

國立交通大學

電機學院 IC 設計產業研發碩士班

碩士論文

針對低功率設計的新匯流排編碼技術



A New Bus Encoding Scheme for Low Power Design

研究生：蘇彥欽

指導教授：陳紹基 博士

中華民國九十六年八月

針對低功率設計的新匯流排編碼技術

A New Bus Encoding Scheme for Low Power Design

研 究 生：蘇彥欽

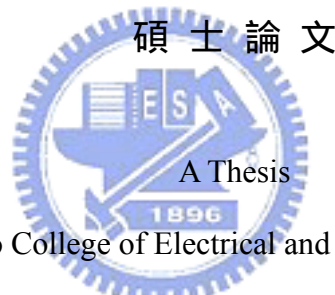
Student：Yen-Chin Su

指導教授：陳紹基 博士

Advisor：Dr. Sau-Gee Chen

國 立 交 通 大 學

電機學院 IC 設計產業研發碩士班



Submitted to College of Electrical and Computer Engineering

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Industrial Technology R & D Master Program on

IC Design

August 2007

Hsinchu, Taiwan, Republic of China

中華民國九十六年八月

針對低功率設計的新匯流排編碼技術


研究生：蘇彥欽

指導教授：陳紹基 博士

國立交通大學

電機學院 IC 設計產業研發碩士專班

摘要

The logo of National Tsing Hua University is a circular seal with a gear-like border. Inside the seal, there are the letters 'ES A' and the year '1896'.

對於低功率設計來說，降低匯流排的電位變化數目是個相當重要的議題。在本篇論文中，我們提出一種新的匯流排編碼技術來降低匯流排電位變化數目，此方法根據資料特性動態地選擇 identify、invert 或 xnor 運算來執行匯流排編碼。我們比較了 BI、BITS、hihrTS 以及 EXODUS 等編碼方式，以隨機亂數、不同形式的音訊、影像以及經過 DCT 轉換的影像等檔案來做實驗。之後我們以 VHDL 來實現我們的架構以及論文中所提及的方法，得到編碼器/解碼器所需的功耗。根據匯流排的動態功耗計算方式，我們由實驗結果推導並比較以不同方法編碼時的實際功耗。最後我們以 IEEE 802.16e 基頻傳送端為例，利用 CoWare Platform Architect 建立 ARM-Based 系統晶片虛擬平台並以本文所提及之編碼方式來模擬。


A New Bus Encoding Scheme for Low Power Design

Student : Yen-Chin Su

Advisor : Dr. Sau-Gee Chen

Industrial Technology R & D Master Program of Electrical and Computer Engineering College National Chiao Tung University

ABSTRACT

The logo of National Chiao Tung University is a circular emblem with a gear-like border. Inside the circle, there is a stylized building or structure with the letters 'ES' and 'A' on it. Below the building, the year '1896' is inscribed. The logo is positioned behind the abstract text.

Reducing the number of bus transitions is an important issue for low power design. In this thesis, we propose a new bus encoding scheme to reduce bus transitions. This scheme dynamically adapts to data patterns and applies identity, invert and xnor bus encoding operations accordingly. We compared our encoding scheme with BI, BITS, hihrTS and EXODUS encoding schemes. We perform experiments with random data streams, different types of audio input, image files and DCT-transformed image files. Then we implement our structure and all mentioned schemes in VHDL and obtain the power consumption of encoder/decoder. According to the dynamic power consumption of bus model, we calculate and compare the power consumption for all experiments. Finally we use CoWare Platform Architect to build an ARM-Based SoC virtual platform and implement all encoding schemes. The study case is IEEE 802.16e baseband TX.

誌 謝

首先感謝恩師 陳紹基教授兩年來的細心指導，在每次的報告及討論中，給予寶貴的意見與指導，使得研究能順利的完成。再來是感謝實驗室所有學長及同學以及 IC 設計產碩專班同學在課業與研究上所給予的幫助與指導，更感謝家人的支持與鼓勵，使我能完成這份論文。

祝福所有幫助過我的人，

事事順心、身體健康！

蘇彥欽 2007/08/31



Contents

Chapter 1	Introduction.....	1
1.1	Importance of Low Power for Embedded Systems.....	1
1.2	Research Motivation	1
1.3	Organization of This Thesis	2
Chapter 2	Background.....	3
2.1	Bus Model	3
2.2	Bus Model with Self and Coupling Capacitances	4
2.3	Bus Transitions	4
2.4	Power Dissipation.....	5
2.5	Bus Encoding	8
Chapter 3	Existing Encoding Schemes.....	9
3.1	BI Encoding Scheme	9
3.2	BITS Encoding Scheme	13
3.3	hihrTS Encoding Scheme	16
3.4	EXODUS Encoding Scheme.....	19
3.5	Summary and Observation	24
Chapter 4	The Proposed Bus Encoding Schemes.....	27
4.1	Design Overview.....	27

4.2	Algorithm of Dynamic BI-XNOR Encoding.....	27
4.2.1	Algorithm of Encoding	27
4.2.2	Algorithm of Decoding.....	32
4.3	Architecture of Dynamic BI-XNOR Encoding.....	33
4.3.1	The Encoding Stage.....	33
4.3.2	The Decoding Stage.....	35
Chapter 5	Simulation Results	38
5.1	Simulation Overview.....	38
5.2	Simulation Results.....	39
5.2.1	Self Transitions	39
5.2.2	Correlative Transitions	43
5.3	Experiment with Sample Patterns.....	47
5.3.1	Random Data Streams	47
5.3.2	Audio Files	48
5.3.3	Gray Image Files	53
5.3.4	Color Image Files	56
5.3.5	DCT-transformed Image Files	58
5.3.6	Comparisons	60
Chapter 6	Implementation Results.....	62
6.1	Implementation Results of Encoder/Decoder.....	62
6.2	Dynamic Power Estimation.....	63

6.2.1	Random Data Streams	64
6.2.2	Audio Files	64
6.2.3	Gray Image Files	65
6.2.4	Color Image Files	66
6.2.5	DCT-transformed Image Files	66
Chapter 7	Application on Virtual Platform.....	68
7.1	ESL Design Methodology and Virtual Platform	68
7.2	Simulation Results.....	70
Chapter 8	Conclusions	72
References	73
About the Author	75



List of Figures

Fig.2.1	General Bus Model	3
Fig.2.2	Bus Model with Self and Coupling Capacitances	4
Fig.2.3	Self Transitions.....	5
Fig.2.4	Correlative Transitions	5
Fig.2.5	(a) Charging Event of Self Capacitance (b) Discharging Event of Self Capacitance.....	6
Fig.2.6	Charging / Discharging Events of Coupling Capacitance (a) 10 to 01 (b) 01 to 10	7
Fig.2.7	Bus Encoding	8
Fig.3.1	Flow Chart of BI Encoding Scheme.....	10
Fig.3.2	An Example of BI Encoding Scheme	11
Fig.3.3	Encoder Structure of BI Scheme	12
Fig.3.4	Decoder Structure of BI Scheme	13
Fig.3.5	Flow Chart of BITS Encoding Scheme.....	14
Fig.3.6	An Example of BITS Encoding Scheme	14
Fig.3.7	Encoder Structure of BITS Scheme	15
Fig.3.8	Decoder Structure of BITS Scheme	16
Fig.3.9	Flow Chart of hihrTS Encoding Scheme.....	17
Fig.3.10	An Example of hihrTS Encoding Scheme	17

Fig.3.11	Encoder Structure of hihrTS Scheme	18
Fig.3.12	Decoder Structure of hihrTS Scheme	19
Fig.3.13	Flow Chart of EXODUS Encoding Scheme	21
Fig.3.14	An Example of EXODUS Encoding Scheme.....	22
Fig.3.15	Encoder Structure of EXODUS Scheme.....	23
Fig.3.16	Decoder Structure of EXODUS Scheme.....	24
Fig.3.17	An Example in Which Some Encoding Schemes Make No. of Transitions Become Worse	25
Fig.3.18	An Example in Which BI Encoding is Not The Best Scheme	26
Fig.4.1	Flow Chart of Dynamic BI-XNOR Encoding Scheme	29
Fig.4.2	An Example of Dynamic BI-XNOR Encoding Scheme.....	30
Fig.4.3	Encoder Structure of Dynamic BI-XNOR Encoding Scheme	34
Fig.4.4	Decoder Structure of Dynamic BI-XNOR Encoding Scheme	36
Fig.5.1	Averaged Conditional Self Transitions after Encoding.....	39
Fig.5.2	Averaged Conditional Self Transition Reduction after Encoding.....	40
Fig.5.3	Averaged Self Transitions after Encoding	42
Fig.5.4	Averaged Self Transition Reduction after Encoding	42
Fig.5.5	Averaged Conditional Correlative Transitions after Encoding.....	43
Fig.5.6	Averaged Conditional Correlative Transition Reduction after Encoding	44
Fig.5.7	Averaged Correlative Transitions after Encoding.....	46

Fig.5.8	Averaged Correlative Transition Reduction after Encoding	46
Fig.5.9	An Example of Voice Input (a)Decimal Notation (b)16-bit Two's Complement Notation (c)Transmission with an 8-bit Wide Bus	48
Fig.5.10	Waveform of a Classical Music File.....	49
Fig.5.11	Waveform of a Pop Music File	49
Fig.5.12	Waveform of a Normal Speech File	50
Fig.5.13	Waveform of a Noisy Speech File.....	51
Fig.5.14	Waveform of a Song File.....	51
Fig.5.15	Gray Image File 1	53
Fig.5.16	Gray Image File 2	54
Fig.5.17	Gray Image File 3	54
Fig.5.18	Color Image File 1	56
Fig.5.19	Color Image File 2	57
Fig.5.20	Color Image File 3	57
Fig.7.1	An ESL Design Methodology.....	69
Fig.7.2	Block Diagram of Virtual Platform	69

List of Tables

Table 2.1	Power Analysis of Self Transitions.....	6
Table 3.1	Encoding Rule of EXODUS Encoding Scheme.....	21
Table 4.1	Analysis and Comparison of XNOR / Invert + XNOR Operations.....	31
Table 5.1	No. of Averaged Conditional Self Transitions	46
Table 5.2	No. of Averaged Self Transitions for Each Encoding Scheme.....	463
Table 5.3	No. of Averaged Conditional Correlative Transitions	465
Table 5.4	No. of Averaged Correlative Transitions for Each Encoding Scheme.....	466
Table 5.5	Experimental Result of Random Data Streams.....	467
Table 5.6	Experimental Result of Classical Music	49
Table 5.7	Experimental Result of Pop Music.....	50
Table 5.8	Experimental Result of Normal Speech.....	50
Table 5.9	Experimental Result of Noisy Speech	51
Table 5.10	Experimental Result of a Song File.....	52
Table 5.11	Averaged Self Transition Reduction of Audio Files	52
Table 5.12	Averaged Correlative Transition Reduction of Audio Files.....	53
Table 5.13	Experimental Result of Gray Image File 1.....	54
Table 5.14	Experimental Result of Gray Image File 2.....	54

Table 5.15	Experimental Result of Gray Image File 3.....	55
Table 5.16	Averaged Self Transition Reduction of Gray Image Files.....	55
Table 5.17	Averaged Correlative Transition Reduction of Gray Image Files.	55
Table 5.18	Experimental Result of Color Image File 1.....	56
Table 5.19	Experimental Result of Color Image File 2.....	57
Table 5.20	Experimental Result of Color Image File 3.....	57
Table 5.21	Averaged Self Transition Reduction of Color Image Files.....	58
Table 5.22	Averaged Correlative Transition Reduction of Color Image Files	58
Table 5.23	Experimental Result of DCT-transformed Image File 1.....	59
Table 5.24	Experimental Result of DCT-transformed Image File 2.....	59
Table 5.25	Experimental Result of DCT-transformed Image File 3.....	59
Table 5.26	Averaged Self Transition Reduction of DCT-transformed Image Files.....	60
Table 5.27	Averaged Correlative Transition Reduction of DCT-transformed Image Files	60
Table 5.28	Performance Comparison of Self Transition Reduction.....	61
Table 5.29	Performance Comparison of Correlative Transition Reduction....	61
Table 6.1	Implementation Results of Encoder/Decoder for Each Scheme.....	62
Table 6.2	Self Capacitance and Coupling Capacitance in Various IC Processes	63
Table 6.3	Bus Transitions of Random Data when The Frequency is 100MHz.....	64

Table 6.4	Power Consumption of Random Data for Each Scheme	64
Table 6.5	Bus Transitions of Audio Files when The Frequency is 100MHz..	65
Table 6.6	Power Consumption of Audio Files for Each Scheme.....	65
Table 6.7	Bus Transitions of Gray Image Files when The Frequency is 100MHz	65
Table 6.8	Power Consumption of Gray Image Files for Each Scheme.....	65
Table 6.9	Bus Transitions of Color Image Files when The Frequency is 100MHz	66
Table 6.10	Power Consumption of Color Image Files for Each Scheme.....	66
Table 6.11	Bus Transitions of DCT-transformed Image Files when The Frequency is 100MHz	67
Table 6.12	Power Consumption of DCT-transformed Image Files for Each Scheme.....	67
Table 7.1	Total Memory and HW Model Accesses	70
Table 7.2	Comparisons of Self Transitions (ARM to Bus)	70
Table 7.3	Comparisons of Correlative Transitions (ARM to Bus)	71
Table 7.4	Comparisons of Self Transitions (Memory to Bus)	71
Table 7.5	Comparisons of Correlative Transitions (Memory to Bus)	71

Chapter 1

Introduction

1.1 Importance of Low Power for Embedded Systems

Low power design has become increasingly important for embedded systems, especially used in portable applications. So many portable applications are now widely used such as cell phones, personal digital assistants (PDA), portable media players (PMP) and other recreational applications like PSP, iPod, etc. All these applications need batteries and have a tendency to have long stand-alone operating time. So low power design has become an important issue for these systems. The buses consume a great portion of power in typical embedded systems. Since bus transition results in power consumption in the buses, reducing the number of bus transitions is rather important. This thesis introduces the existing bus encoding schemes which reduce the bus transitions, and then proposes a new scheme that has a better performance than before.

1.2 Research Motivation

We investigate existing bus encoding schemes and find that some schemes may result in the number of bus transitions in certain conditions. In these conditions, “Do

Nothing” might be the best bus encoding scheme. So we propose a new encoding scheme which dynamically adapts to data patterns and apply the identity, invert and xnor bus encoding operations accordingly. Thus it has a better result than the known bus encoding scheme for the reduction of bus transitions and achieves the goal of low power. Besides the simulation, we also implement all encoding schemes in RTL so as to estimate and compare the power consumption performances.

1.3 Organization of This Thesis

In the remaining part of the thesis, Chapter 2 introduces the bus model, the power consumption in the buses and the concepts of bus encoding. In Chapter 3, we introduce the existing bus encoding schemes including BI, BITS, hihrTS and EXODUS schemes. In Chapter 4, we propose a new encoding scheme – Dynamic BI-XNOR, which dynamically adapts to data patterns and applies the identity, invert and xnor bus encoding operations accordingly. In Chapter 5, we present the simulation results and related analyses. In Chapter 6, we compare all the schemes realized in RTL and estimate the power consumption. In Chapter 7, we use CoWare Platform Architect to build an ARM-Based SoC virtual platform and implement all encoding schemes. The study case is IEEE 802.16e baseband transmitter. Finally, we summarize our conclusions in Chapter 8.

Chapter 2

Background

In this section, we briefly introduce the bus model with intrinsic capacitances. The intrinsic capacitances include self capacitances and coupled capacitances. They involve in the power dissipation when the data is transmitted in the bus lines. Finally we introduce the concepts of bus encoding.

2.1 Bus Model



The general bus model is shown in Fig.2.1. The processor fetches instructions from the memory and reads data from or writes data into memory or register through bus lines.

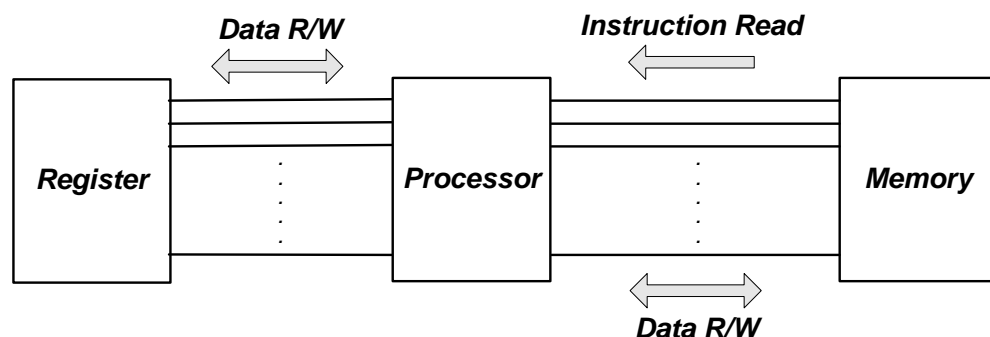


Fig.2.1 General Bus Model

2.2 *Bus Model with Self and Coupling Capacitances*

Fig.2.2 shows the bus model with self capacitances and coupling capacitances. A self capacitance (C_s) is generated between a bus line and ground. A coupling capacitance (C_c) is generated between any two adjacent bus lines.

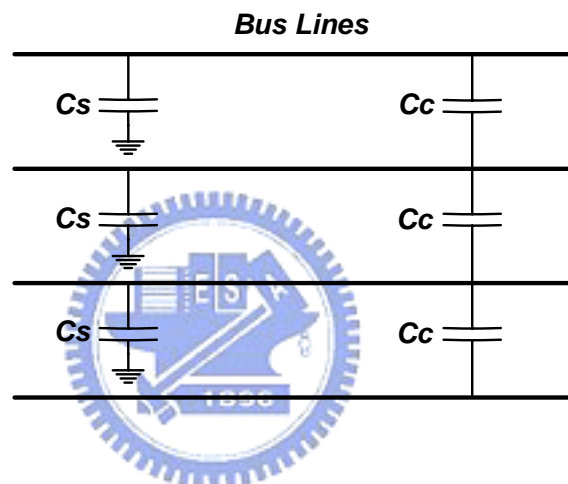


Fig.2.2 Bus Model with Self and Coupling Capacitances

2.3 *Bus Transitions*

Bus transitions (i.e., signal transitions or bit transitions) include two types – self transitions and correlative transitions. A self transition means the bit switches from 0 to 1 or 1 to 0 in the same bus line. A correlative transition occurs when two adjacent bus lines make opposite transitions at the same time. For example, one bus line switches from 0 to 1 while its adjacent bus line switches from 1 to 0. Therefore every correlative transition involves in two self transitions. Fig.2.3 shows the self transitions in a bus model. A self

transition occurs in bus line 2 (0 to 1) and another occurs in bus line 4 (1 to 0). Fig.2.4 shows correlative transitions in a bus model. A correlative transition occurs between bus line 2 and bus line 3 since the bus line 2 switches from 0 to 1 and the bus line 3 switches from 1 to 0.

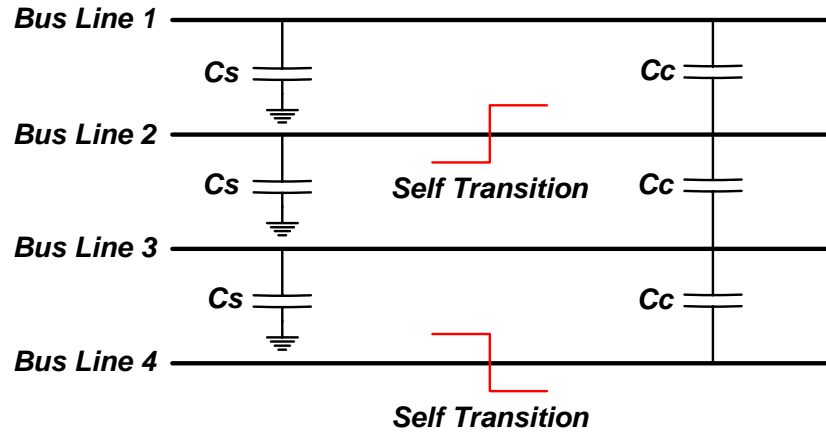


Fig.2.3 Self Transitions

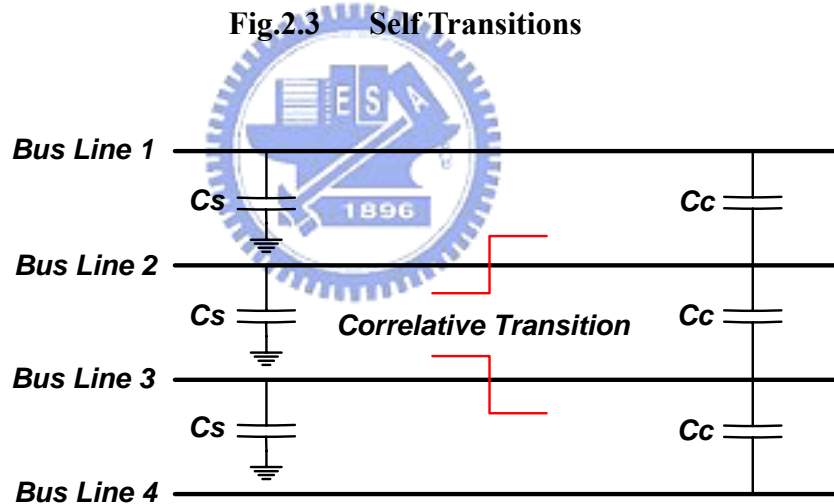


Fig.2.4 Correlative Transitions

2.4 Power Dissipation

Power dissipation in buses is related to the intrinsic capacitances of the bus lines (i.e., self and coupling capacitances) and the bus transitions occurring in the wires. Fig.2.5 [1]

shows the charging and discharging on the self capacitances when a self transition occurs.

