

國立交通大學

電子工程學系 電子研究所碩士班

碩 士 論 文

針對晶片匯流排降低電感電容耦合效
應之編碼技術



On-chip Bus Encoding for Inductance and
Capacitance Crosstalk Reduction

研 究 生：黃俊盛

指 導 教 授：周景揚 博士

中 華 民 國 九 十 四 年 六 月

針對晶片匯流排降低電感電容耦合效應之編碼技術

On-chip Bus Encoding for Inductance and Capacitance
Crosstalk Reduction

研究生：黃俊盛

Student: Jiun-Sheng Huang

指導教授：周景揚 教授

Advisor: Jing-Yang Jou



A Thesis
Submitted to Institute of Electronics
College of Electronics Engineering and Computer Science
National Chiao Tung University
In partial Fulfillment of the Requirements
for the degree of
MASTER OF SCIENCE
in
Electronics Engineering

Hsinchu, Taiwan, Republic of China
June 2005

中華民國九十四年六月

針對晶片匯流排降低電感電容耦合效應之編碼技術

研究生：黃俊盛

指導教授：周景揚 博士

國立交通大學

電子工程學系 電子研究所碩士班



在深次微米的製程技術之下，電子元件的體積被持續的縮小，長導線的延遲將會成為決定晶片效能的重要因素。由於電感與電容的耦合效應在長導線上越來越顯著，訊號在一個時脈週期能夠傳輸的距離縮短了許多。為了減少耦合效應，目前為止學者專家已提出了許多種類的匯流排編碼技術以解決這個問題。然而，大部分的編碼技術只考慮了電阻與電容的效應，而電感的效應卻被忽略了。因此，我們在這篇論文中提出了一個有彈性的匯流排編碼技術，它可以明顯降低導線間電感與電容的耦合效應。我們還更進一步將這個編碼技術和曲線近似(curve fitting)結合，以達到在一個時脈週期內增加最大傳輸距離的功能。實驗結果顯示我們的編碼技術可以明顯的降低導線間電感和電容的耦合效應，並且進一步可以增加匯流排的最大傳輸距離。

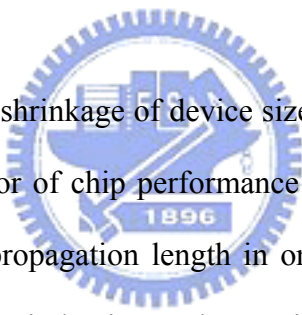
On-chip Bus Encoding for Inductance and Capacitance Crosstalk Reduction

Student : Jiun-Sheng Huang

Advisor : Dr. Jing-Yang Jou

Department of Electronics & Institute of Electronics
National Chiao Tung University

Abstraction

The logo of National Chiao Tung University is a circular emblem. It features a central shield with a book and a torch, surrounded by a gear-like border. The year '1896' is inscribed at the bottom of the shield.

With the continuous shrinkage of device sizes, the global interconnect delay becomes a dominant factor of chip performance in deep-submicron technology. Furthermore, the signal propagation length in one clock cycle could be greatly reduced due to the strong inductive and capacitive coupling effects on global interconnects. In order to reduce the coupling effects, many bus encoding methods have been proposed. However, most of them consider only resistance and capacitance effects. In this thesis, we propose a flexible bus encoding flow that can **reduce the inductive and capacitive coupling delay on on-chip bus**. In addition, combining with a curve fitting method, our encoding scheme can be utilized to **increase the maximum propagation length in one clock cycle**. Simulation results show that our flow can significantly reduce the inductance and capacitance coupling delay and, hence, increase the maximum propagation length of the given bus structure.

Acknowledgments

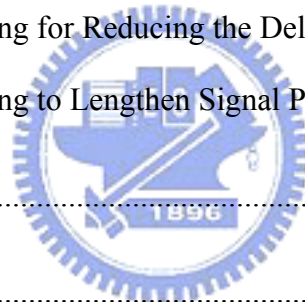
I would like to express my sincere gratitude to my advisor, Dr. Jing-Yang Jou, for his insightful suggestion and patient didactics throughout the course and this work in the two years of my graduate life. It is also fortune for me to stay with such great and kind members in the EDA group. I am also indebted to Shang-Wei Tu, for his patient guidance and constructive suggestions on my work. Finally, special thanks go to my girlfriend 陳佩弦, my parents, 黃國欽 and 吳惠聆, and my sister, 黃唯婷, for their limitless encouragement and love.



