否有不作DRC的區域？ *No*

7-2. 是否有LVS錯誤？ *No*

驗證LVS 軟體？ *Calibre*

是否有非CIC提供的BlackBox？ *No*

設計者簽名: 薛智文/黃德瑋 指導教授簽名: 林進燈