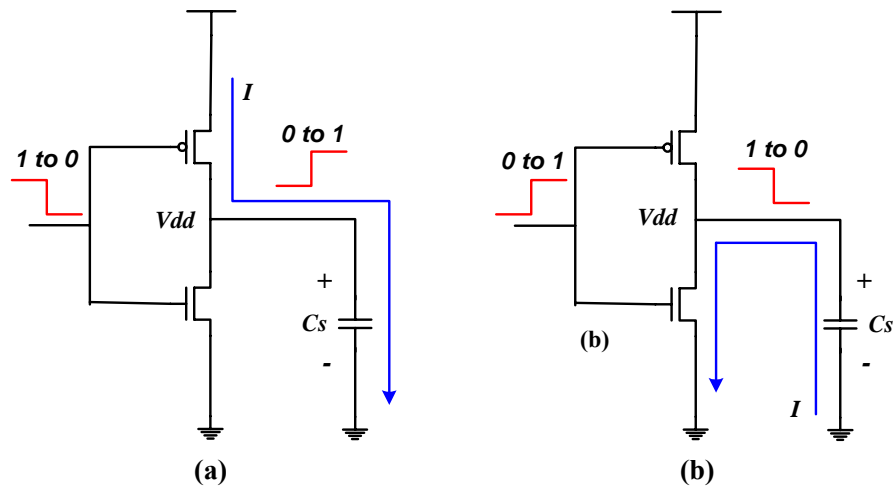


Fig.2.5 (a) Charging Event of Self Capacitance (b) Discharging Event of Self Capacitance

Table 2.1 shows the power analysis for self transitions. The power dissipation of both events is derived from the equation:

$$\begin{aligned}
 P_S &= \int_0^{\infty} I(t)V_{dd} dt \\
 &= \int_0^{\infty} C_S \frac{(dV_{dd})}{dt} V_{dd} dt \\
 &= C_S \int_0^{\infty} V_{dd} (dV_{dd}) \\
 &= \frac{1}{2} C_S V_{dd}^2
 \end{aligned} \tag{2.1}$$

where V_{dd} is the power supply on each bus wire.

Table 2.1 Power Analysis of Self Transitions

Transition Mode	Event	Initial Stored Energy	Final Stored Energy	Energy Dissipation
0 to 1	Discharge	$\frac{1}{2} C_S V_{dd}^2$	0	$\frac{1}{2} C_S V_{dd}^2$
1 to 0	Charge	0	$\frac{1}{2} C_S V_{dd}^2$	$\frac{1}{2} C_S V_{dd}^2$

Fig.2.6 shows the charging and discharging scenario on the coupling capacitances when a correlative transition occurs. [1]

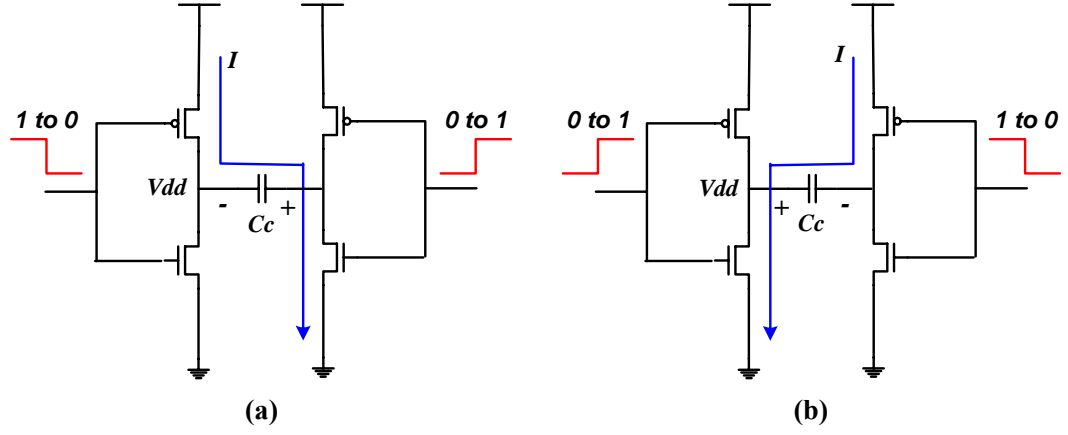


Fig.2.6 Charging / Discharging Events of Coupling Capacitance (a)10 to 01 (b)01 to 10

Since the relative change in the potential difference of the capacitance is $2V_{dd}$, and a correlative transition involves in two self transitions, the power dissipation caused by a correlative transition is expressed as

$$\begin{aligned}
 P_C &= \int_0^{\infty} I(t)(2V_{dd})^2 dt + 2 \times \frac{1}{2} C_S V_{dd}^2 \\
 &= 4 \int_0^{\infty} C_C \frac{(dV_{dd})}{dt} V_{dd}^2 dt + C_S V_{dd}^2 \\
 &= 4C_C \int_0^{\infty} V_{dd} (dV_{dd}) + C_S V_{dd}^2 \\
 &= 2C_C V_{dd}^2 + C_S V_{dd}^2
 \end{aligned} \tag{2.2}$$

So we obtain the averaged power dissipation of the bus lines per bus cycle

$$P_{bus} = \gamma_S \left(\frac{1}{2} C_S V_{dd}^2 \right) + \gamma_C (2C_C V_{dd}^2 + C_S V_{dd}^2) \tag{2.3}$$

where

γ_S is the number of averaged self transitions per bus cycle.

γ_C is the number of averaged correlative transitions per bus cycle.

Thus to lower bus transitions will result in the reduction of power dissipation.

2.5 Bus Encoding

Bus encoding means to encode the original data patterns so that the encoded patterns have fewer bus transitions than that of the original patterns. Above all, the encoded data must be able to be decoded to the original data patterns correctly. Fig.2.7 shows the model of bus encoding.

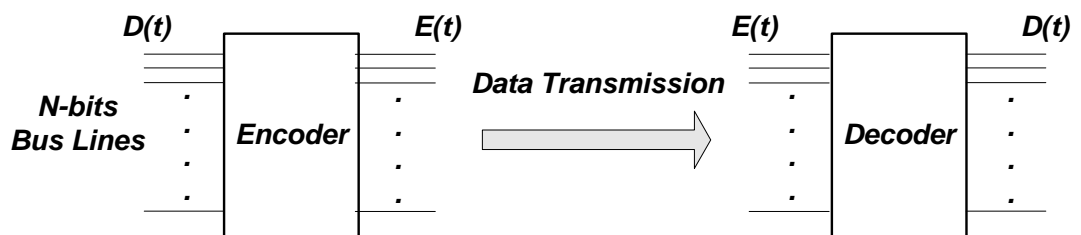


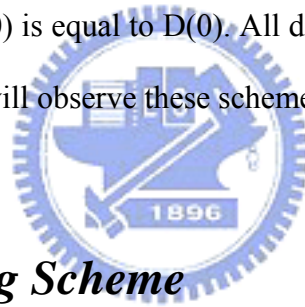
Fig.2.7 Bus Encoding

Bus encoding aims at the reduction of self transitions. Since every correlative transition involves in two self transitions, the reduction of self transitions will naturally result in the reduction of correlative transitions. In this thesis, we use an 8-bit bus as an encoding set.

Chapter 3

Existing Encoding Schemes

In this section, we introduce four known bus encoding schemes in detail. They are bus invert (BI), transition signaling combined with bus inverter (BITS), half-identity half-reverse and transition signaling (hihrTS) and exclusive or-xnor duo scheme (EXODUS). For all these encoding schemes, the first data pattern $D(0)$ is not encoded. So the first encoded pattern $E(0)$ is equal to $D(0)$. All discussions and examples are based on 8-bit wide bus. Finally we will observe these schemes and discuss their pros and cons.



3.1 BI Encoding Scheme

BI encoding scheme [2] [3] applies inverse operation to encode data patterns and an extra control bit Inv is needed to indicate the decoder how the data pattern is encoded. In the encoding process, the first pattern $D(0)$ is not encoded so $E(0)$ is equal to $D(0)$. From $D(1)$, the previous encoded pattern is compared with the current un-encoded pattern. That means, the previous encoded pattern $E(t)$ is compared with current data pattern $D(t+1)$. If the number of self transitions is less than or equal to four, the Inv bit is set to 0 and all bits of the current data pattern will not be changed, i.e.,

$$E(t+1) = D(t+1) \quad (3.1)$$

If the number of self transitions is greater than four, the Inv bit is set to 1 and all bits of

current data pattern will be inverted, i.e.,

$$E(t+1) = [D(t+1)]' \quad (3.2)$$

And the Inv bit is 1. The encoding flow chart is in shown Fig.3.1.

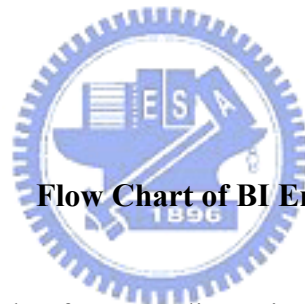


Fig.3.1 Flow Chart of BI Encoding Scheme

Fig.3.2 shows an example of BI encoding. Fig.3.2(a) shows the data patterns of 8-bit bus to be transmitted. The first data pattern $D(0)$ is not encoded, so we get $E(0)$ in Fig.3.2(b). Then we compare $E(0)$ and $D(1)$. Since there are five self transitions between $E(0)$ and $D(1)$, so we invert $D(1)$ to get $E(1)$ in Fig.3.2(c) and its Inv bit is 1. Then we compare $E(1)$ and $D(2)$. Since there are three self transitions between $E(1)$ and $D(2)$, we obtain $E(2)$ which is equal to $D(2)$ in Fig.3.2(d) and its Inv bit is 0.

<i>D</i> (0)	<i>D</i> (1)	<i>D</i> (2)		<i>E</i> (0)	<i>D</i> (1)	<i>D</i> (2)		<i>E</i> (0)	<i>E</i> (1)	<i>D</i> (2)		<i>E</i> (0)	<i>E</i> (1)	<i>E</i> (2)
1	0	1		1	0	1		1	1	1		1	1	1
0	1	0		0	1	0		0	0	0		0	0	0
1	0	1		1	0	1		1	1	1		1	1	1
1	1	1	...	1	1	1	...	1	0	1	...	1	0	1
0	1	0		0	1	0		0	0	0		0	0	0
1	0	1		1	0	1		1	1	1		1	1	1
1	1	1		1	1	1		1	0	1		1	0	1
0	0	0		0	0	0		0	1	0		0	1	0
<i>Inv</i>				<i>Inv</i>	-			<i>Inv</i>	-	1		<i>Inv</i>	-	1
	(a)			(b)				(c)				(d)		

Fig.3.2 An Example of BI Encoding Scheme

Fig.3.3 shows the encoder structure of BI scheme. The first set of XOR gates compare $E(t)$ and $D(t+1)$ to detect if they have self transitions. If they are different the output is 1, otherwise the output is 0. The comparison results are sent to multiplexer to gather statistics and send a control bit *Inv*. If the number of self transitions is greater than four, the multiplexer will send an output 1, otherwise the output is 0. In the second set of XOR gates, the *Inv* bit indicates whether $D(t+1)$ should be inverted or not. If the *Inv* bit is 1, $D(t+1)$ will be inverted to get the encoded result $E(t+1)$. Otherwise $E(t+1)$ is just equal to $D(t+1)$.

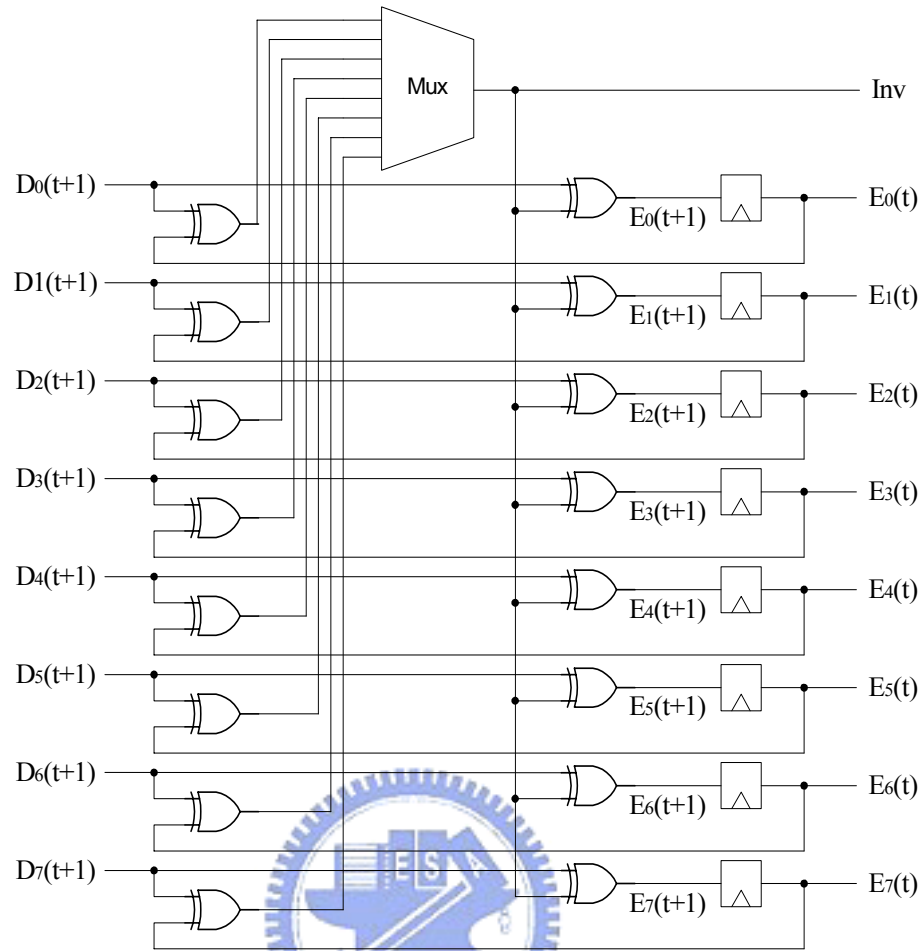


Fig.3.3 Encoder Structure of BI Scheme

Fig.3.4 shows the decoder structure of BI scheme. It's quite easy to decode. In the set of XOR gates, the Inv bit indicates whether $E(t+1)$ should be inverted or not. If Inv is 1, $E(t+1)$ is inverted to get $D(t+1)$. Otherwise $D(t+1)$ is equal to $E(t+1)$.

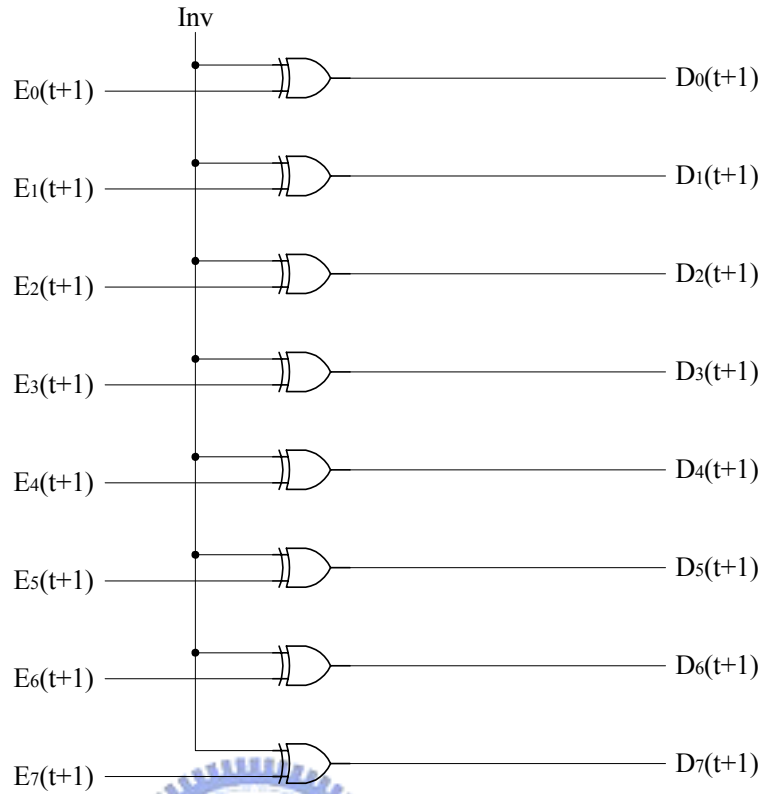


Fig.3.4 Decoder Structure of BI Scheme

3.2 BITS Encoding Scheme

BITS encoding scheme [3] applies both XOR and inverse operations to encode data patterns. The same as BI scheme, this scheme needs an extra control bit Inv, too. $E(0)$ is also equal to $D(0)$. Then this scheme calculates the number of 1's of each data pattern except $D(0)$. If the number of 1's is less than or equal to four,

$$E(t+1) = D(t+1) \oplus E(t) \tag{3.3}$$

And the Inv bit is 0. Oppositely if the number of 1's is greater than four,

$$E(t+1) = [D(t+1)]' \oplus E(t) \tag{3.4}$$

And the Inv bit is 1. The encoding flow chart is shown in Fig.3.5.

Fig.3.5 Flow Chart of BITS Encoding Scheme

Fig.3.6 shows an example of BITS encoding. Fig.3.6(a) shows the data patterns of 8-bit bus to be transmitted. The first data pattern D(0) is not encoded, so we get E(0) in Fig.3.6(b). Then we calculate the number of 1's in D(1). Since there are four 1's in D(1), so $E(1) = D(1) \oplus E(0)$, as shown in Fig.3.6(c) and the Inv bit is 0. Then we calculate the number of 1's in D(2). Since there are five 1's in D(2), $E(2) = [D(2)]' \oplus E(1)$ as shown in Fig.3.6(d) and the Inv bit is 1.

<i>D(0)</i>	<i>D(1)</i>	<i>D(2)</i>	<i>E(0)</i>	<i>D(1)</i>	<i>D(2)</i>	<i>E(0)</i>	<i>E(1)</i>	<i>D(2)</i>	<i>E(0)</i>	<i>E(1)</i>	<i>E(2)</i>
1	0	1	1	0	1	1	1	1	1	1	1
0	1	0	0	1	0	0	1	0	0	1	0
1	0	1	1	0	1	1	1	1	1	1	1
1	1	1	1	1	1	1	0	1	1	0	0
0	1	0	0	1	0	0	1	0	0	1	0
1	0	1	1	0	1	1	1	1	1	1	1
1	1	1	1	1	1	1	0	1	1	0	0
0	0	0	0	0	0	0	0	0	0	0	1
<i>Inv</i>			<i>Inv</i>	-		<i>Inv</i>	-	0	<i>Inv</i>	-	0
	(a)			(b)			(c)			(d)	

Fig.3.6 An Example of BITS Encoding Scheme

Fig.3.7 shows the encoder structure of BITS. The multiplexer calculates the

summation of $D_0(t+1)$ to $D_7(t+1)$ and sends the control bit Inv. In the first set of XOR gates, the Inv indicates whether $D(t+1)$ should be inverted or not. In the second set of XOR gates, $D(t+1)$ or $[D(t+1)]'$ executes XOR operation with $E(t)$ to get $E(t+1)$.

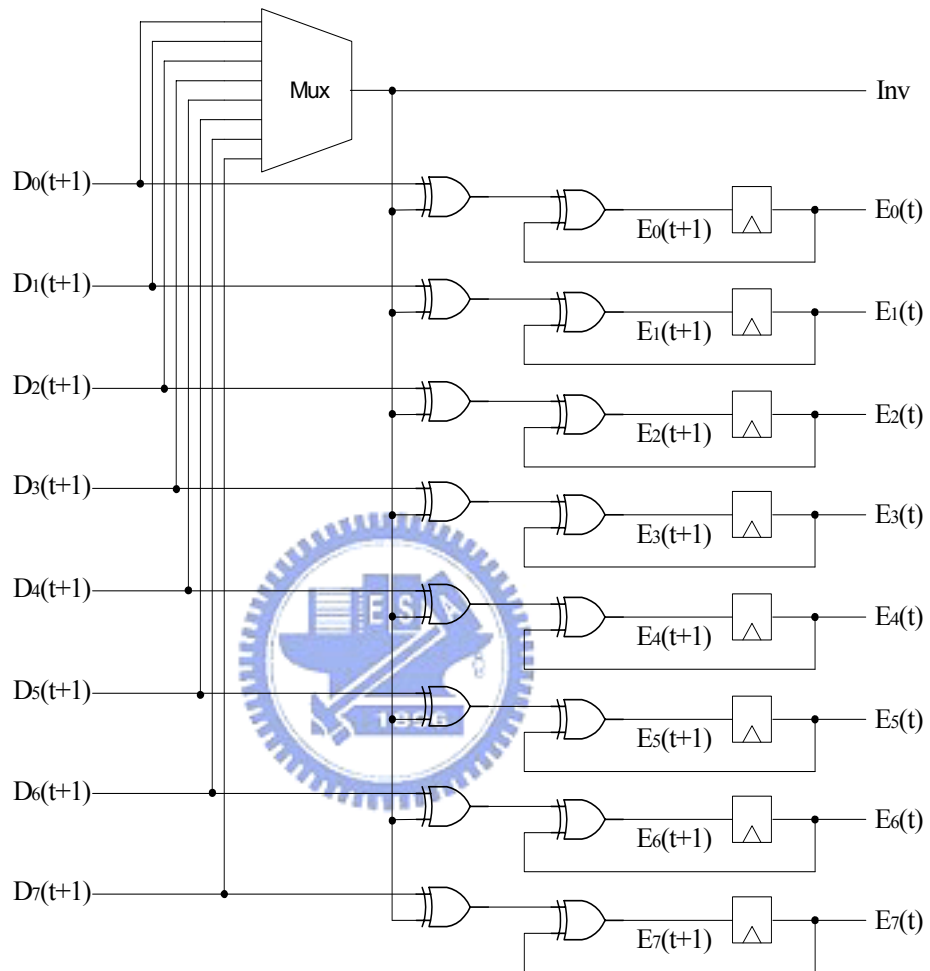


Fig.3.7 Encoder Structure of BITS Scheme

Fig.3.8 shows the decoder structure of BITS scheme. In the first set of XOR gates, $E(t+1)$ executes XOR operation with $E(t)$. In the second set of XOR gates, the Inv bit indicates whether $D(t+1)$ should be inverted or not.