Contents

摘要	i
Abstraction	ii
Acknowledgments	iii
Contents	iv
List of Tables	vi
List of Figures	vii
Chapter 1 Introduction	1
1.1 Related Works on Bus Encoding	9
1.2 Our Approach	12
Chapter 2 Bus Encoding for Reducing the Delay Time	15
2.1 Preliminary	15
2.1.1 Assumption and Problem Input	15
2.1.2 The Bus Structure	16
2.2 Overall Encoding Flow	18
2.3 Build the <i>RLC</i> Model	21
2.4 SPICE Simulation for <i>Basis Vectors</i>	22
2.5 Build the Transition Graph	25
2.6 Find a Maximum Valid Code Set	29
2.7 Reduce Simulation Time by Using Superposition Theorem	32
2.8 Simulation Results of Delay Reduction	35

2.9	Delay Reduction under Different Working Conditions.....	37
2.10	The Performance Comparison between Shield Insertion Technique and Our Encoding Method	44
Chapter 3	Bus Encoding to Lengthen Signal Propagation.....	47
3.1	Problem Formulation.....	49
3.2	Maximum Signal Propagation Length Estimation Flow	49
3.3	The Curve Fitting Method	51
3.4	The Termination Condition	53
3.5	Simulation Results.....	56
Chapter 4	Conclusions	60
4.1	Bus Encoding for Reducing the Delay Time.....	60
4.2	Bus Encoding to Lengthen Signal Propagation.....	61
Reference	62
Vita	V-1



List of Tables

Table 1: The simulation results of 5-bit bus considering RLC effects when inductive coupling effects are dominant [4].	8
Table 2: The simulation results of 5-bit bus considering <i>RC</i> effects [4].	8
Table 3: The reduction (%) of LESS and bus invert method [11].	11
Table 4: The transition delays of a 2-bit bus without encoding.	20
Table 5: The transition delays of the generated valid code set.	20
Table 6: The run time improvement using our method.	34
Table 7: The wire overheads versus different delay constraints for a 4-bit data bus.	39
Table 8: The encoding results for different working frequencies for a 3-bit bus.	41
Table 9: The encoding results for different working frequencies for a 6-bit bus.	42
Table 10: The encoding results for different working frequencies for a 8-bit bus.	42
Table 11: The encoding results for different working frequencies for a 9-bit bus.	43

List of Figures

Figure 1: Delay for gate and global wire versus feature size [1].	2
Figure 2: Coupling and Area Capacitance of interconnects (a) in deep submicron technology and (b) in older silicon technology.	3
Figure 3: Interconnect delay for various coupling factors (fcc) [3].	5
Figure 4: An example of the worst-case switching pattern that incurs the largest delay while considering only inductance coupling effects.	7
Figure 5: The basic bus encoding architecture.	9
Figure 6: Examples of capacitive coupling between adjacent wires [11].	10
Figure 7: Slim encoder circuit block diagram of the LESS [11].	11
Figure 8: Delay function of signal wires [12].	12
Figure 9: The coplanar bus structure.	16
Figure 10: The overall bus structure including the encoder and decoder.	17
Figure 11: The overall encoding flow.	18
Figure 12: The encoded 3-bit bus with the generated valid code set.	20
Figure 13: Extract <i>RLC</i> and generate the corresponding SPICE file.	21
Figure 14: HSPICE simulations with the <i>minimum basis vectors</i> of the <i>minimum basis vector set</i>	24
Figure 15: An example of superposition.	26

Figure 16: The voltage waveforms of the signal wires obtained (a) by directly conducting HSPICE simulation and (b) by using superposition method.	27
Figure 17: The transition graph.	28
Figure 18: Our greedy algorithm to find a maximum clique.	30
Figure 19: The greedy search result of Figure 17.....	31
Figure 20: The number of extended clique size by using the second loop in Figure 18.....	32
Figure 21: The run time of our method and pure HSPICE simulation.....	33
Figure 22: Transition delay of the bus (6 bit) before encoding.	36
Figure 23: Transition delay of the bus (7 bit) after encoding.	36
Table 12: The encoding results for different wire dimensions for a 6-bit bus.....	44
Figure 24: The bus structure (a) with the use of our encoding method and (b) with the use of shield insertion technique.	45
Figure 25: The worst-case transition delay by using our method and the shield insertion technique.	46
Figure 26: Trends for the clock locality metric [17].	48
Figure 26: The maximum signal propagation length estimation flow.....	49
Figure 27: Curve fitting with different functions.	52
Figure 28: The decision policy of our flow.	53