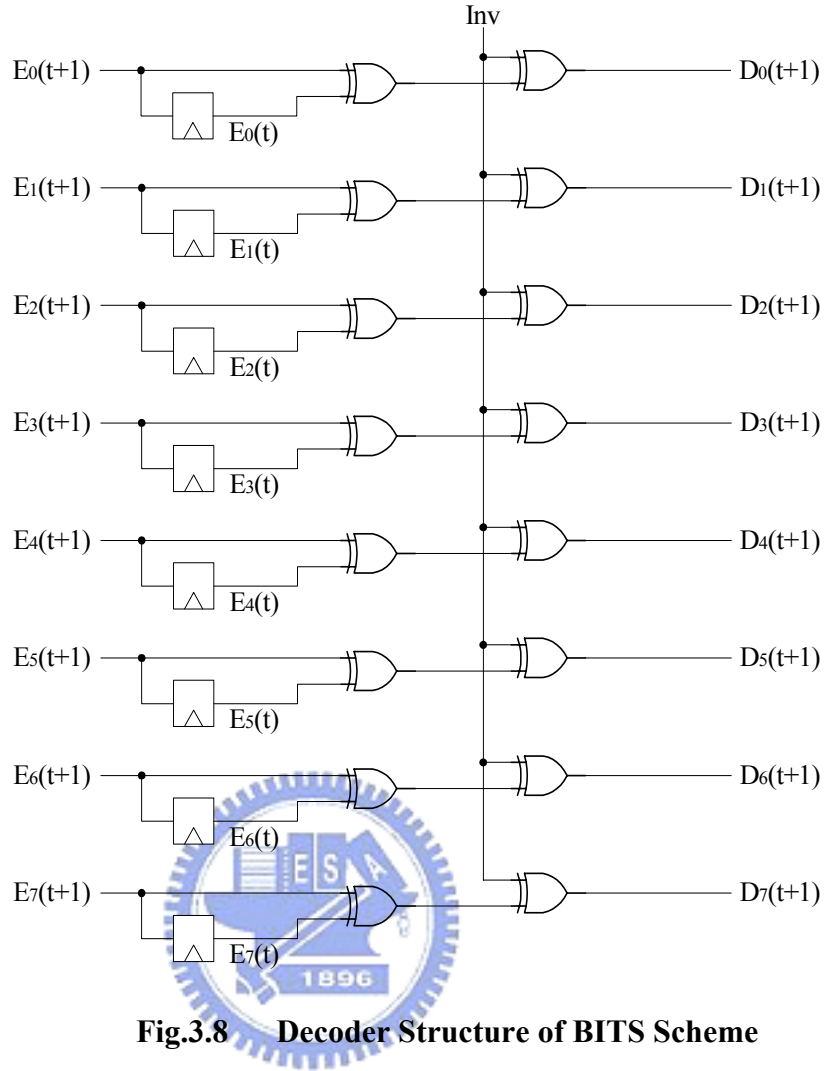


Fig.3.8 Decoder Structure of BITS Scheme

3.3 *hihrTS* Encoding Scheme

HihrTS encoding scheme [3] applies both XOR and inverse operations to encode data patterns. Unlike BI and BITS schemes, this scheme does not need any control bit. This scheme uses the first bit of each data pattern as encoder and decoder hint. So the first bit of each data pattern will not be encoded. For other seven bits, if the first bit is 0,

$$E(t+1) = D(t+1) \quad E(t) \tag{3.5}$$

if the first bit is 1,

$$E(t+1) = [D(t+1)]' \quad E(t) \tag{3.6}$$

The encoding flow chart is in shown Fig.3.9.

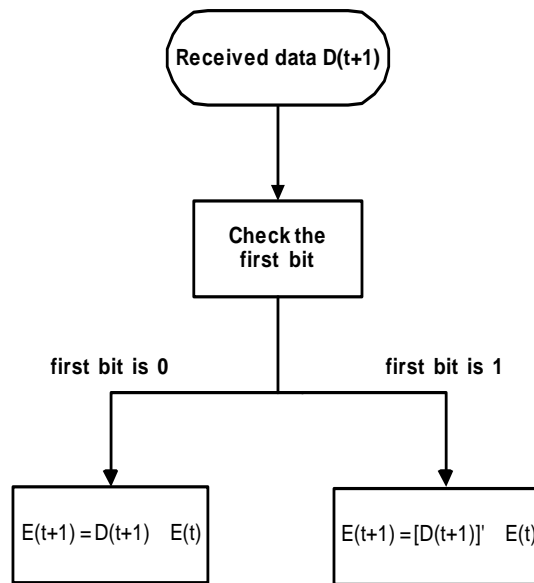


Fig.3.9 Flow Chart of hihrTS Encoding Scheme

Fig.3.10 shows an example of hihrTS encoding scheme. Fig.3.10(a) shows the data patterns of 8-bit bus to be transmitted. The first data pattern $D(0)$ is not encoded, and $E(0)=D(0)$ as shown in Fig.3.10(b). Next, we consider data pattern $D(1)$. Since its first bit is 0, we get $E(1) = D(1) E(0)$ except the first bit as depicted in Fig.3.10(c). Then we consider the data pattern $D(2)$. Since its first bit is 1, $E(2) = [D(2)]' E(1)$ except the first bit as shown in Fig.3.10(d).

$D(0)$	$D(1)$	$D(2)$	$E(0)$	$D(1)$	$D(2)$	$E(0)$	$E(1)$	$D(2)$	$E(0)$	$E(1)$	$E(2)$
1	0	1	1	0	1	1	0	1	1	0	1
0	1	0	0	1	0	0	1	0	0	1	0
1	0	1	1	0	1	1	1	1	1	1	1
1	1	1	1	1	1	1	0	1	1	0	0
0	1	0	0	1	0	0	1	0	0	1	0
1	0	1	1	0	1	1	1	1	1	1	1
1	1	1	1	1	1	1	0	1	1	0	0
0	0	0	0	0	0	0	0	0	0	0	1
(a)			(b)			(c)			(d)		

Fig.3.10 An Example of hihrTS Encoding Scheme

Fig.3.11 shows the encoder structure of hihrTS encoding scheme. In the first set of XOR gates, $D_0(t+1)$ executes XOR operation with the other seven bits ($D_1(t+1)$ to $D_7(t+1)$) respectively. If $D_0(t+1)$ is 1, all the other seven bits will be inverted. In the second set of XOR gates, $D(t+1)$ or $[D(t+1)']$ executes XOR operation with $E(t)$ to get the $E(t+1)$.

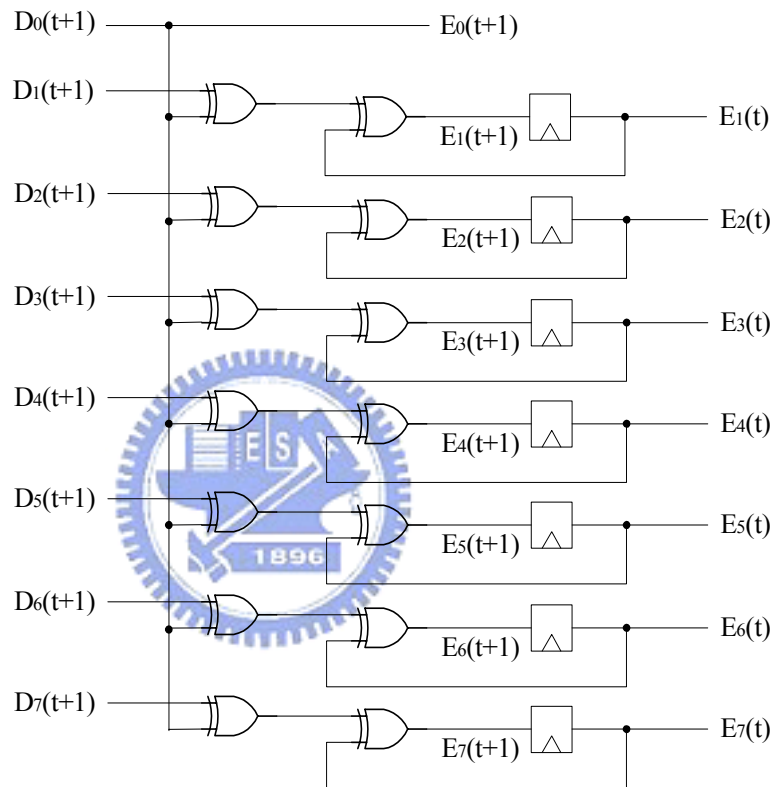


Fig.3.11 Encoder Structure of hihrTS Scheme

Fig.3.12 shows the decoder structure of hihrTS scheme. $D_0(t+1)$ is equal to $E_0(t+1)$. In the first set of XOR gates, $E(t+1)$ executes XOR operation with $E(t)$. In the second set of XOR gates, $E_0(t+1)$ indicates whether $D(t+1)$ should be inverted or not.

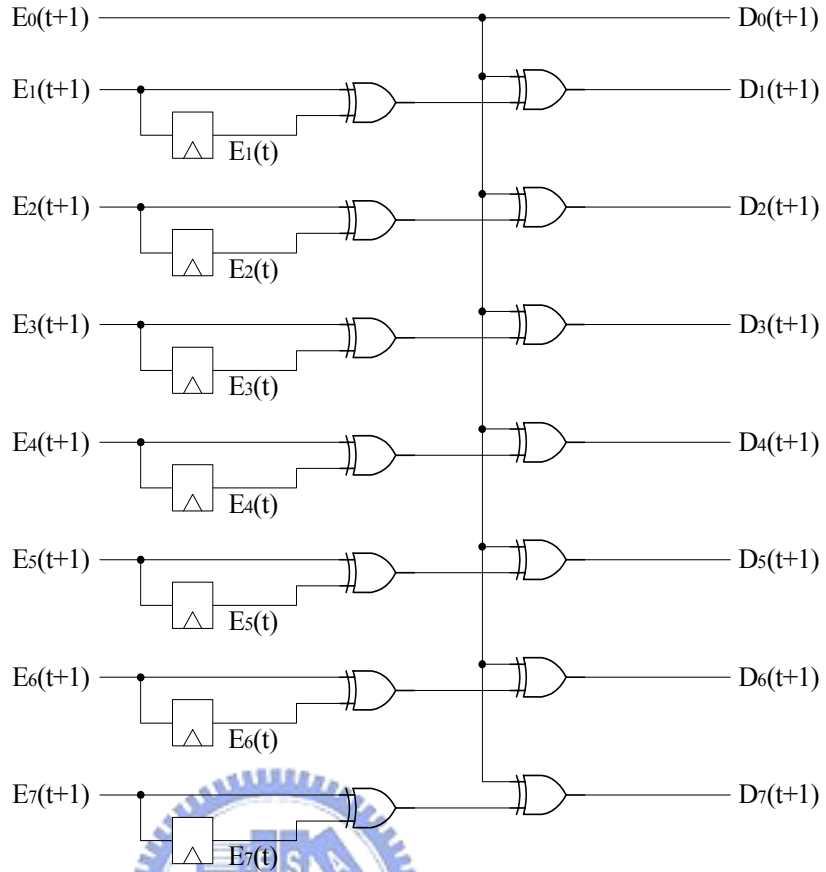


Fig.3.12 Decoder Structure of hihrTS Scheme

3.4 EXODUS Encoding Scheme

BXODUS encoding scheme [4] applies XOR-XNOR duo (XON type) or XNOR-XOR duo (XNO type) logic to encode data patterns. This scheme needs two extra control bits to indicate the decoder which encoding type the data pattern is encoded. For 8-bit wide bus data patterns, each pattern is divided into two subsets $D^{M^4}(t+1)$ and $D^{L^4}(t+1)$. Each subset consists of four bits and an additional control bit. For each subset, the two most significant bits $D^{M^2}(t+1)$ and the two least significant bits $D^{L^2}(t+1)$ are independently encoded as follows :

XON type (XOR-XNOR duo) :

$$E^{M^2}(t+1) = D^{M^2}(t+1) \oplus E^{M^2}(t)$$

$$E^{L^2}(t+1) = D^{L^2}(t+1) \oplus E^{L^2}(t) \quad (3.7)$$

XNO type (XNOR-XOR duo) :

$$E^{M^2}(t+1) = D^{M^2}(t+1) \oplus E^{M^2}(t)$$

$$E^{L^2}(t+1) = D^{L^2}(t+1) \oplus E^{L^2}(t) \quad (3.8)$$

Assume $D(t+1)$ is 0010 and $E(t)$ is 1101. The two MSBs of $D(t+1)$ are expressed as $D^{M^2}(t+1)$ which is equal to 00 and the two LSBs of $D(t+1)$ are expressed as $D^{L^2}(t+1)$ which is equal to 10. $E^{M^2}(t)$ is 11 and $E^{L^2}(t)$ is 01. If the encoding type is XON, $E^{M^2}(t+1)$ is obtained from $D^{M^2}(t+1) \oplus E^{M^2}(t)$.

$D^{M^2}(t+1)$	$E^{M^2}(t)$	$=$	$E^{M^2}(t+1)$
0	1		1
0	1		1

And $E^{L^2}(t+1)$ is obtained from $D^{L^2}(t+1) \oplus E^{L^2}(t)$.

$D^{L^2}(t+1)$	$E^{L^2}(t)$	$=$	$E^{L^2}(t+1)$
1	0		0
0	1		0

So $E(t+1)$ is 1100.

Table 3.1 shows the encoding rule of EXODUS encoding scheme. Note that it has encoding priority. For example, if a subset of four bits is 0101, it will be encoded by XNO type but not XON type because x10x group is prior to xxx1 group. The encoding flow chart is shown in Fig.3.13.

Table 3.1 Encoding Rule of EXODUS Encoding Scheme

Encoding Priority	Group of $D(t+1)$	Encoding Type
1	$x01x$	XON
	$x10x$	XNO
2	$xxx0$	XNO
	$xxx1$	XON
<i>x represents don't-care bit</i>		



Fig.3.13 Flow Chart of EXODUS Encoding Scheme

Fig.3.14 shows an example of EXODUS encoding scheme. Fig.3.14(a) shows the data patterns of the 8-bit bus to be transmitted. The first encoded data $E(0)$ is directly set to the original data pattern $D(0)$ as shown in Fig.3.14(b). Then we divide $D(1)$ into two subsets of four bits, which are 0101 and 1010, respectively. Since 0101 is in the group of “ $x10x$ ” and 1010 is in the group of “ $x01x$ ”, we apply XNO type to 0101 and XON type to 1010. Then we get $E(1)$ in Fig.3.14(c). The control bit of subset one is XNO and that of subset two is XON. Next, we divide $D(2)$ into two subsets of four bits, which are 1011 and 0110, respectively. Since 1011 is in the group of “ $x01x$ ” and 0110 is in the group of

“xxx0”, we apply XON type to 1011 and XNO type to 0110. Then we get E(2) in Fig.3.14(d). The control bit of subset one is XON and that of subset two is XNO.

<i>D(0)</i>	<i>D(1)</i>	<i>D(2)</i>		<i>E(0)</i>	<i>D(1)</i>	<i>D(2)</i>		<i>E(0)</i>	<i>E(1)</i>	<i>D(2)</i>		<i>E(0)</i>	<i>E(1)</i>	<i>E(2)</i>
1	0	1		1	0	1		1	0	1		1	0	1
0	1	0		0	1	0		0	0	0		0	0	0
1	0	1		1	0	1		1	1	1		1	1	1
1	1	1	...	1	1	1	...	1	0	1	...	1	0	0
0	1	0		0	1	0		0	1	0		0	1	0
1	0	1		1	0	1		1	1	1		1	1	1
1	1	1		1	1	1		1	1	1		1	1	0
0	0	0		0	0	0		0	1	0		0	1	1
Ctrl				Ctrl	-			Ctrl	-	XNO		Ctrl	-	XNO XON
					-				-	XON			-	XON XNO
	(a)				(b)				(c)				(d)	

Fig.3.14 An Example of EXODUS Encoding Scheme

Fig.3.15 shows the encoder structure of EXODUS encoding scheme. Multiplexer 1 decides the group of the four MSBs of $D(t+1)$ and send a control bit C_M . Multiplexer 2 decides the group of the four LSBs of $D(t+1)$ and send another control bit C_L . For the four MSBs of $D(t+1)$, $D^{M4}(t+1)$ executes XOR operation with $E^{M4}(t)$ by the first set of XOR gates. In the second set of XOR gates, if C_M is 0 $E^{M2}(t+1)$ is equal to $D^{M2}(t+1) \oplus E^{M2}(t)$, and $E^{L2}(t+1)$ is equal to $D^{L2}(t+1) \oplus E^{L2}(t)$ because $[D^{L2}(t+1) \oplus E^{L2}(t)]'$ is equal to $D^{L2}(t+1) \oplus E^{L2}(t)$. The inversion gates denote that the two LSBs and the two MSBs of the four MSBs have opposite operations. If C_M is 1, all operations are reverse. For the four LSBs of $D(t+1)$, C_L has the same function as C_M in all operations.

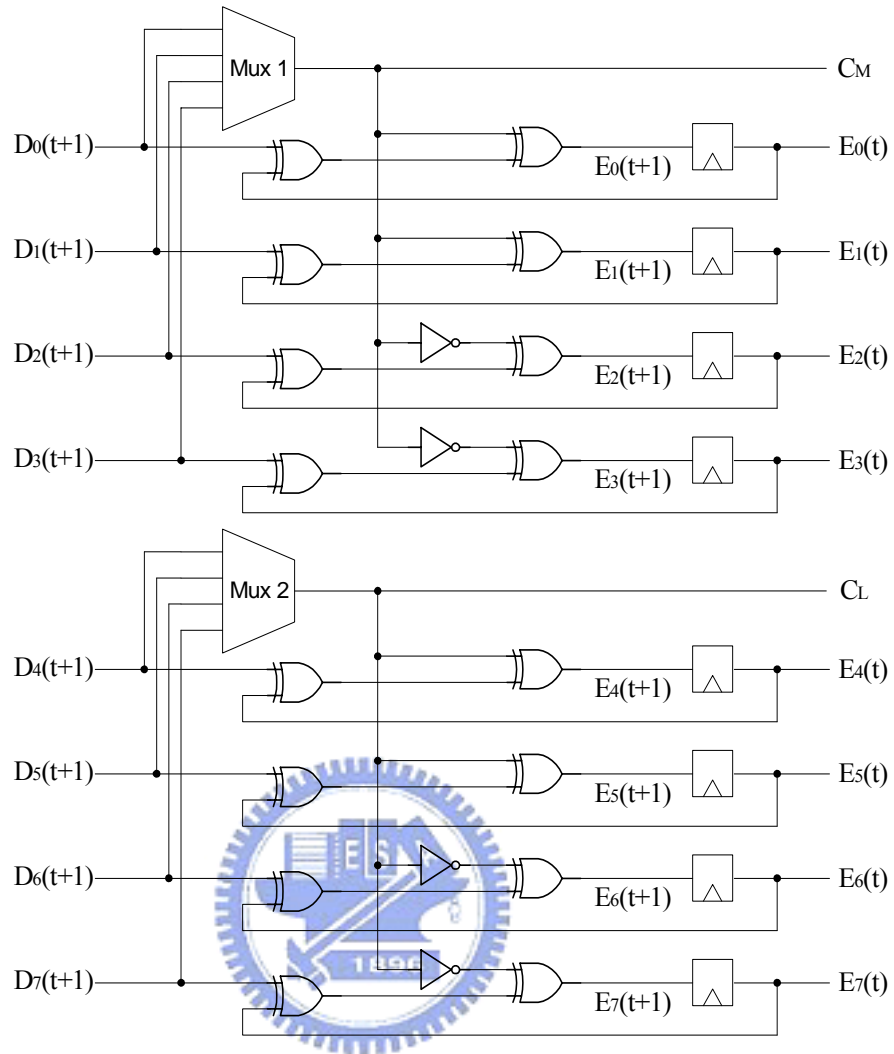


Fig.3.15 Encoder Structure of EXODUS Scheme

Fig.3.16 shows the decoder structure of EXODUS scheme. In the first set of XOR gates, $E(t+1)$ executes XOR operation with $E(t)$. In the second set of XOR gates, C_M indicates the four MSBs whether $D(t+1)$ should be inverted or not and C_L indicates the four LSBs whether $D(t+1)$ should be inverted or not. The inversion gates denote that the two MSBs and the two LSBs of each 4-bit subset have opposite operations.