Figure 29: (a) The distance difference $\Delta D (|invalid\ length - valid\ length|)$ is larger than user-required precision $\Delta D_{required}$. (b) $\Delta D < \Delta D_{required}$55

Figure 30: The maximum propagation length vs. different wire overheads for a 4-bit data bus.57

Figure 31: The maximum propagation length vs. different wire overheads for a 6-bit data bus.58

Figure 32: The maximum propagation length after adding one more bit as wire overhead.59



Chapter 1

Introduction



The trend of shrinking feature size implies shorter gate lengths, smaller interconnect pitch (more congested interconnect), lower device threshold voltage and supply voltage, etc. With aggressive scaling of transistor gate lengths, the gate delay has been dramatically reduced. In addition, the use of copper and low-k materials lessen the impact on scaling of the wiring delay [1]. However, the benefit of material changes alone is insufficient to reduce the global interconnect delay and meet the overall performance requirement. As shown in **Figure 1** [1], the global interconnect delay becomes more important comparing with the gate delay as the technology advances. Therefore, the global interconnect delay becomes a crucial issue for overall performance optimization in today's high performance circuit designs.

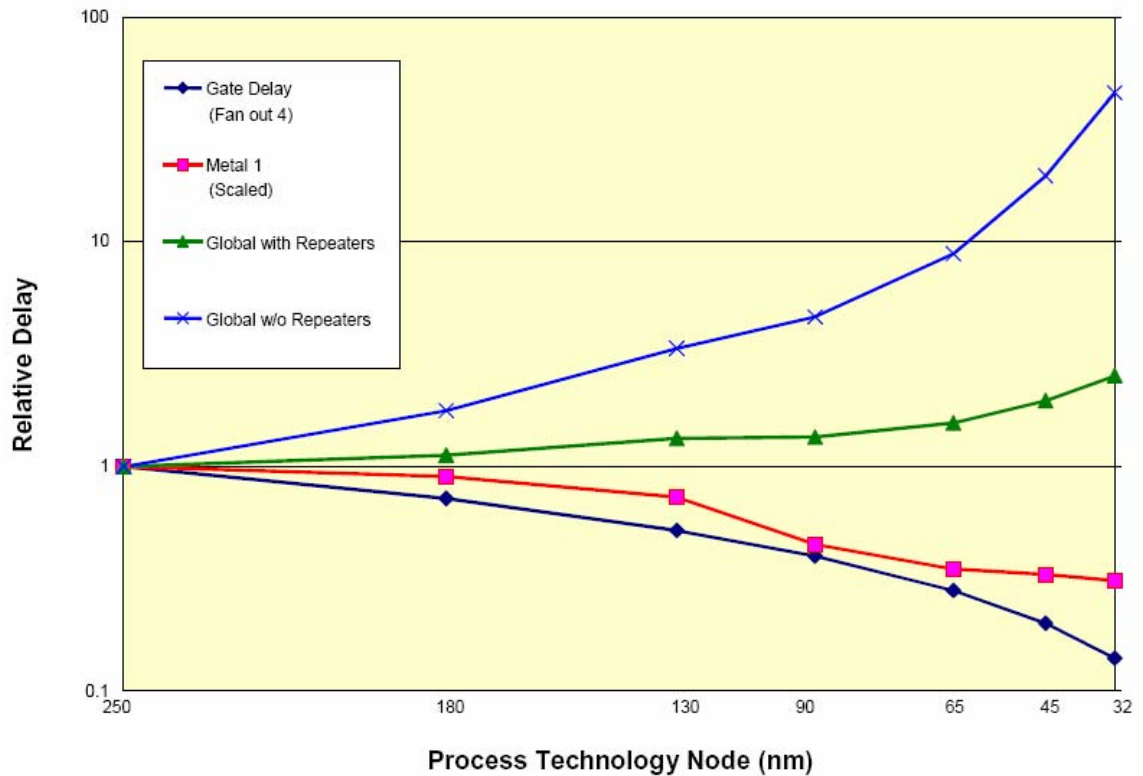
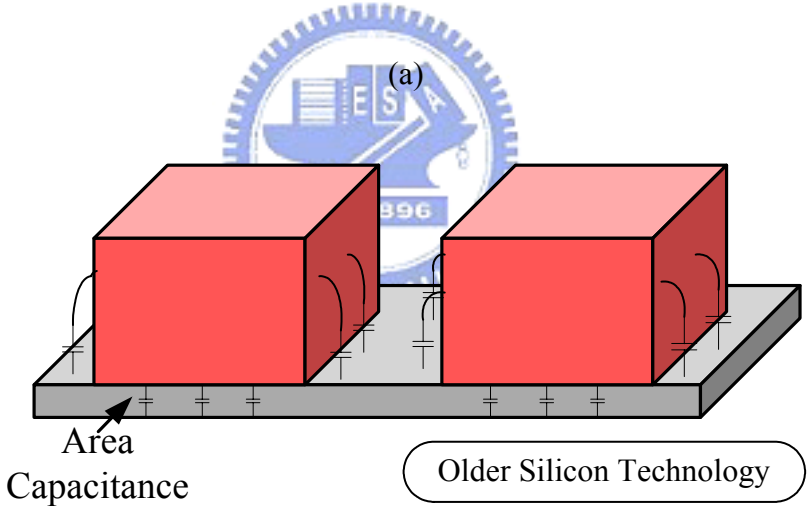
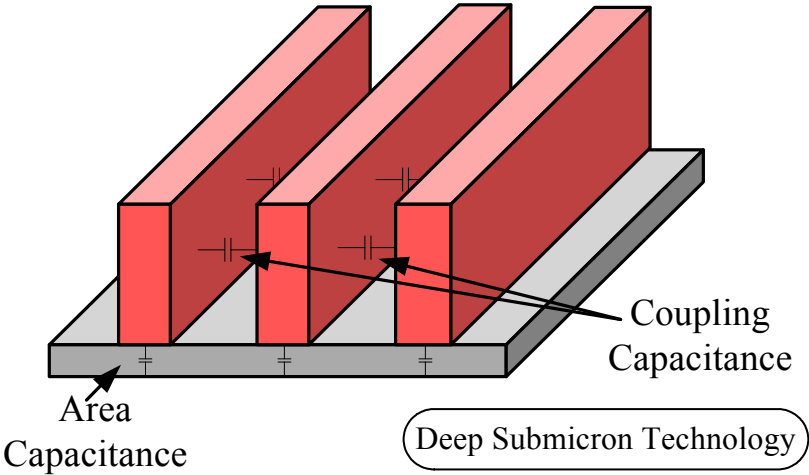