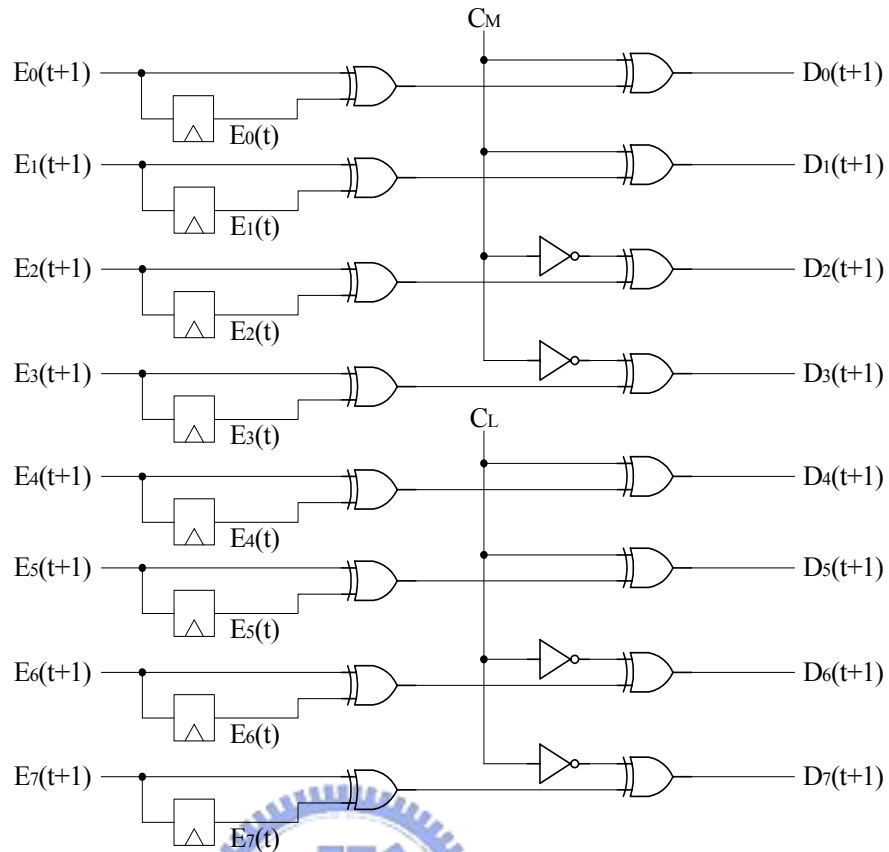


Fig.3.16 Decoder Structure of EXODUS Scheme

3.5 Summary and Observation

For all the bus encoding schemes mentioned above, $E(t+1)$ is obtained from $[D(t+1) \text{ op } E(t)]$ except the first pattern $E(0)$. The encoder and decoder function must satisfy the following function [5]:

$$[D(t+1) \text{ op } E(t)] \text{ op } E(t) = D(t+1) \quad (3.9)$$

Since $[D(t+1) \text{ op } E(t)]$ is equal to $E(t+1)$, the original data pattern $D(t+1)$ can be easily decoded by the equation $[E(t+1) \text{ op } E(t)]$ except $D(0)$. $D(0)$ is just equal to $E(0)$ because it is not encoded. Only four elementary Boolean functions satisfy equation (3.9). They are identify, invert, XOR and XNOR. Due to this property we know why all those encoding schemes never use other operations the four functions to perform encoding.

All the encoding schemes mentioned above aim at reducing transitions in the bus lines. But for some encoding schemes, the data patterns may generate more transitions after encoding than before. For example, assume the original data patterns are from 01010101 to 01010101, in this case there is no transition. In BITS and hihrTS encoding, the encoded patterns are from 01010101 to 00000000, which have four self transitions. In EXODUS encoding, the encoded patterns are from 01010101 to 11001100, which have four self transitions, too. We could easily understand, in this case “Do Nothing” is the best encoding scheme. So in this case BI encoding is better than the other schemes, because the encoded patterns are from 01010101 to 01010101 with no transition after BI encoding. Fig.3.17 shows this example.

<i>origin</i>		<i>BI</i>		<i>BITS</i>		<i>hihrTS</i>		<i>EXODUS</i>	
<i>E(t)</i>	<i>D(t+1)</i>	<i>E(t)</i>	<i>E(t+1)</i>	<i>E(t)</i>	<i>E(t+1)</i>	<i>E(t)</i>	<i>E(t+1)</i>	<i>E(t)</i>	<i>E(t+1)</i>
0	0	0	0	0	0	0	0	0	1
1	1	1	1	1	0	1	0	1	1
0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	1	0	1	0
0	0	0	0	0	0	0	0	0	1
1	1	1	1	1	0	1	0	1	1
0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	1	0	1	0

no transition *no transition* *4 transitions* *4 transitions* *4 transitions*

Fig.3.17 An Example in Which Some Encoding Schemes Make No. of Transitions Become Worse

However, BI is not always the best way. For example, assume the original data patterns are from 01010111 to 00000000, in this case there are five self transitions. In BI encoding, the encoded patterns are from 01010111 to 11111111, which have three self transitions. But in BITS and hihrTS encodings, the encoded patterns are from 01010111 to 01010111, which have no transition. Fig.3.18 shows this example.

<i>origin</i>		<i>BI</i>		<i>BITS</i>		<i>hihrTS</i>		<i>EXODUS</i>	
<i>E(t)</i>	<i>D(t+1)</i>	<i>E(t)</i>	<i>E(t+1)</i>	<i>E(t)</i>	<i>E(t+1)</i>	<i>E(t)</i>	<i>E(t+1)</i>	<i>E(t)</i>	<i>E(t+1)</i>
0	0	0	1	0	0	0	0	0	1
1	0	1	1	1	1	1	1	1	0
0	0	0	1	0	0	0	0	0	0
1	0	1	1	1	1	1	1	1	1
0	0	0	1	0	0	0	0	0	1
1	0	1	1	1	1	1	1	1	0
1	0	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1
5 transitions		3 transitions		no transition		no transition		3 transitions	

Fig.3.18 An Example in Which BI Encoding is Not The Best Scheme

From above, we know that each scheme has its superiority in specific conditions. If we can analyze this characteristic and gather statistics, we may find a better encoding scheme that has the most improvement on bus transitions.



Chapter 4

The Proposed Bus Encoding Schemes

In this chapter, our encoding schemes are described. Since this thesis focuses on 8-bit wide bus, so is the following discussion. The architectures with the multiple of 8-bit are also suitable for the proposed schemes. For example, if a system uses 16-bit bus to transmit data, we can divide the 16-bit bus into two 8-bit buses for hardware saving. If speed issue is of primary concern, we can use two sets of the proposed encoding architecture.



4.1 Design Overview

We aim to propose dynamic BI-XNOR encoding, which includes three encoding operations - identity, invert and XNOR. As mentioned in the previous chapter, in some conditions “Do Nothing” is better than other complicated encoding schemes. The object of this design is to choose the most suitable encoding operation dynamically for different conditions of data patterns.

4.2 Algorithm of Dynamic BI-XNOR Encoding

4.2.1 Algorithm of Encoding

Let $D(t)$ denote an 8-bits data pattern on a bus at time t , and $E(t)$ is the encoded pattern of $D(t)$. Computing the self transitions between $E(t)$ and its next data pattern $D(t+1)$, if the number of self transitions is less than or equal to 2, the encoding function is:

$$E(t+1) = D(t+1) \quad (4.1)$$

If the number of self transitions is greater than or equal to 6, the encoding function is:

$$E(t+1) = [D(t+1)]' \quad (4.2)$$

If the number of self transitions is equal to 3, 4 or 5, divide the 8-bits bus into two subsets $D_s(t+1)$ with each subset consisting of 4-bits. The four MSBs are $D_s^M(t+1)$ and the four LSBs are $D_s^L(t+1)$. Compute the number of 1's in each subset $D_s(t+1)$. If the number of 1's is greater than or equal to the number of 0's (i.e., the number of 1's is greater than or equal to 2), the encoding function is:

$$E_s(t+1) = D_s(t+1) \quad E_s(t) \quad (4.3)$$

If the number of 1's is less than the number of 0's (i.e., the number of 1's is less than 2), the encoding function is:

$$E_s(t+1) = [D_s(t+1)]' \quad E_s(t) \quad (4.4)$$

The encoding flow chart is shown in Fig.4.1.



Fig.4.1 Flow Chart of Dynamic BI-XNOR Encoding Scheme

Assume a series of data patterns $D(0), D(1), D(2), \dots, D(t), D(t+1), \dots, D(\text{end})$ are waiting for transmission. The first pattern $E(0)$ is equal to $D(0)$. Then $E(1)$ is obtained from $E(0)$ and $D(1)$ after encoding similarly for all the other $E(t)$. Then we obtain a series of encoded patterns $E(0), E(1), E(2), \dots, E(t), E(t+1), \dots, E(\text{end})$.

Fig.4.2 shows an example of dynamic BI-XNOR encoding. Fig.4.2(a) shows data patterns of 8-bit bus to be transmitted. $E(0) = D(0)$ is shown in Fig.4.2(b). Then we compare $E(0)$ and $D(1)$. Since there is only one self transition between them, from

equation (4.1) we get $E(1)$ which is equal to $D(1)$ in Fig.4.2(c). Then we compare $E(1)$ and $D(2)$. Since there seven self transitions between them, from equation (4.2) we invert $D(2)$ to get $E(2)$ in Fig.4.2(d). Next, we compare $E(2)$ and $D(3)$. There are four self transitions between them, so we divide $D(3)$ into two subsets $D_S^M(3)$ and $D_S^L(3)$. Consider $D_S^M(3)$. It has three 1's, from equation (4.3) we get $E_S^M(3)$ which is equal to $D_S^M(3) \oplus E_S^M(2)$ in Fig.4.2(e). Finally we consider $D_S^L(3)$. It has one 1's, from equation (4.4) we get $E_S^L(3)$ which is equal to $[D_S^L(3)]' \oplus E_S^L(2)$ in Fig.4.2(e).

<table style="margin: auto;"> <thead> <tr><th>$D(0)$</th><th>$D(1)$</th><th>$D(2)$</th><th>$D(3)$</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>0</td></tr> </tbody> </table> <p>(a)</p>	$D(0)$	$D(1)$	$D(2)$	$D(3)$	0	0	1	1	0	0	1	0	1	0	1	1	1	1	1	1	0	0	1	0	1	1	0	0	1	1	0	1	0	0	1	0	<table style="margin: auto;"> <thead> <tr><th>$E(0)$</th><th>$D(1)$</th><th>$D(2)$</th><th>$D(3)$</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>0</td></tr> </tbody> </table> <p>(b)</p>	$E(0)$	$D(1)$	$D(2)$	$D(3)$	0	0	1	1	0	0	1	0	1	0	1	1	1	1	1	1	0	0	1	0	1	1	0	0	1	1	0	1	0	0	1	0	<table style="margin: auto;"> <thead> <tr><th>$E(0)$</th><th>$E(1)$</th><th>$D(2)$</th><th>$D(3)$</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>0</td></tr> </tbody> </table> <p>(c)</p>	$E(0)$	$E(1)$	$D(2)$	$D(3)$	0	0	1	1	0	0	1	0	1	0	1	1	1	1	1	1	0	0	1	0	1	1	0	0	1	1	0	1	0	0	1	0
$D(0)$	$D(1)$	$D(2)$	$D(3)$																																																																																																											
0	0	1	1																																																																																																											
0	0	1	0																																																																																																											
1	0	1	1																																																																																																											
1	1	1	1																																																																																																											
0	0	1	0																																																																																																											
1	1	0	0																																																																																																											
1	1	0	1																																																																																																											
0	0	1	0																																																																																																											
$E(0)$	$D(1)$	$D(2)$	$D(3)$																																																																																																											
0	0	1	1																																																																																																											
0	0	1	0																																																																																																											
1	0	1	1																																																																																																											
1	1	1	1																																																																																																											
0	0	1	0																																																																																																											
1	1	0	0																																																																																																											
1	1	0	1																																																																																																											
0	0	1	0																																																																																																											
$E(0)$	$E(1)$	$D(2)$	$D(3)$																																																																																																											
0	0	1	1																																																																																																											
0	0	1	0																																																																																																											
1	0	1	1																																																																																																											
1	1	1	1																																																																																																											
0	0	1	0																																																																																																											
1	1	0	0																																																																																																											
1	1	0	1																																																																																																											
0	0	1	0																																																																																																											
<table style="margin: auto;"> <thead> <tr><th>$E(0)$</th><th>$E(1)$</th><th>$E(2)$</th><th>$D(3)$</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td></tr> </tbody> </table> <p>(d)</p>	$E(0)$	$E(1)$	$E(2)$	$D(3)$	0	0	0	1	0	0	0	0	1	0	0	1	1	1	0	1	0	0	0	0	1	1	1	0	1	1	1	1	0	0	0	0	<table style="margin: auto;"> <thead> <tr><th>$E(0)$</th><th>$E(1)$</th><th>$E(2)$</th><th>$E(3)$</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td></tr> </tbody> </table> <p>(e)</p>	$E(0)$	$E(1)$	$E(2)$	$E(3)$	0	0	0	0	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0																																					
$E(0)$	$E(1)$	$E(2)$	$D(3)$																																																																																																											
0	0	0	1																																																																																																											
0	0	0	0																																																																																																											
1	0	0	1																																																																																																											
1	1	0	1																																																																																																											
0	0	0	0																																																																																																											
1	1	1	0																																																																																																											
1	1	1	1																																																																																																											
0	0	0	0																																																																																																											
$E(0)$	$E(1)$	$E(2)$	$E(3)$																																																																																																											
0	0	0	0																																																																																																											
0	0	0	1																																																																																																											
1	0	0	0																																																																																																											
1	1	0	0																																																																																																											
0	0	0	0																																																																																																											
1	1	1	1																																																																																																											
1	1	1	0																																																																																																											
0	0	0	0																																																																																																											

Fig.4.2 An Example of Dynamic BI-XNOR Encoding Scheme

Why we adopt XNOR operation on the subsets? Let's analyze the relationship between the self transition and the data type first. Assume a data stream is from $0(E(t))$ to $1(D(t+1))$ which has a self transition. If we adopt XNOR operation on $E(t)$ and $D(t+1)$, $E(t+1)$ will become 0. Thus the data stream will go from $0(E(t))$ to $0(E(t+1))$, such that the self transition is vanished. Assume another data stream is from $1(E(t))$ to $1(D(t+1))$

which has no self transition. If we adopt XNOR operation on $E(t)$ and $D(t+1)$, $E(t+1)$ will still be 1. Thus the data stream is $1(E(t))$ to $1(E(t+1))$, and the self transition is still none. From this we know when the number of 1's is greater than that of 0's in $D(t+1)$, XNOR operation is suitable for encoding no matter $E(t)$ is 0 or 1. Next, let us analyze the conditions when $D(t+1)$ is 0. Assume a data stream transits from $1(E(t))$ to $0(D(t+1))$ which causes a self transition. If we adopt invert and XNOR operations on $E(t)$ and $D(t+1)$, $E(t+1)$ will become 1. Thus the data stream will switch from $1(E(t))$ to $1(E(t+1))$, such that the self transition is vanished. Assume another data stream is from $0(E(t))$ to $0(D(t+1))$ which has no self transition. If we adopt invert-XNOR operation on $E(t)$ and $D(t+1)$, $E(t+1)$ will still be 0. Thus the data stream is $0(E(t))$ to $0(E(t+1))$, and there is still no self transition. From this we know when the number of 1's is less than that of 0's in $D(t+1)$, invert-XNOR operation is suitable for encoding no matter $E(t)$ is 0 or 1. Table 4.1 shows the analysis and comparison.

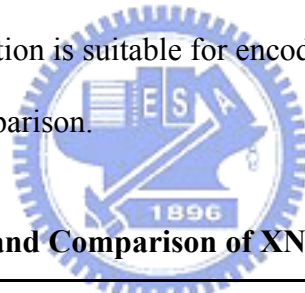


Table 4.1 Analysis and Comparison of XNOR / Invert-XNOR Operations

<i>Encoding</i>	<i>Before Encoding</i>			<i>After Encoding</i>		
	<i>E(t)</i>	<i>D(t+1)</i>	<i>Self Transition</i>	<i>E(t)</i>	<i>E(t+1)</i>	<i>Self Transition</i>
<i>XNOR</i>	<i>0</i>	<i>1</i>	<i>1</i>	<i>0</i>	<i>0</i>	<i>0</i>
	<i>1</i>	<i>1</i>	<i>0</i>	<i>1</i>	<i>1</i>	<i>0</i>
<i>Invert-XNOR</i>	<i>1</i>	<i>0</i>	<i>1</i>	<i>1</i>	<i>1</i>	<i>0</i>
	<i>0</i>	<i>0</i>	<i>0</i>	<i>0</i>	<i>0</i>	<i>0</i>

Since the total number of 1's decides the encoding function in the subset, why not use this rule for all data patterns? Assume $E(t)$ is 00011111 and $D(t+1)$ is 00011111, too. There is no transition in this condition. If we perform XNOR operation on $E(t)$ and $D(t+1)$, $E(t+1)$ is 11111111. The number of self transitions between $E(t)$ and $E(t+1)$ is three which becomes worse than un-encoded pattern. In this case, no encoding is the best

encoding policy. So we consider the number of self transitions in the original pattern first to dynamically choose appropriate encoding functions. We divide the original pattern into two subsets, because the simulation result show this way has a better performance than just encoding the whole bits of the data.

4.2.2 Algorithm of Decoding

Our encoding scheme simple adopts identity, inverse and XNOR operations, which are easy to decode. Consider $E(t+1)$, if it is obtained from identity operation, $D(t+1)$ is equal to $E(t+1)$. If $E(t+1)$ is obtained from inversion operation, $D(t+1)$ is obtained from the inverse of $E(t+1)$. If the subset $E_s(t+1)$ is obtained from XNOR operation (i.e.,

$E_s(t+1) = D_s(t+1) \oplus E_s(t)$), the subset $D_s(t+1)$ is derived from the equation:

$$\begin{aligned}
 E_s(t+1) &= D_s(t+1) \oplus E_s(t) \\
 &= [D_s(t+1) \oplus E_s(t)] \oplus E_s(t) \\
 &= D_s(t+1) \oplus [E_s(t) \oplus E_s(t)] \\
 &= D_s(t+1) \oplus \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \\
 &= D_s(t+1)
 \end{aligned} \tag{4.5}$$

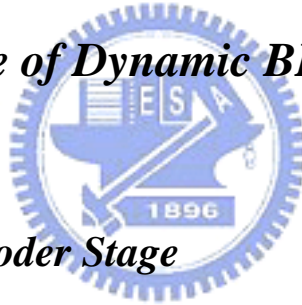
If the subset $E_s(t+1)$ is obtained from invert-XNOR operation (i.e., $E_s(t+1) =$

$[D_s(t+1)]' \oplus E_s(t)$), the subset $D_s(t+1)$ is derived from the equation:

$$\begin{aligned}
& \{ E_s(t+1) \quad E_s(t) \}' \\
& = \{ [[D_s(t+1)]' \quad E_s(t)] \quad E_s(t) \}' \\
& = \{ [D_s(t+1)]' \quad [E_s(t) \quad E_s(t)] \}' \\
& = \{ [D_s(t+1)]' \quad \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \}' & (4.6) \\
& = \{ [D_s(t+1)]' \}' \\
& = D_s(t+1)
\end{aligned}$$

Because the first pattern is not encoded, we easily obtain $D(0)$ which is equal to $E(0)$. Then $E(1), E(2), \dots, E(t), E(t+1), \dots, E(\text{end})$ are decoded in order and $D(1), D(2), \dots, D(t), D(t+1), \dots, D(\text{end})$ are obtained.

4.3 Architecture of Dynamic BI-XNOR Encoding



4.3.1 The Encoder Stage

The encoder structure of dynamic BI-XNOR scheme is shown in Fig.4.3. In this encoding architecture, three control bits C_0, C_1 and C_2 are needed. We divide this architecture into five stages and discuss each stage in detail.

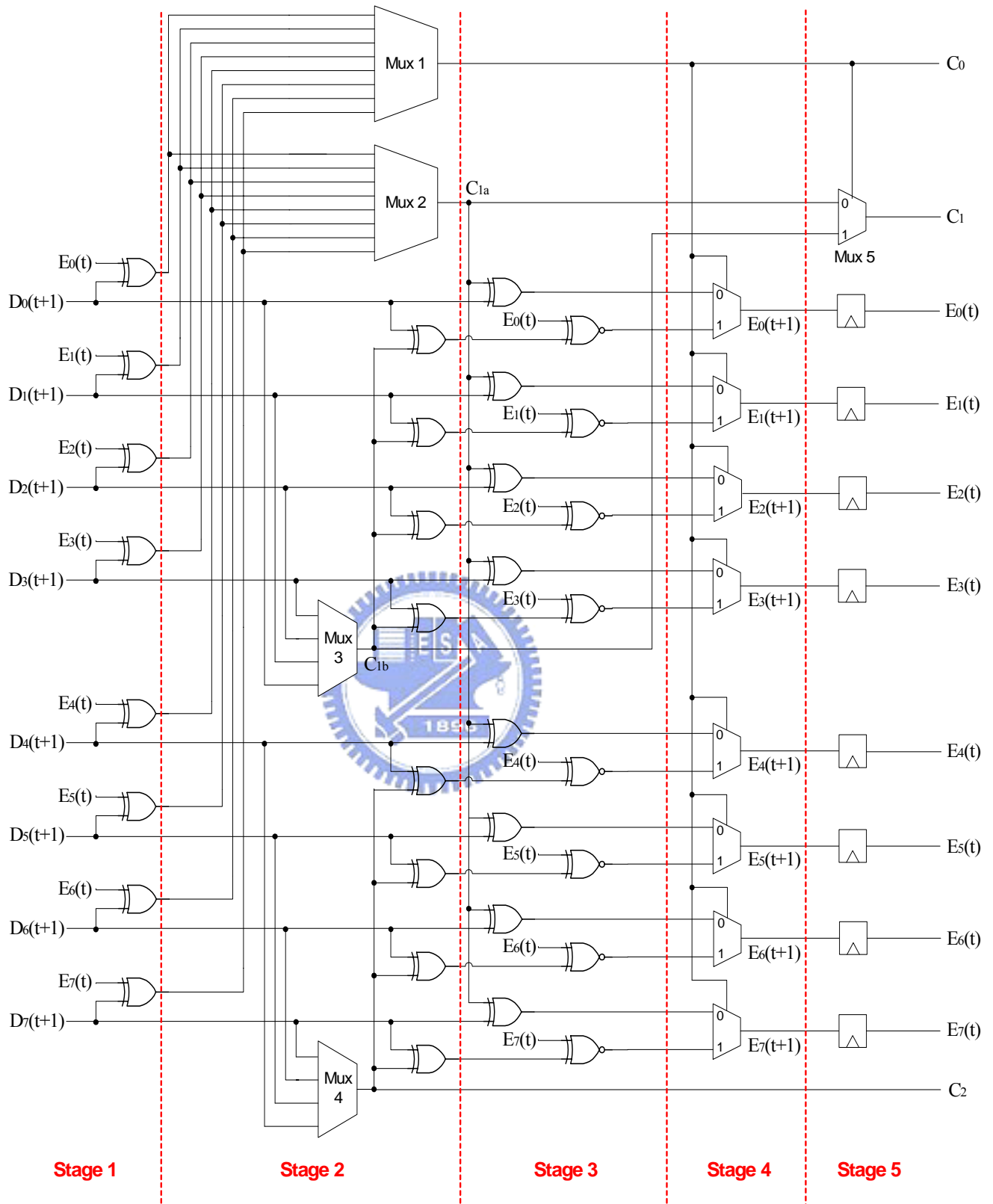


Fig.4.3 Encoder Structure of Dynamic BI-XNOR Encoding Scheme

In stage 1, $D(t+1)$ and $E(t)$ are compared by the set of XOR gates. If they are identical, there is no self transition between $D(t+1)$ and $E(t)$ and the XOR gate will send 0. If they are different, there is a self transition and the XOR gate will send 1.

In stage 2, multiplexer 1 calculates the number of self transitions between $D(t+1)$ and $E(t)$ and sends output C_0 . If the number of self transitions is 0 to 2 or 6 to 8, C_0 is set to 0, otherwise C_0 is set to 1. Multiplexer 2 calculates the number of self transitions between $D(t+1)$ and $E(t)$ and sends output C_{1a} . If the number of self transitions is 6 to 8, C_{1a} is set to 1, otherwise C_{1a} is set to 0. Multiplexer 3 gathers a statistic for the four MSBs of $D(t+1)$ and sends output C_{1b} . If the number of 1's is greater than or equal to the number of 0's (i.e., the number of 1's is greater than or equal to 2), C_{1b} is set to 0. Otherwise C_{1b} is set to 1. Multiplexer 4 does the same operations with multiplexer 3 for the four LBSs of $D(t+1)$ and sends output C_2 . C_{1b} and C_2 decides whether $D(t+1)$ should be inverted or not in the set of XOR gates within stage 2.

In stage 3, C_{1a} decides whether $D(t+1)$ should be inverted or not in the set of XOR gates. $D(t+1)$ or $[D(t+1)]'$ executes XNOR operation with $E(t)$.

In stage 4, the set of 2-to-1 multiplexers controlled by C_0 are needed. They select the correct value of $E(t+1)$.

In stage 5, the output $E(t)$ is sent back to the input in the next cycle time. In multiplexer 5, either C_{1a} or C_{1b} is sent as an output C_1 because C_{1a} is valid when C_0 is 0 and C_{1b} is valid when C_0 is 1.

4.3.2 The Decoder Stage

The decoder structure of dynamic BI-XNOR is shown in Fig.4.4. We divide this architecture into three stages and discuss each stage in detail.

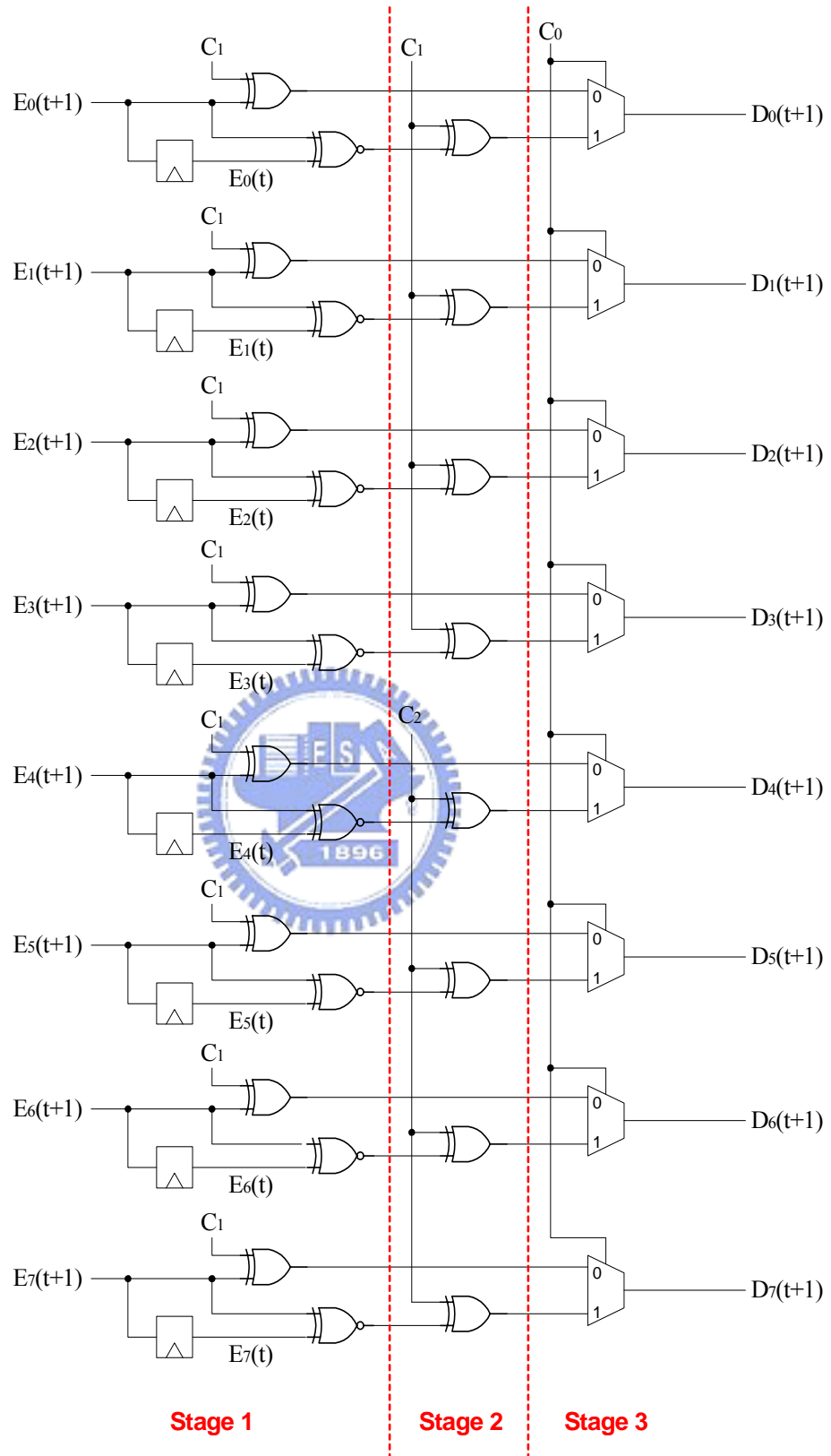


Fig.4.4 Decoder Structure of Dynamic BI-XNOR Encoding Scheme

In stage 1, $E(t+1)$ execute XNOR operation with $E(t)$ in the set of XNOR gates. C_1 indicates whether $E(t+1)$ should be inverted or not in the set of XOR gates.

In stage 2, C_1 indicates whether the four MSBs should be inverted or not and C_2 indicates whether the four LSBs should be inverted or not in the set of XOR gates. Note that C_1 of stage 2 is different from C_1 of stage 1. C_1 of stage 1 is C_{1a} in the encoder and C_1 of stage 2 is C_{1b} in the encoder. They are not both valid in the same cycle time.

In stage 3, the set of 2-to-1 multiplexers controlled by C_0 select the valid value of $D(t+1)$. The decoding is completed here.



Chapter 5

Simulation Results

In this chapter, we will make comparisons of self transitions and correlative transitions for dynamic BI-XNOR encoding and other encoding schemes mentioned in chapter 3.

5.1 Simulation Overview



We will analyze an 8×2 data matrix

A_{11}	A_{12}
A_{21}	A_{22}
A_{31}	A_{32}
A_{41}	A_{42}
A_{51}	A_{52}
A_{61}	A_{62}
A_{71}	A_{72}
A_{81}	A_{82}

where the first column A_{x1} denotes the previous encoded pattern $E(t)$ and the second column A_{x2} denotes the current pattern $D(t+1)$ waiting for encoding. We calculate the self transitions between $E(t)$ and $D(t+1)$ before $D(t+1)$ is encoded, and then calculate the self transitions between $E(t)$ and $E(t+1)$ again after $D(t+1)$ is encoded. We will focus on the original self transitions and analyze the improvement in each condition when there are 0, 1, 2, ..., 8 transitions.

5.2 Simulation Results

5.2.1 Self Transitions

Fig.5.1 shows the conditional averaged self transitions after encoding. The x-axis denotes the condition when there are N self transitions between $E(t)$ and $D(t+1)$, the y-axis denotes the average self transitions between $E(t)$ and $E(t+1)$ after encoding.

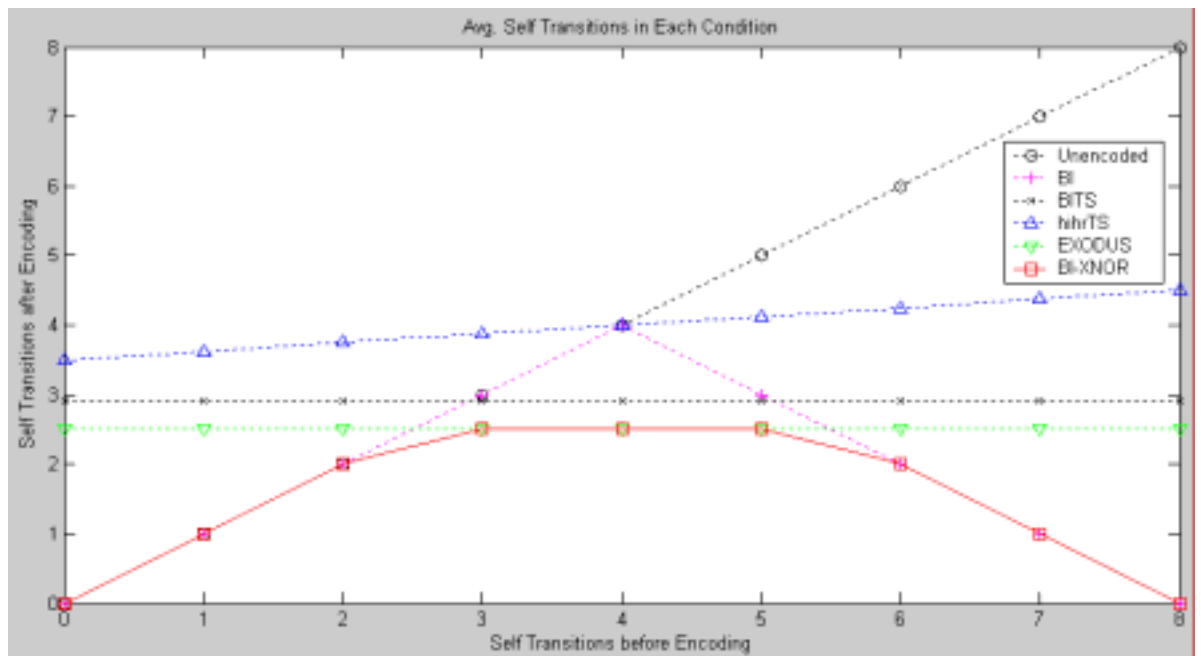


Fig.5.1 Averaged Conditional Self Transitions after Encoding

Fig.5.2 shows the conditional averaged self transition reduction after encoding. The x-axis denotes the condition when there are N self transitions between $E(t)$ and $D(t+1)$, the y-axis denotes the averaged self transition reduction between $E(t)$ and $E(t+1)$ after encoding.

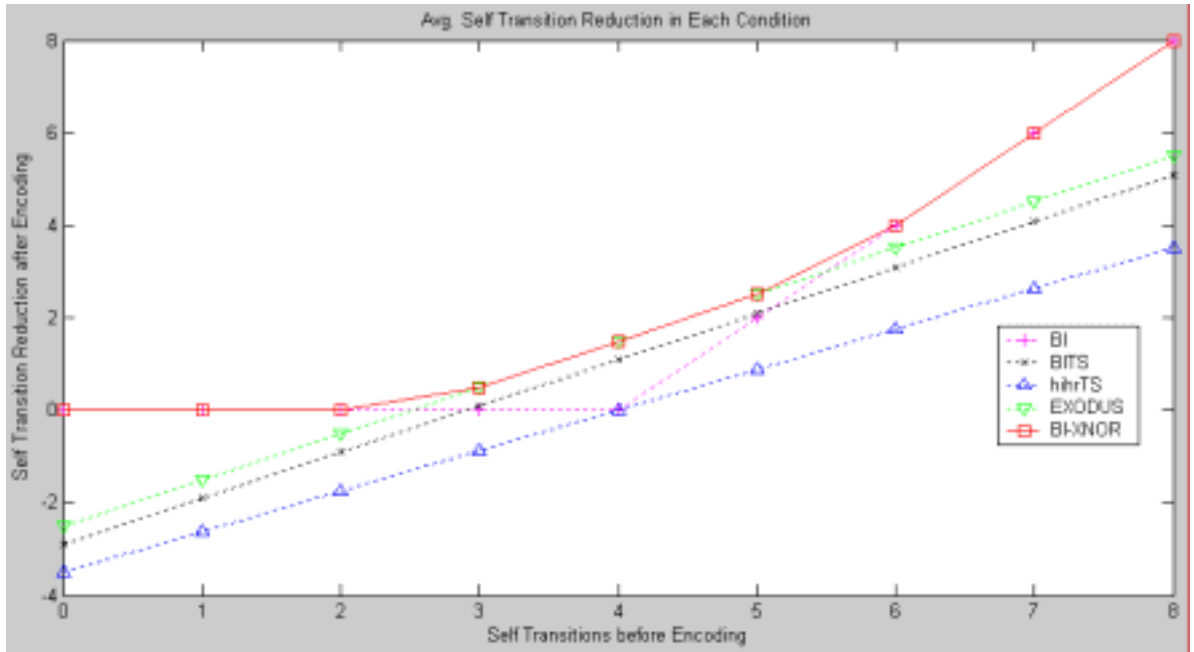


Fig.5.2 Averaged Conditional Self Transition Reduction after Encoding

Compared with BITS and hihrTS schemes, our scheme is superior to these two schemes in all conditions regardless of the number of self transitions in x-axis. Compared with EXODUS encoding, our scheme has similar performance with EXODUS when there are 3, 4 and 5 self transitions in x-axis. However, our scheme has better result than EXODUS in other conditions (i.e., when there are 0, 1, 2, 6, 7 and 8 self transitions in x-axis). Compared with BI encoding, our scheme has similar performance with BI encoding when there are 0, 1, 2, 6, 7 and 8 self transitions in x-axis since our scheme has the same encoding rule in these conditions. But our scheme has better result than it when there are 3, 4 and 5 self transitions in x-axis. Table 5.1 shows the number of averaged conditional self transitions for each encoding scheme. This table also shows the probability of each condition when there are N self transitions before encoding.

Table 5.1 No. of Averaged Conditional Self Transitions

		<i>Self Transitions after Encoding</i>					
<i>Self Transitions before Encoding</i>	<i>Probability</i>	<i>Un-encoded</i>	<i>BI</i>	<i>BITS</i>	<i>hihrTS</i>	<i>EXODUS</i>	<i>BI-XNOR</i>
0	0.39%	0	0	2.91	3.5	2.5	0
1	3.13%	1	1	2.91	3.625	2.5	1
2	10.94%	2	2	2.91	3.75	2.5	2
3	21.88%	3	3	2.91	3.875	2.5	2.5
4	27.34%	4	4	2.91	4	2.5	2.5
5	21.88%	5	3	2.91	4.125	2.5	2.5
6	10.94%	6	2	2.91	4.25	2.5	2
7	3.13%	7	1	2.91	4.375	2.5	1
8	0.39%	8	0	2.91	4.5	2.5	0

Observing BITS, hihrTS and EXODUS encoding schemes, these three schemes make the self transitions become worse than that of un-encoded data when there are 0, 1 and 2 self transitions in x-axis. HihrTS encoding even also has worse result when there are 3 self transitions in x-axis. The averaged self transitions are equal to the weighted average of conditional self transitions in Table 5.1.

Since our scheme has the least self transitions and the most reduction of self transitions in all conditions, the averaged self transitions of our scheme will also be the least of all. Fig.5.3 shows the comparison of averaged self transitions of all encoding schemes. Fig.5.4 shows the percentage reduction of the averaged self transitions. Table 5.2 shows the number of averaged self transitions in each encoding scheme.

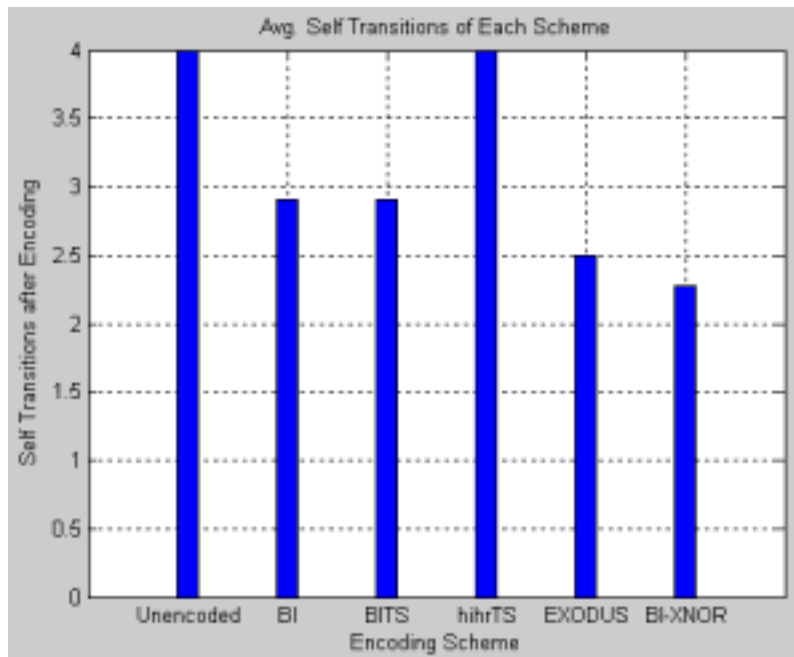


Fig.5.3 Averaged Self Transitions after Encoding

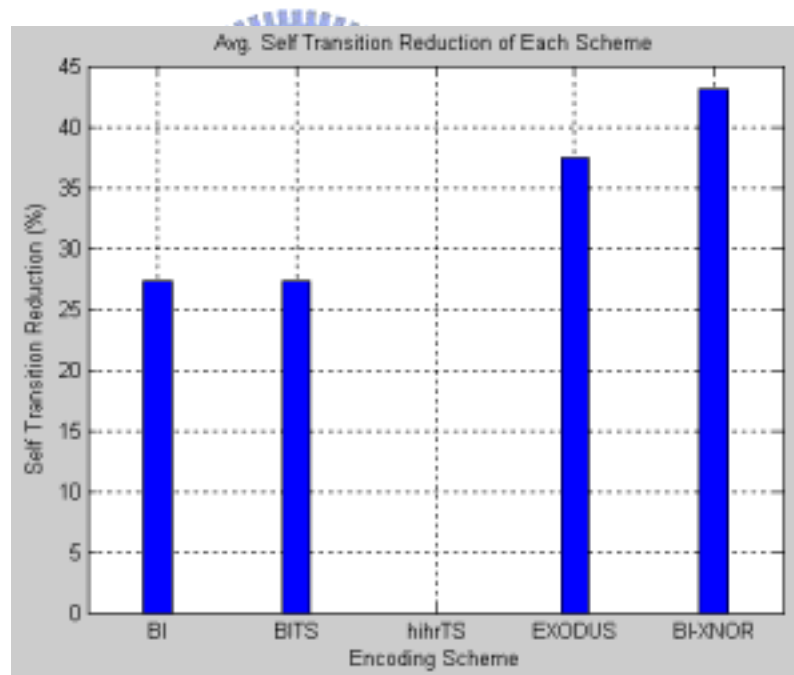


Fig.5.4 Averaged Self Transition Reduction after Encoding

Table 5.2 No. of Averaged Self Transitions for Each Encoding Scheme

Encoding Scheme	Unencoded	BI	BITS	hihrTS	EXODUS	BI-XNOR
Self Transitions	4	2.91	2.91	4	2.5	2.28
Percentage of Reduction	0%	27.3%	27.3%	0%	37.5%	43%

HihrTS encoding scheme has not any improvement in the reduction of averaged self transitions. It is because this scheme use the first bit as the encoding hint. Since we use all possible test patterns here, hihrTS encoding may loose its superiority in this simulation.

5.2.2 Correlative Transitions

Fig.5.5 shows the averaged conditional correlative transitions after encoding. The x-axis denotes the condition when there are N self transitions between E(t) and D(t+1), the y-axis denotes the averaged self transitions between E(t) and E(t+1) after encoding.

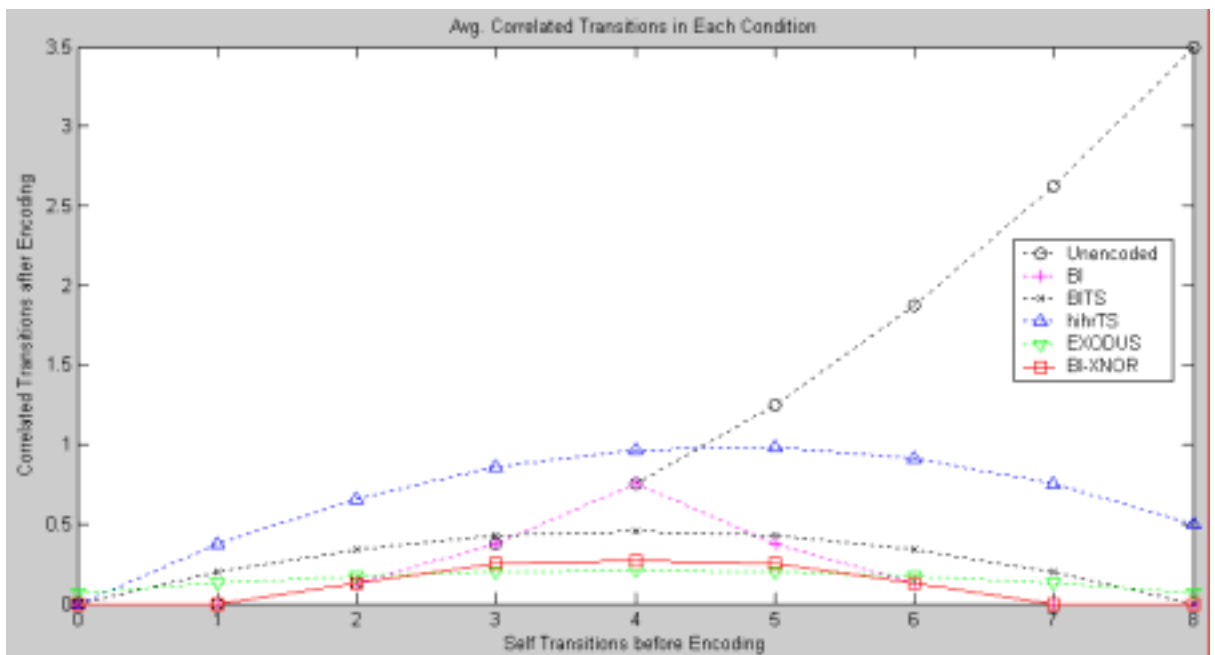


Fig.5.5 Averaged Conditional Correlative Transitions after Encoding

Fig.5.6 shows the averaged conditional correlative transition reduction after encoding. The x-axis denotes the condition when there are N self transitions between $E(t)$ and $D(t+1)$, the y-axis denotes the averaged correlative transition reduction between $E(t)$ and $E(t+1)$ after encoding.

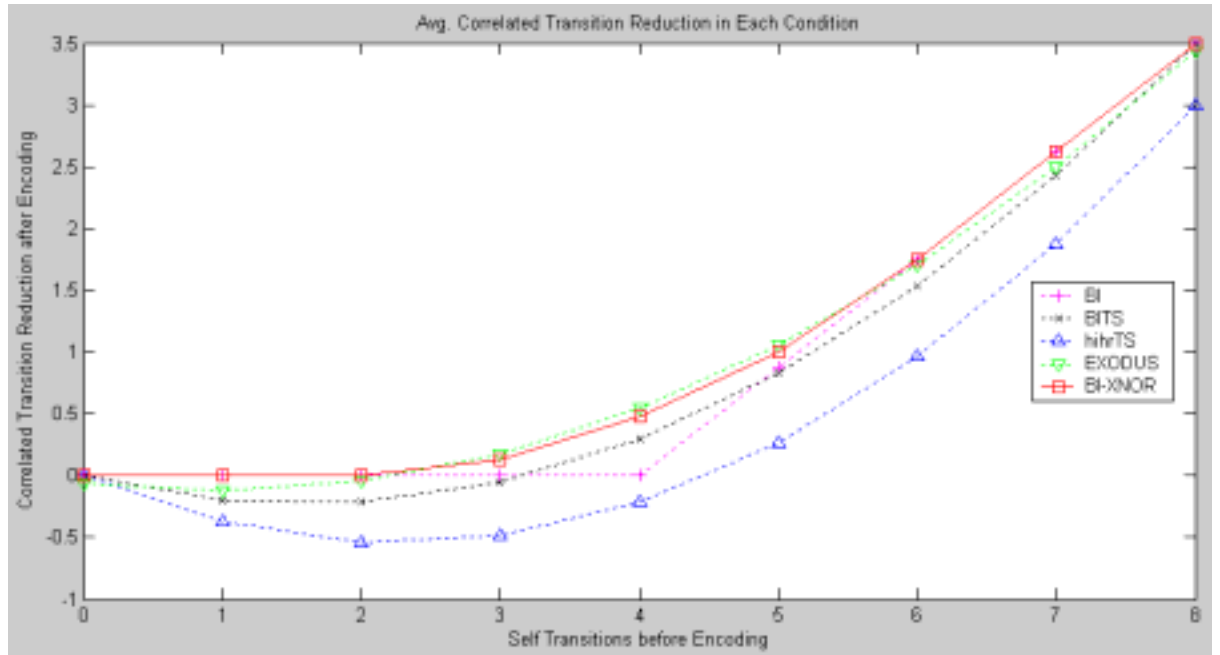


Fig.5.6 Averaged Conditional Correlative Transition Reduction after Encoding

Compared with BITS and hihrTS schemes, our scheme is superior or similar to these two schemes in all conditions regardless of the number of self transitions in x-axis. Compared with BI encoding, our scheme has similar performance with BI encoding when there are 0, 1, 2, 6, 7 and 8 self transitions in x-axis because our scheme has the same encoding rule in these conditions. But our scheme has better result than it when there are 3, 4 and 5 self transitions in x-axis. Compared with EXODUS encoding, our scheme is superior to EXODUS encoding when there are 0, 1, 2, 6, 7 and 8 self transitions in x-axis but inferior to EXODUS encoding when there are 3, 4 and 5 self transitions. Table 5.3 shows the number of averaged conditional correlative transitions for each encoding scheme.

Table 5.3 No. of Averaged Conditional Correlative Transitions

		<i>Correlative Transitions after Encoding</i>					
<i>Self Transitions before Encoding</i>	<i>Probability</i>	<i>Un-encoded</i>	<i>BI</i>	<i>BITS</i>	<i>hihrTS</i>	<i>EXODUS</i>	<i>BI-XNOR</i>
0	0.39%	0	0	0	0	0.063	0
1	3.13%	0	0	0.198	0.375	0.125	0
2	10.94%	0.125	0.125	0.340	0.661	0.170	0.125
3	21.88%	0.375	0.375	0.425	0.857	0.196	0.251
4	27.34%	0.750	0.750	0.453	0.964	0.205	0.266
5	21.88%	1.250	0.375	0.425	0.982	0.196	0.251
6	10.94%	1.875	0.125	0.340	0.911	0.170	0.125
7	3.13%	2.625	0	0.198	0.75	0.125	0
8	0.39%	3.500	0	0	0.5	0.063	0

Observing BITS, hihrTS and EXODUS encoding schemes, BITS encoding makes the correlative transitions become worse than that of un-encoded data when there are 1, 2 and 3 self transitions in x-axis. HihrTS encoding makes the correlative transitions become worse when there are 1, 2, 3 and 4 self transitions in x-axis. EXODUS encoding makes the correlative transitions become worse when there are 0, 1 and 2 self transitions in x-axis. The averaged correlative transitions are equal to the weighted average of conditional correlative transitions in Table 5.3.

Fig.5.7 shows the comparison of averaged correlative transitions of all encoding schemes. Fig.5.8 shows the percentage reduction of averaged correlative transitions. Table 5.4 shows the number of averaged correlative transitions in each encoding scheme.

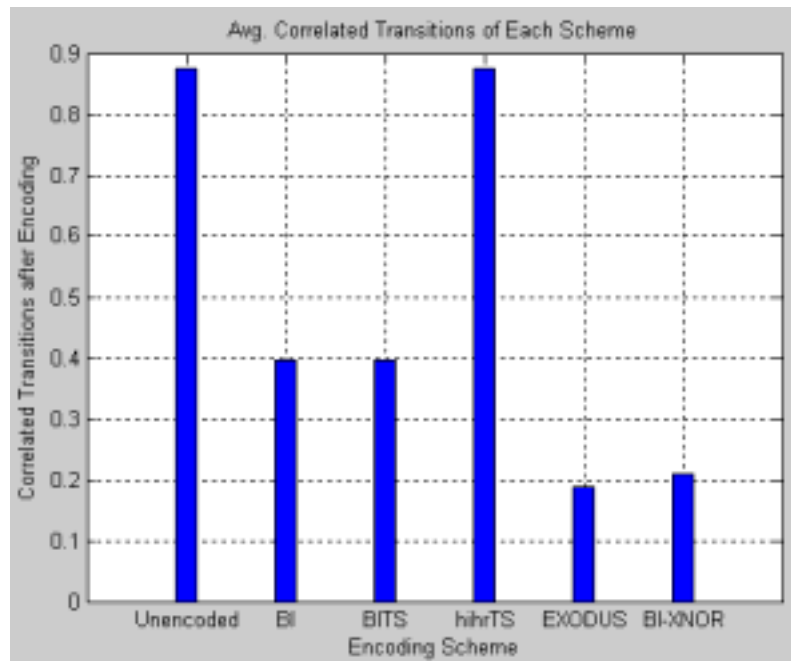


Fig.5.7 Averaged Correlative Transitions after Encoding

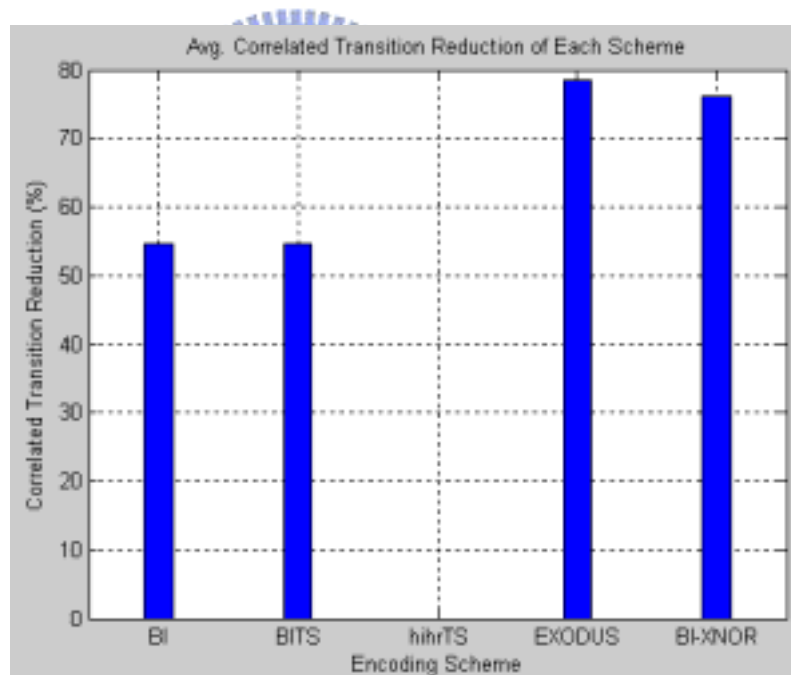


Fig.5.8 Averaged Correlative Transition Reduction after Encoding

Table 5.4 No. of Averaged Correlative Transitions for Each Encoding Scheme

Encoding Scheme	Unencoded	BI	BITS	Hihr-TS	EXODUS	BI-XNOR
Correlative Transitions	0.875	0.396	0.396	0.875	0.188	0.209
Percentage of Reduction	0%	54.7%	54.7%	0%	78.5%	76.1%

Though our scheme is inferior to EXODUS encoding in averaged correlative transitions, our scheme assures that the conditional correlative transitions will never become worse. Because EXODUS encoding makes the correlative transitions become worse when there are 0, 1 and 2 self transitions in x-axis.

5.3 Experiment with Sample Patterns

To evaluate the efficiency of our encoding, we perform experiments with random data streams, different types of audio input, image files and image files with DCT transformation.

5.3.1 Random Data Streams



We simulate all schemes with random data streams. Each data stream has 100,000 patterns and the bus width is eight bits. The experimental result is shown in Table 5.5. It is roundly matching our simulation result in Table 5.2 and Table 5.4. This proves our simulation method by an 8×2 matrix is correct.

Table 5.5 Experimental Result of Random Data Streams

<i>Encoding Scheme</i>	<i>Unencoded</i>	<i>BI</i>	<i>BITS</i>	<i>hihrTS</i>	<i>EXODUS</i>	<i>BI-XNOR</i>
<i>Self Transitions</i>	400,100	290,782	290,846	399,404	249,930	225,954
<i>Percentage of Reduction</i>	-	27.3%	27.3%	0.2%	37.5%	44.0%
<i>Correlative Transitions</i>	87,870	39,955	39,833	87,501	18,683	22,459
<i>Percentage of Reduction</i>	-	54.5%	54.7%	0.4%	78.7%	74.4%

the experimental result. Fig.5.13 is the waveform of a noisy speech file and Table 5.9 is the experimental result. Fig.5.14 is the waveform of a song file and Table 5.10 is the experimental result.

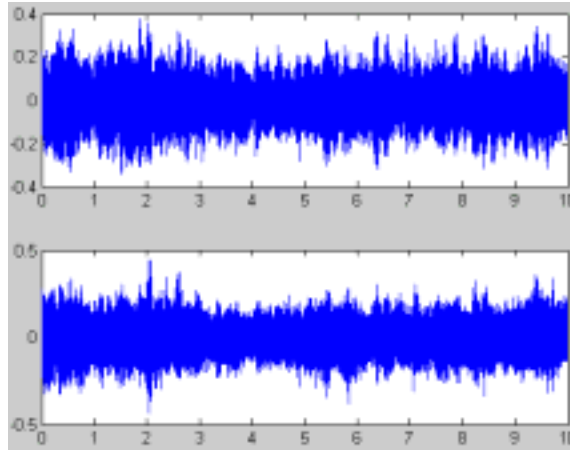


Fig.5.10 Waveform of a Classic Musical File

Table 5.6 Experimental Result of Classical Music

<i>Encoding Scheme</i>	<i>Unencoded</i>	<i>BI</i>	<i>BITS</i>	<i>hihrTS</i>	<i>EXODUS</i>	<i>BI-XNOR</i>
<i>Self Transitions</i>	7,039,529	5,124,745	4,132,657	5,531,791	4,978,785	3,308,088
<i>Percentage of Reduction</i>	-	27.2%	41.3%	21.4%	29.3%	53.0%
<i>Correlative Transitions</i>	885,246	402,248	586,994	1,009,538	614,085	269,986
<i>Percentage of Reduction</i>	-	54.6%	33.7%	-14.0%	30.6%	69.5%

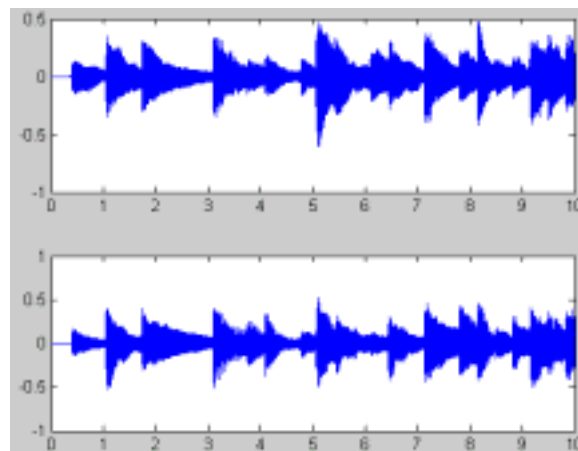


Fig.5.11 Waveform of a Pop Music File

Table 5.7 Experimental Result of Pop Music

<i>Encoding Scheme</i>	<i>Unencoded</i>	<i>BI</i>	<i>BITS</i>	<i>hihrTS</i>	<i>EXODUS</i>	<i>BI-XNOR</i>
<i>Self Transitions</i>	6,765,271	4,930,279	3,938,904	5,284,240	5,030,128	3,189,566
<i>Percentage of Reduction</i>	-	27.1%	41.8%	21.9%	25.6%	52.9%
<i>CorrelativeTransitions</i>	832,480	379,215	555,660	962,606	588,521	257,796
<i>Percentage of Reduction</i>	-	54.4%	33.3%	-15.6%	29.3%	69.0%

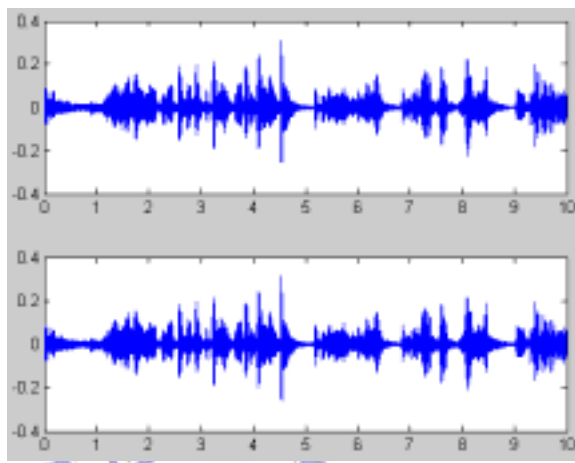


Fig.5.12 Waveform of a Normal Speech File

Table 5.8 Experimental Result of Normal Speech

<i>Encoding Scheme</i>	<i>Unencoded</i>	<i>BI</i>	<i>BITS</i>	<i>hihrTS</i>	<i>EXODUS</i>	<i>BI-XNOR</i>
<i>Self Transitions</i>	6,878,350	5,119,560	3,271,491	4,567,920	5,216,049	2,912,749
<i>Percentage of Reduction</i>	-	25.6%	52.4%	33.6%	24.2%	57.7%
<i>CorrelativeTransitions</i>	412,745	192,404	422,230	828,932	791,055	204,986
<i>Percentage of Reduction</i>	-	53.4%	-2.3%	-100.8%	-91.7%	50.3%

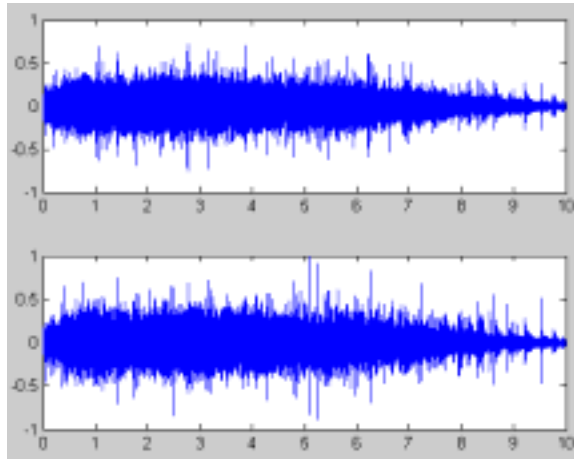


Fig.5.13 Waveform of a Noisy Speech File

Table 5.9 Experimental Result of Noisy Speech

<i>Encoding Scheme</i>	<i>Unencoded</i>	<i>BI</i>	<i>BITS</i>	<i>hihrTS</i>	<i>EXODUS</i>	<i>BI-XNOR</i>
<i>Self Transitions</i>	7,034,513	5,126,147	4,144,264	5,544,866	4,913,900	3,315,180
<i>Percentage of Reduction</i>	-	27.1%	41.1%	21.2%	30.1%	52.9%
<i>Correlative Transitions</i>	888,533	404,188	590,707	1,016,636	596,681	275,707
<i>Percentage of Reduction</i>	-	54.5%	33.5%	-14.4%	32.8%	69.0%

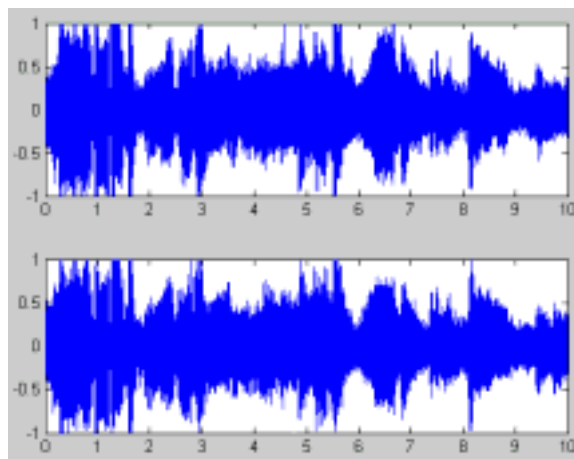


Fig.5.14 Waveform of a Song File

Table 5.10 Experimental Result of a Song File

<i>Encoding Scheme</i>	<i>Unencoded</i>	<i>BI</i>	<i>BITS</i>	<i>hihrTS</i>	<i>EXODUS</i>	<i>BI-XNOR</i>
<i>Self Transitions</i>	7,049,922	5,124,058	4,690,920	6,172,301	4,553,272	3,599,330
<i>Percentage of Reduction</i>	-	27.3%	33.5%	12.4%	35.4%	48.9%
<i>Correlative Transitions</i>	1,203,822	545,492	693,539	1,172,629	457,195	330,969
<i>Percentage of Reduction</i>	-	54.7%	42.4%	2.6%	62.0%	72.5%

From the above experiments, we summarize them in Table 5.11 and Table 5.12. For the audio inputs, our scheme has a better improvement than other schemes in both self and correlative transitions.

Table 5.11 Averaged Self Transition Reduction of Audio Files

	<i>Unencoded</i>	<i>BI</i>	<i>BITS</i>	<i>hihrTS</i>	<i>EXODUS</i>	<i>BI-XNOR</i>
<i>Classic Music</i>	7,039,529	5,124,745	4,132,657	5,531,791	4,978,785	3,308,088
<i>Pop Music</i>	6,765,271	4,930,279	3,938,904	5,284,240	5,030,128	3,189,566
<i>Normal Speech</i>	6,878,350	5,119,560	3,271,491	4,567,920	5,216,049	2,912,749
<i>Noisy Speech</i>	7,034,513	5,126,147	4,144,264	5,544,866	4,913,900	3,315,180
<i>Song</i>	7,049,922	5,124,058	4,690,920	6,172,301	4,553,272	3,599,330
<i>Average</i>	6,953,517	5,084,958	4,035,647	5,420,224	4,938,427	3,264,983
<i>Reduction</i>	-	26.9%	42.0%	22.1%	29.0%	53.0%

Table 5.12 Averaged Correlative Transition Reduction of Audio Files

	<i>Unencoded</i>	<i>BI</i>	<i>BITS</i>	<i>hhrTS</i>	<i>EXODUS</i>	<i>BI-XNOR</i>
<i>Classic Music</i>	885,246	402,248	586,994	1,009,538	614,085	269,986
<i>Pop Music</i>	832,480	379,215	555,660	962,606	588,521	257,796
<i>Normal Speech</i>	412,745	192,404	422,230	828,932	791,055	204,986
<i>Noisy Speech</i>	888,533	404,188	590,707	1,016,636	596,681	275,707
<i>Song</i>	1,203,822	545,492	693,539	1,172,629	457,195	330,969
<i>Average</i>	844,565	384,709	569,826	998,068	609,507	267,889
<i>Reduction</i>	-	54.4%	32.5%	-18.2%	27.8%	68.3%

5.3.3 Gray Image Files

We perform experiments with three gray image files. The input signals are from RAW files. All gray image files have the resolution of 256×256. Fig.5.15 is the first sample image file. This image is read as a 256×256 matrix, each element of this matrix is an integer ranging from 0 to 255. We use an unsigned 8-bit integer to represent each element. Then we transmit this image file and compare all encoding schemes. Table 5.13 is the experimental result of gray image 1. Fig.5.16 is the second sample image file and Table 5.14 is its experimental result. Fig.5.17 is the third sample image file and Table 5.15 is its experimental result.



Fig.5.15 Gray Image File 1

Table 5.13 Experimental Result of Gray Image File 1

<i>Encoding Scheme</i>	<i>Unencoded</i>	<i>BI</i>	<i>BITS</i>	<i>hihrTS</i>	<i>EXODUS</i>	<i>BI-XNOR</i>
<i>Self Transitions</i>	174,721	144,795	197,200	258,165	160,138	140,769
<i>Percentage of Reduction</i>	-	17.1%	-12.9%	-47.8%	8.3%	19.4%
<i>CorrelativeTransitions</i>	44,165	32,626	26,406	60,585	8,083	12,737
<i>Percentage of Reduction</i>	-	26.1%	40.2%	-37.2%	81.7%	71.2%

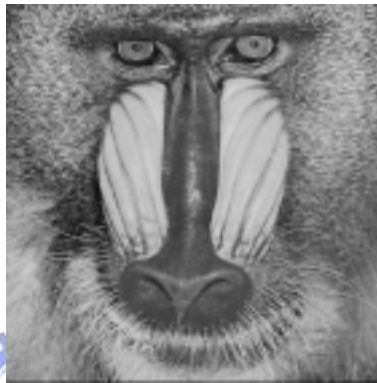


Fig.5.16 Gray Image File 2

Table 5.14 Experimental Result of Gray Image File 2

<i>Encoding Scheme</i>	<i>Unencoded</i>	<i>BI</i>	<i>BITS</i>	<i>hihrTS</i>	<i>EXODUS</i>	<i>BI-XNOR</i>
<i>Self Transitions</i>	211,112	164,844	202,728	276,158	168,098	156,068
<i>Percentage of Reduction</i>	-	21.9%	4.0%	-30.8%	20.4%	26.1%
<i>CorrelativeTransitions</i>	54,615	35,384	24,342	66,680	7,965	14,275
<i>Percentage of Reduction</i>	-	35.2%	55.4%	-22.1%	85.4%	73.9%



Fig.5.17 Gray Image File 3

Table 5.15 Experimental Result of Gray Image File 3

<i>Encoding Scheme</i>	<i>Unencoded</i>	<i>BI</i>	<i>BITS</i>	<i>hihrTS</i>	<i>EXODUS</i>	<i>BI-XNOR</i>
<i>Self Transitions</i>	155,861	132,757	189,373	223,724	152,637	131,067
<i>Percentage of Reduction</i>	-	14.8%	-21.5%	-43.5%	2.1%	15.9%
<i>Correlative Transitions</i>	39,101	29,645	26,669	47,672	13,644	13,198
<i>Percentage of Reduction</i>	-	24.2%	31.8%	-21.9%	65.1%	66.2%

We summarize these three experimental results in Table 5.16 and Table 5.17. For the gray image files, our scheme has a better improvement than other schemes in self transitions. In the reduction of correlative transitions, our scheme is inferior to EXODUS by 8% but superior to other three schemes.

Table 5.16 Averaged Self Transition Reduction of Gray Image Files

	<i>Unencoded</i>	<i>BI</i>	<i>BITS</i>	<i>hihrTS</i>	<i>EXODUS</i>	<i>BI-XNOR</i>
<i>Gray Image 1</i>	174,721	144,795	197,200	258,165	160,138	140,769
<i>Gray Image 2</i>	211,112	164,844	202,728	276,158	168,098	156,068
<i>Gray Image 3</i>	155,861	132,757	189,373	223,724	152,637	131,067
<i>Average</i>	108,339	88,479	117,860	151,609	96,175	85,581
<i>Reduction</i>	-	18.3%	-8.8%	-39.9%	11.2%	21.0%

Table 5.17 Averaged Correlative Transition Reduction of Gray Image Files

	<i>Unencoded</i>	<i>BI</i>	<i>BITS</i>	<i>hihrTS</i>	<i>EXODUS</i>	<i>BI-XNOR</i>
<i>Gray Image 1</i>	44,165	32,626	26,406	60,585	8,083	12,737
<i>Gray Image 2</i>	54,615	35,384	24,342	66,680	7,965	14,275
<i>Gray Image 3</i>	39,101	29,645	26,669	47,672	13,644	13,198
<i>Average</i>	27,576	19,531	15,483	34,987	5,938	8,042
<i>Reduction</i>	-	29.2%	43.9%	-26.9%	78.5%	70.8%

5.3.4 Color Image Files

We perform the experiment with three color image files. The input signals are from JPEG files. All color image files have the resolution of 256×256 . Fig.5.18 is the color image file 1. This image is read as a $256 \times 256 \times 3$ matrix, each element of this matrix is an integer ranging from 0 to 255. The $256 \times 256 \times 1$ denotes red (R) elements, the $256 \times 256 \times 2$ denotes green (G) elements and the $256 \times 256 \times 3$ denotes blue (B) elements of the image. We use an unsigned 8-bit integer to represent each element and transmit R, G and B elements of this image file in order and compare all encoding schemes. Table 5.18 is the experimental result. Fig.5.19 is the color image file 2 and Table 5.19 is the experimental result. Fig.5.19 is the color image file 3 and Table 5.19 is the experimental result.



Fig.5.18 Color Image File 1

Table 5.18 Experimental Result of Color Image File 1

<i>Encoding Scheme</i>	<i>Unencoded</i>	<i>BI</i>	<i>BITS</i>	<i>hihrTS</i>	<i>EXODUS</i>	<i>BI-XNOR</i>
<i>Self Transitions</i>	497,581	411,391	593,922	766,689	490,460	414,331
<i>Percentage of Reduction</i>	-	17.3%	-19.4%	-54.1%	1.4%	16.7%
<i>Correlative Transitions</i>	127,228	94,341	77,325	177,376	28,071	40,223
<i>Percentage of Reduction</i>	-	25.8%	39.2%	-39.4%	77.9%	68.4%



Fig.5.19 Color Image File 2

Table 5.19 Experimental Result of Color Image File 2

<i>Encoding Scheme</i>	<i>Unencoded</i>	<i>BI</i>	<i>BITS</i>	<i>hihrTS</i>	<i>EXODUS</i>	<i>BI-XNOR</i>
<i>Self Transitions</i>	555,496	458,354	607,542	797,543	485,404	452,955
<i>Percentage of Reduction</i>	-	17.5%	-9.4%	-43.6%	12.6%	18.5%
<i>Correlative Transitions</i>	140,901	105,181	79,021	186,689	25,359	39,743
<i>Percentage of Reduction</i>	-	25.4%	43.9%	-32.5%	82.0%	71.8%

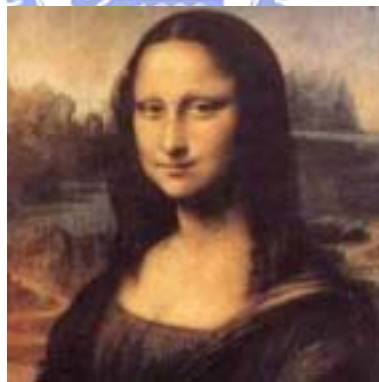


Fig.5.20 Color Image File 3

Table 5.20 Experimental Result of Color Image File 3

<i>Encoding Scheme</i>	<i>Unencoded</i>	<i>BI</i>	<i>BITS</i>	<i>hihrTS</i>	<i>EXODUS</i>	<i>BI-XNOR</i>
<i>Self Transitions</i>	439,664	379,122	600,933	717,710	451,243	398,815
<i>Percentage of Reduction</i>	-	13.8%	-36.7%	-63.2%	-2.6%	9.3%
<i>Correlative Transitions</i>	111,997	91,017	87,190	153,378	30,326	36,839
<i>Percentage of Reduction</i>	-	18.7%	22.1%	-36.9%	72.9%	67.1%

We summarize these three experimental results in Table 5.21 and Table 5.22. In the reduction of self transitions, our scheme is inferior to BI by 1.4% but superior to other three schemes. In the reduction of correlative transitions, our scheme is inferior to EXODUS by 8.5% but superior to other three schemes.

Table 5.21 Averaged Self Transition Reduction of Color Image Files

	<i>Unencoded</i>	<i>BI</i>	<i>BITS</i>	<i>hihrTS</i>	<i>EXODUS</i>	<i>BI-XNOR</i>
<i>Color Image 1</i>	497,581	411,391	593,922	766,689	490,460	414,331
<i>Color Image 2</i>	555,496	458,354	607,542	797,543	485,404	452,955
<i>Color Image 3</i>	439,664	379,122	600,933	717,710	451,243	398,815
<i>Average</i>	298,548	249,773	360,479	456,388	285,421	253,220
<i>Reduction</i>	-	16.3%	-20.7%	-52.9%	4.4%	15.2%

Table 5.22 Averaged Correlative Transition Reduction of Color Image Files

	<i>Unencoded</i>	<i>BI</i>	<i>BITS</i>	<i>hihrTS</i>	<i>EXODUS</i>	<i>BI-XNOR</i>
<i>Color Image 1</i>	127,228	94,341	77,325	177,376	28,071	40,223
<i>Color Image 2</i>	140,901	105,181	79,021	186,689	25,359	39,743
<i>Color Image 3</i>	111,997	91,017	87,190	153,378	30,326	36,839
<i>Average</i>	76,025	58,108	48,707	103,489	16,751	23,361
<i>Reduction</i>	-	23.6%	35.9%	-36.1%	78.0%	69.3%

5.3.5 DCT-transformed Image Files

We use Fig.5.15, Fig.5.16 and Fig.5.17 as the input images, but we perform DCT transformation before transmitting these images. We divide this image into 8×8 blocks and perform DCT transformation on each block. Then we round off each parameter to

integer and use 8-bit two's complement to represent each parameter. We shift all parameters to eight bits and use zigzag scan to transmit all elements in 8-bit wide bus. The DCT transformation makes the appearance of zero become more frequent. Table 5.23, Table 5.24 and Table 5.25 are the experimental results.

Table 5.23 Experimental Result of DCT-transformed Image File 1

<i>Encoding Scheme</i>	<i>Unencoded</i>	<i>BI</i>	<i>BITS</i>	<i>hhrTS</i>	<i>EXODUS</i>	<i>BI-XNOR</i>
<i>Self Transitions</i>	54,518	16,378	10,542	16,872	254,254	12,368
<i>Percentage of Reduction</i>	-	70.0%	80.7%	69.1%	-366.4%	77.3%
<i>Correlative Transitions</i>	1,542	1,122	1,044	1,203	60,451	693
<i>Percentage of Reduction</i>	-	27.2%	32.3%	22.0%	-3820.3%	55.1%

Table 5.24 Experimental Result of DCT-transformed Image File 2

<i>Encoding Scheme</i>	<i>Unencoded</i>	<i>BI</i>	<i>BITS</i>	<i>hhrTS</i>	<i>EXODUS</i>	<i>BI-XNOR</i>
<i>Self Transitions</i>	129,114	30,446	19,605	34,993	245,507	25,523
<i>Percentage of Reduction</i>	-	76.4%	84.8%	72.9%	-90.1%	80.2%
<i>Correlative Transitions</i>	2,879	2,230	1,484	1,725	56,341	1,580
<i>Percentage of Reduction</i>	-	22.5%	48.5%	40.1%	-1857.0%	45.1%

Table 5.25 Experimental Result of DCT-transformed Image File 3

<i>Encoding Scheme</i>	<i>Unencoded</i>	<i>BI</i>	<i>BITS</i>	<i>hhrTS</i>	<i>EXODUS</i>	<i>BI-XNOR</i>
<i>Self Transitions</i>	52,857	16,825	11,085	17,349	254,703	12,473
<i>Percentage of Reduction</i>	-	68.2%	79.0%	67.2%	-381.9%	76.4%
<i>Correlative Transitions</i>	1,736	1,285	1,140	1,354	57,496	806
<i>Percentage of Reduction</i>	-	26.0%	34.3%	22.0%	-3212.0%	53.6%

We summarize the experimental results in Table 5.26 and Table 5.27. In the

reduction of self transitions, our scheme is inferior to BITS by 4.2% but superior to other three schemes. In the reduction of correlative transitions, our scheme is superior to all the other schemes.

Table 5.26 Averaged Self Transition Reduction of DCT-transformed Image Files

	<i>Unencoded</i>	<i>BI</i>	<i>BITS</i>	<i>hihrTS</i>	<i>EXODUS</i>	<i>BI-XNOR</i>
<i>Image 1</i>	54,518	16,378	10,542	16,872	254,254	12,368
<i>Image 2</i>	129,114	30,446	19,605	34,993	245,507	25,523
<i>Image 3</i>	52,857	16,825	11,085	17,349	254,703	12,473
<i>Average</i>	47,298	12,730	8,246	13,843	150,893	10,073
<i>Reduction</i>	-	73.1%	82.6%	70.7%	-219.0%	78.7%

Table 5.27 Averaged Correlative Transition Reduction of DCT-transformed Image Files

	<i>Unencoded</i>	<i>BI</i>	<i>BITS</i>	<i>hihrTS</i>	<i>EXODUS</i>	<i>BI-XNOR</i>
<i>Image 1</i>	1,542	1,122	1,044	1,203	60,451	693
<i>Image 2</i>	2,879	2,230	1,484	1,725	56,341	1,580
<i>Image 3</i>	1,736	1,285	1,140	1,354	57,496	806
<i>Average</i>	1,231	927	734	856	34,858	616
<i>Reduction</i>	-	24.7%	40.4%	30.5%	-2730.7%	50.0%

We find EXODUX encoding is very bad for the transmission of image files with DCT transformation. This is because the DCT transformation makes the appearance of zero become more frequent. When the data stream is from 00000000 to 00000000, all schemes except EXODUX will not cause any transition. EXODUX encoding will generate four self transitions for this kind of data stream.

5.3.6 Comparisons

The comparison of self transitions is shown in Table 5.28. The experimental result shows our scheme has the best performance in the types of random data, audios and gray image files. For the color image files, our scheme is inferior to BI by 1.4% but better than other three schemes. For DCT-transformed image files, our scheme is inferior to BITS by 4.2% but better than other three schemes.

Table 5.28 Performance Comparison of Self Transition Reduction

	<i>BI</i>	<i>BITS</i>	<i>hihrTS</i>	<i>EXODUS</i>	<i>BI-XNOR</i>
<i>Random Data</i>	27.3%	27.3%	0%	37.5%	44.0%
<i>Audio</i>	26.9%	42.0%	22.1%	29.0%	53.0%
<i>Gray Image</i>	18.7%	-8.8%	-39.9%	11.2%	21.0%
<i>Color Image</i>	16.3%	-20.7%	-52.9%	4.4%	15.2%
<i>Image with DCT</i>	73.1%	82.6%	70.7%	-219.0%	78.7%

The comparison of correlative transitions is shown in Table 5.29. The experimental result shows our scheme has the best performance in the types of audios and DCT-transformed image files. For the random data streams, gray image files and color image files, our scheme is inferior to EXODUS but better than other three schemes. But EXODUS is useless for the transmission of DCT-transformed image files.

Table 5.29 Performance Comparison of Correlative Transition Reduction

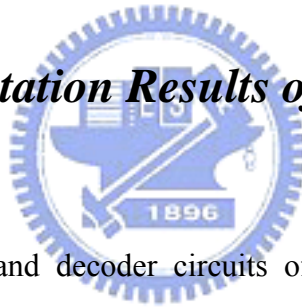
	<i>BI</i>	<i>BITS</i>	<i>hihrTS</i>	<i>EXODUS</i>	<i>BI-XNOR</i>
<i>Random Data</i>	54.5%	54.7%	0.4%	78.7%	74.4%
<i>Audio</i>	54.4%	32.5%	-18.2%	27.8%	68.3%
<i>Gray Image</i>	29.2%	43.9%	-26.9%	78.5%	70.8%
<i>Color Image</i>	23.6%	35.9%	-36.1%	78.0%	69.3%
<i>Image with DCT</i>	24.7%	40.4%	30.5%	-2730.7%	50.0%

Chapter 6

Implementation Results

In this chapter, we implement our structure and all mentioned schemes in VHDL, then compare the area and power consumption of the encoder/decoder. Next, we analyze the dynamic power consumption of each encoding scheme when the data patterns are transmitted.

6.1 Implementation Results of Encoder/Decoder



From the encoder and decoder circuits of BI, BITS, hihr-TS, EXODUS and BI-XNOR methods, we implement them in VHDL. Table 6.1 is the implementation results of encoder/decoder for each encoding scheme. The process is 0.18 μ m UMC technology. The clock frequency is set to be 100MHz.

Table 6.1 Implementation Results of Encoder/Decoder for Each Scheme

Scheme	Encoder		Decoder	
	Area (μ m ²)	Power Consumption (μ W)	Area (μ m ²)	Power Consumption (μ W)
BI	1680	80	213	40
BITS	1806	162	1427	106
hihrTS	1387	121	1257	93
EXODUS	1770	152	1417	97
BI-XNOR	4065	225	1670	152

6.2 Dynamic Power Estimation

From equation (2.3), the power dissipation resulted from bus transitions P_{bus} is $\gamma_s(\frac{1}{2}C_sV_{dd}^2) + \gamma_c(2C_cV_{dd}^2 + C_sV_{dd}^2)$ where γ_s is the number of the averaged self transitions per bus cycle and γ_c is the number of averaged correlative transitions per bus cycle. To estimate whole power consumption, we should also consider the power consumption of encoder/decoder for each encoding scheme. So we obtain the following total power dissipation:

$$P_t = P_{enc} + P_{dec} + \gamma_s(\frac{1}{2}C_sV_{dd}^2) + \gamma_c(2C_cV_{dd}^2 + C_sV_{dd}^2) \quad (6.1)$$

Since our VHDL simulation is based on the clock frequency of 100MHz, 100 mega data patterns are transmitted in one second. From the experimental results in chapter 5.3, we can estimate the value of γ_s and γ_c in the frequency of 100MHz. For general I/O buses, the driving voltage is 3.3V. Table 6.2 [6] shows the value of C_s and C_c in various IC processes. We assume the power consumption of each transition is independent and the length of bus wire is 3cm (standard LQFP 216pin). According to equation (6.1), we could estimate the dynamic power consumption for each encoding scheme when the data patterns are transmitted.

Table 6.2 Self Capacitance and Coupling Capacitance in Various IC Processes

Process	0.25 μ m		0.18 μ m		0.10 μ m		0.05 μ m	
	C_s	C_c	C_s	C_c	C_s	C_c	C_s	C_c
(10^{-18} F/ μ m)	10.96	40.76	10.5	44.75	9.97	49.35	15.75	46.82

In the following sections, we estimate the dynamic power consumption of the bus when different types of sample patterns are transmitted.

6.2.1 Random Data Streams

In the experiment in chapter 5.3.1 and the experimental result in Table 5.5, 100,000 random data patterns are transmitted. According to the same ratio, the self and correlative transitions are normalized and shown in Table 6.3 when the clock frequency is 100MHz. Table 6.4 is the power consumption of each encoding scheme.

Table 6.3 Bus Transitions of Random Data when The Frequency is 100MHz

<i>Encoding Scheme</i>	<i>Unencoded</i>	<i>BI</i>	<i>BITS</i>	<i>hihrTS</i>	<i>EXODUS</i>	<i>BI-XNOR</i>
<i>Self Transitions</i>	400,100,000	290,782,000	290,846,000	399,404,000	249,930,000	225,954,000
<i>Correlative Transitions</i>	87,870,000	39,955,000	39,833,000	87,501,000	18,683,000	22,459,000

Table 6.4 Power Consumption of Random Data for Each Scheme

	<i>Unencoded</i>	<i>BI</i>	<i>BITS</i>	<i>hihrTS</i>	<i>EXODUS</i>	<i>BI-XNOR</i>
<i>Power Consumption (μW)</i>	3,557	1,924	2,068	3,758	1,288	1,498
<i>Reduction</i>	-	45.9%	41.9%	-5.7%	63.8%	57.9%

6.2.2 Audio Files

In the experiment in chapter 5.3.3 and the experimental result in Table 5.11 and Table 5.12, 441000 points are transmitted. Since each point has 2 channels and each channel is 16 bits, 1764000 data patterns are transmitted. According to the same ratio, the self and correlative transitions are normalized and shown in Table 6.5 when the clock frequency is 100MHz. Table 6.6 is the power consumption of each encoding scheme.

Table 6.5 Bus Transitions of Audio Files when The Frequency is 100MHz

<i>Encoding Scheme</i>	<i>Unencoded</i>	<i>BI</i>	<i>BITS</i>	<i>hihrTS</i>	<i>EXODUS</i>	<i>BI-XNOR</i>
<i>Self Transitions</i>	394,190,306	288,262,925	228,778,175	307,268,934	279,956,179	185,089,739
<i>CorrelativeTransitions</i>	47,877,834	21,808,900	32,303,061	56,579,819	34,552,551	15,186,451

Table 6.6 Power Consumption of Audio Files for Each Scheme

	<i>Unencoded</i>	<i>BI</i>	<i>BITS</i>	<i>hihrTS</i>	<i>EXODUS</i>	<i>BI-XNOR</i>
<i>Power Consumption (μW)</i>	2,240	1,327	1,716	2,589	1,858	1,191
<i>Reduction</i>	-	40.8%	23.4%	-15.6%	17.1%	46.8%

6.2.3 Gray Image Files

In the experiment in chapter 5.3.3 and the experimental result in Table 5.16 and Table 5.17, 65536 data patterns are transmitted. According to the same ratio, the self and correlative transitions are normalized and shown in Table 6.7 when the clock frequency is 100MHz. Table 6.8 is the power consumption of each encoding scheme.

Table 6.7 Bus Transitions of Gray Image Files when The Frequency is 100MHz

<i>Encoding Scheme</i>	<i>Unencoded</i>	<i>BI</i>	<i>BITS</i>	<i>hihrTS</i>	<i>EXODUS</i>	<i>BI-XNOR</i>
<i>Self Transitions</i>	165,312,195	135,008,240	179,840,088	231,336,975	146,751,404	130,586,243
<i>CorrelativeTransitions</i>	42,077,637	29,801,941	23,625,183	53,385,925	9,060,669	12,271,118

Table 6.8 Power Consumption of Gray Image Files for Each Scheme

	<i>Unencoded</i>	<i>BI</i>	<i>BITS</i>	<i>hihrTS</i>	<i>EXODUS</i>	<i>BI-XNOR</i>
<i>Power Consumption (μW)</i>	1,658	1,325	1,348	2,355	797	1,002
<i>Reduction</i>	-	20.1%	18.7%	-42.0%	51.9%	39.6%

6.2.4 Color Image Files

In the experiment in chapter 5.3.4 and the experimental result in Table 5.21 and Table 5.22, 196608 data patterns are transmitted since each element has three parts – R, G and B. According to the same ratio, the self and correlative transitions are normalized and shown in Table 6.9 when the clock frequency is 100MHz. Table 6.10 is the power consumption of each encoding scheme.

Table 6.9 Bus Transitions of Color Image Files when The Frequency is 100MHz

<i>Encoding Scheme</i>	<i>Unencoded</i>	<i>BI</i>	<i>BITS</i>	<i>hihrTS</i>	<i>EXODUS</i>	<i>BI-XNOR</i>
<i>Self Transitions</i>	151,849,365	127,041,117	183,349,101	232,130,941	145,172,628	128,794,352
<i>Correlative Transitions</i>	38,668,315	29,555,257	24,773,661	52,637,227	8,519,999	11,882,019

Table 6.10 Power Consumption of Color Image Files for Each Scheme

	<i>Unencoded</i>	<i>BI</i>	<i>BITS</i>	<i>hihrTS</i>	<i>EXODUS</i>	<i>BI-XNOR</i>
<i>Power Consumption (μW)</i>	1,524	1,303	1,392	2,332	776	986
<i>Reduction</i>	-	14.5%	8.7%	-53.0%	49.1%	35.3%

6.2.5 DCT-transformed Image Files

In the experiment in chapter 5.3.5 and the experimental result in Table 5.26 and Table 5.27, 65536 data patterns are transmitted. According to the same ratio, the self and correlative transitions are normalized and shown in Table 6.7 when the clock frequency is 100MHz. Table 6.8 is the power consumption of each encoding scheme.

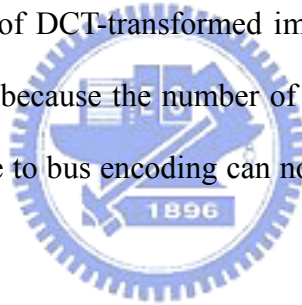
Table 6.11 Bus Transitions of DCT-transformed Image Files when The Frequency is 100MHz

<i>Encoding Scheme</i>	<i>Unencoded</i>	<i>BI</i>	<i>BITS</i>	<i>hhrTS</i>	<i>EXODUS</i>	<i>BI-XNOR</i>
<i>Self Transitions</i>	72,171,021	19,424,438	12,582,397	21,122,742	230,244,446	15,370,178
<i>CorrelativeTransitions</i>	1,878,357	1,414,490	1,119,995	1,306,152	53,189,087	939,941

Table 6.12 Power Consumption of DCT-transformed Image Files for Each Scheme

	<i>Unencoded</i>	<i>BI</i>	<i>BITS</i>	<i>hhrTS</i>	<i>EXODUS</i>	<i>BI-XNOR</i>
<i>Power Consumption (μW)</i>	185	200	326	293	2,382	434
<i>Reduction</i>	-	-8.1%	-76.2%	-58.4%	-1187.6%	-134.6%

In the transmission of DCT-transformed image files, all schemes are useless for power reduction. This is because the number of bus transition is not large enough, so that the power saving due to bus encoding can not compensate the power consumption of encoder and decoder.



Chapter 7

Application on Virtual Platform

In this chapter, we use CoWare Platform Architect to build an ARM-Based SoC virtual platform [7] and implement all encoding schemes. The study case is IEEE 802.16e baseband TX.

7.1 ESL Design Methodology



Electronic System Level (ESL) design is a front-end process for SoC design. It is a methodology to transform a verified C design to a SoC specification for realization. Fig.7.1 illustrates an ESL design methodology. In the software design part, it uses a virtual platform as part of the integrated design/development and debug environment (IDE). In the hardware register transfer level (RTL) design part, the SystemC golden model used for virtual platform can be reused in the component-level RTL design for functional co-verification. The virtual platform used for SW early development can be reused to co-verify with real platform after implementation.

Fig.7.2 shows the block diagram of our virtual platform. We use CoWare Platform Architect and its supported IP libraries (ARM926EJS_AHB_PSP, AMBA BL, Auxiliary, Peripherals) to build it. We add encoders/decoders before the data patterns are sent to or received from the bus. The bus width is 32 bit.

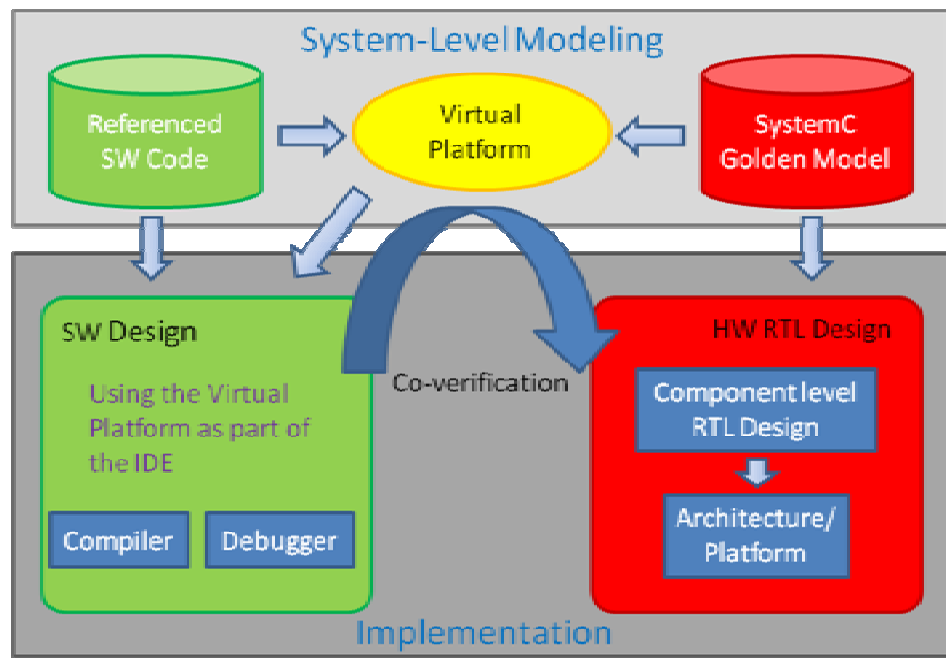


Fig.7.1 An ESL Design Methodology

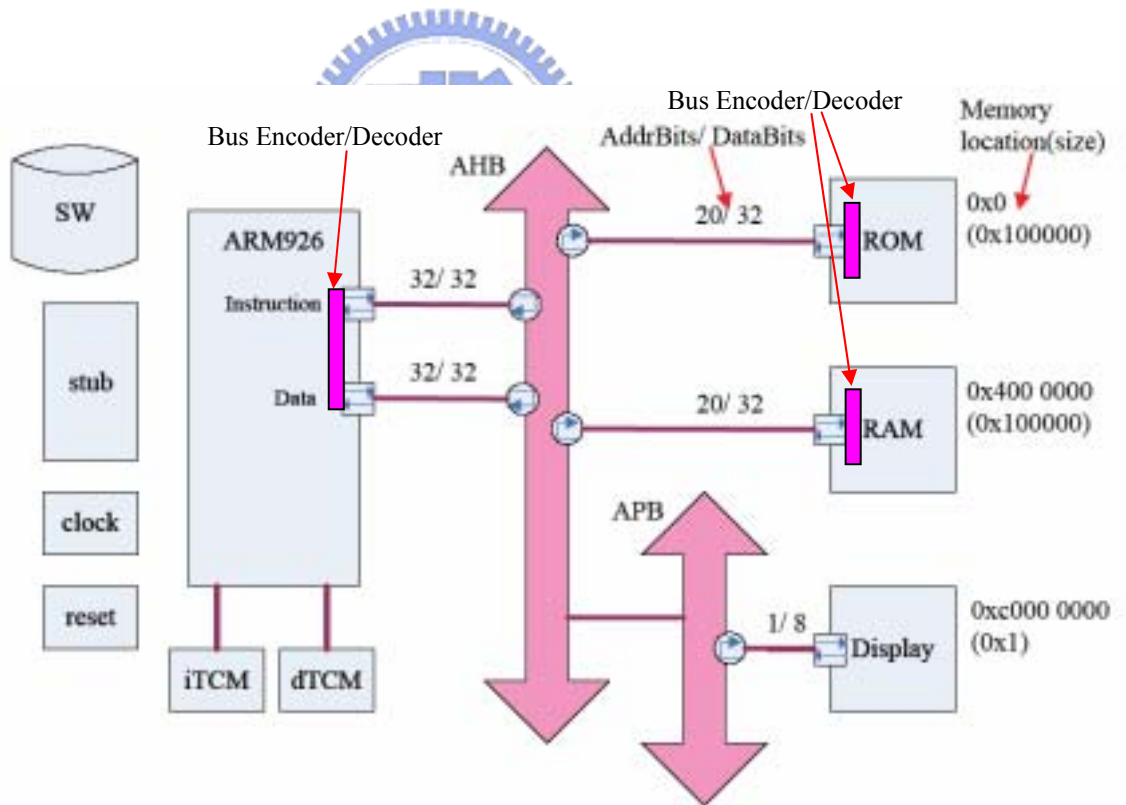


Fig.7.2 Block Diagram of Virtual Platform

7.2 Simulation Results

Table 7.1 shows the access times of memory and hardware model. Table 7.2 and Table 7.3 show the number of bus transitions when data is sent from ARM to the bus. Table 7.4 and Table 7.5 show the number of bus transitions when data is sent from memory to the bus. The results show that our encoding has better performances than other schemes except the correlative transitions in QPSK and 16QAM modulations when data is sent from ARM to the bus.

Table 7.1 Total Memory and HW Model Accesses

<i>Modulation Mode</i>		<i>QPSK</i>	<i>16QAM</i>	<i>64QAM</i>
<i>ROM Access Counts</i>	<i>Read</i>	314,888	589,902	885,465
	<i>Write</i>	0	0	0
	<i>Total</i>	314,888	589,902	885,465
<i>RAM Access Counts</i>	<i>Read</i>	16,956	30,804	44,653
	<i>Write</i>	14,212	26,552	38,893
	<i>Total</i>	31,168	57,356	83,546
<i>TX HW Access Counts</i>	<i>Read</i>	0	0	0
	<i>Write</i>	1,536	1,536	1,536
	<i>Total</i>	1,536	1,536	1,536

Table 7.2 Comparisons of Self Transitions (ARM Bus)

<i>Modulation Mode</i>	<i>QPSK</i>		<i>16QAM</i>		<i>64QAM</i>	
<i>Directly Sent</i>	107,960	-	189,060	-	280,870	-
<i>BI</i>	82,840	23.3%	147,376	22.0%	227,122	19.1%
<i>BITS</i>	57,232	47.0%	98,096	48.1%	151,772	46.0%
<i>hihrTS</i>	62,040	42.5%	113,032	40.2%	178,434	36.5%
<i>EXODUS</i>	184,020	-70.5%	354,568	-87.5%	520,334	-85.3%
<i>BI-XNOR</i>	49,432	54.2%	97,204	48.6%	123,130	56.2%

Table 7.3 Comparisons of Correlative Transitions (ARM → Bus)

<i>Modulation Mode</i>	QPSK		16QAM		64QAM	
<i>Directly Sent</i>	9,013	-	13,176	-	19,807	-
<i>BI</i>	2,775	69.2%	6,625	49.7%	13,894	29.9%
<i>BITS</i>	9,368	-3.9%	14,893	-13.0%	24,813	-25.3%
<i>hihrTS</i>	12,073	-34.0%	23,503	-78.4%	38,830	-96.0%
<i>EXODUS</i>	37,771	-319.1%	74,369	-464.4%	110,362	-457.2%
<i>BI-XNOR</i>	4,065	54.9%	8,487	35.6%	11,106	43.9%

Table 7.4 Comparisons of Self Transitions (Memory → Bus)

<i>Modulation Mode</i>	QPSK		16QAM		64QAM	
<i>Directly Sent</i>	4,497,635	-	8,333,525	-	12,410,989	-
<i>BI</i>	3,587,278	20.2%	6,601,678	20.8%	9,824,100	20.8%
<i>BITS</i>	3,048,295	32.2%	5,606,468	32.7%	8,365,826	32.6%
<i>hihrTS</i>	4,203,409	6.5%	7,713,958	7.4%	11,528,524	7.1%
<i>EXODUS</i>	3,803,103	15.4%	6,949,964	16.6%	10,364,592	16.5%
<i>BI-XNOR</i>	2,363,117	47.5%	4,320,903	48.2%	6,436,267	48.1%

Table 7.5 Comparisons of Correlative Transitions (Memory → Bus)

<i>Modulation Mode</i>	QPSK		16QAM		64QAM	
<i>Directly Sent</i>	476,927	-	885,842	-	1,318,390	-
<i>BI</i>	236,131	50.5%	434,836	50.9%	642,010	51.3%
<i>BITS</i>	514,361	-7.8%	946,754	-6.9%	1,395,713	-5.9%
<i>hihrTS</i>	1,180,592	-147.5%	2,162,730	-144.1%	3,195,046	-142.3%
<i>EXODUS</i>	554,250	-16.2%	1,004,553	-13.4%	1,433,197	-8.7%
<i>BI-XNOR</i>	179,671	62.3%	337,828	61.9%	489,984	62.8%

Chapter 8

Conclusions

In this thesis, a new bus encoding scheme for low power design has been presented that eliminates both self and correlative transitions. The key idea of our scheme is that we found different encoding operation has its superiority in specific conditions. So we analyze this characteristic and propose a new scheme which dynamically adapts to data patterns and applies the most suitable bus encoding operation. Though our discussion is based on 8-bit wide bus, the architectures with the multiple of 8-bit are also suitable. Just use two or more sets of the proposed encoding architecture.

However, the reduction of bus transitions does not mean it will surely cause power reduction. Because all bus encodings have overheads, the power consumption of encoder and decoder must be considered. From equation (6.1), the power consumption of encoder/decoder in certain operating frequency and process is fixed. We easily know when the length of bus line is longer or the number of bus transitions is larger, the efficiency of power reduction due to bus encoding will become better. The experimental result shows our encoding is suitable for the transmission of audio files.

The simulation on virtual platform shows our encoding has a better performance than other schemes except the correlative transitions in the modulation of QPSK and 16QAM when data is sent from ARM to the bus.

References

- [1] M-S Tsai, “**An Effective Amount-Driven Encoding/Decoding Method (ADEM) for Low-Power Data Bus with Coupling,**” M.S. thesis, Institute of Computer Science and Engineering, National Chiao Tung University, Hsin-Chu, Taiwan, 2006.
- [2] M. R. Stan and W. P. Burlison, “**Bus-invert coding for low-power I/O,**” IEEE Trans. VLSI Syst., vol. 3, pp. 49–58, Mar. 1995.
- [3] Youngsoo Shin, Kiyoungh Choi, and Young-Hoon Chang, “**Narrow Bus Encoding for Low-Power DSP Systems,**” VLSI Systems, Vol. 9, pp. 656-660, 2001.
- [4] K.-H. Baek, K.-W. Kim and S.-M. Kang, “**EXODUS Inter-module Bus-Encoding Scheme for System-on-a-chip,**” Electronics Letters, Vol. 36, pp. 615-617, 2000.
- [5] Chin-Tzung Cheng, Wei-Hau Chiao, Jean Jyh-Jiun Shann, Chung-Ping Chung, and Wen-Feng Chen, “**Low-Power BIBITS Encoding with Register Relabeling for Instruction Bus,**” VLSI Design, Automation and Test (VLSI-TSA-DAT), 2005 IEEE VLSI-TSA International Symposium, pages 41-44.
- [6] A. Elkammar, S. Vemuru and N. Scheinberg, “**A Bus Encoding Scheme To Reduce Power Consuming Signal Transitions**”, Proc. Int Conference on Embedded Systems and Applications, pp 12-17, 2004.
- [7] Sheng-Kuo Lu, “**ESL Design and Analysis in WiMAX Baseband Transmitter,**” M.S. thesis, Industrial Technology R&D Master Program on IC Design, National Chiao Tung University, Hsin-Chu, Taiwan, 2007.
- [8] S. P. Khatri, R. K. Brayton, A. L. Sangiovanni-Vincentelli, “**Cross-Talk Noise Immune VLSI Design: using Regular Layout Fabrics,**” Kluwer Academic Press, 2001.
- [9] A. Elkammar, S. Vemuru and N. Scheinberg, “**Bus Encoding Scheme to**

- Eliminate Unwanted Signal Transitions**”, Electronic Design, Test and Applications, 2006. DELTA 2006. Third IEEE International Workshop on 17-19 Page(s):6 pp. 2006.
- [10] Pouladi, A. and Nooshabadi, S., “**Opcode Encoding for Low Power Embedded Systems**,” Circuit and Systems, Vol. 5, pp. 5262-5265, 2005.
- [11] C-M Tsai, “**Low Power I/O Bus Encoding Methods**,” Ph.D. thesis, Institute of Computer Science and Engineering, Yuzn Ze University, Chung-Li, Taiwan, 2003.
- [12] P.Petrov, A.Orailoglu, “**Application-Specific Instruction Memory Customization for Power-Efficient Embedded Processors**,” IEEE Design and Test of Computers magazine, pp. 18-25, 2003.
- [13] Y-M Chen, “**Low-Power Instruction Bus Encoding for Embedded Systems**,” M.S. thesis, Institute of Computer Science and Information Engineering, National Chiao Tung University, Hsin-Chu, Taiwan, 2005.
- [14] Ghoneima, M., Ismail, Y., Khellah, M. and De, V., “**Reducing The Data Switching Activity of Serialized Datastreams**,” Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on 21-24 May 2006 Page(s):4 pp.
- [15] Ramprasad, S., Shanbhag, N.R., Hajj, I.N., “**A Coding Framework for Low-power Address and Data Busses**,” VLSI Systems, IEEE Trans., vol. 7, pp. 212–221, Jun. 1999.

About the Author

姓 名：蘇彥欽 Yen-Chin Su

出 生 地：高雄市

出生日期：1977. 8. 23

學 歷：

1983. 9 ~ 1989. 6 高雄市立愛國國民小學

1989. 9 ~ 1992. 6 高雄市立鼎金國民中學

1992. 9 ~ 1995. 6 高雄市立高雄高級中學

1996. 9 ~ 2000. 6 國立清華大學 電機工程學系 學士

2005. 2 ~ 2007. 8 國立交通大學 電機學院

IC 設計產業研發碩士班 碩士

經 歷：

2002. 5 ~ 2005. 2 創見資訊股份有限公司 研發工程師