Figure 1: Delay for gate and global wire versus feature size [1].

As the supply voltage being reduced and the number of nets continues increasing, the coupling effects have become an important issue on the global interconnect due to the lower noise margin and congested wires. Coupling effects can be caused by the coupling capacitance and inductance between victims and neighboring wires. Capacitive coupling effects refer to the charge injected in quiet nets (victims) by the switching on neighboring nets (aggressors) through the capacitance between them (coupling capacitance). **Figure 2** illustrates the difference of the coupling and area capacitance between the deep submicron technology (DSM) and the older technologies [2]. In the DSM technology, wires are taller and more congested than before. Due to the high aspect ratio and small pitch of interconnects in DSM, the coupling capacitance dominates total wire

capacitance. Thus, for the congested interconnects, the interconnect delay could strongly depend on whether the aggressor signals are switching in the same or the opposing direction to the victim net.



(b)

Figure 2: Coupling and Area Capacitance of interconnects (a) in deep submicron technology and (b) in older silicon technology.

In the following, we give some theoretical explanations of capacitive coupling effects on the interconnect delay. While considering RC effects of interconnects, the wire delay has linear dependency on the time constant $\tau =$

RC_{eff} , where R is the total wire resistance and C_{eff} is the total effective wire capacitance. C_{eff} can be represented by the following equation:

$$C_{eff} = C_0 + C_c \left| \frac{\Delta V_{i-1} - \Delta V_i}{V_{dd}} \right| + C_c \left| \frac{\Delta V_i - \Delta V_{i+1}}{V_{dd}} \right|, \quad (1)$$

where C_0 is the capacitance between the wire and the substrate, C_c is the capacitance between adjacent wires, ΔV_i is the voltage variation of a wire i , and V_{dd} is the supply voltage. The delay of the wire i is the largest (i.e., C_{eff} is maximum) when the adjacent wires $i-1$ and $i+1$ transit inversely with respect to the wire i (both the values of $|\Delta V_{i-1} - \Delta V_i|$ and $|\Delta V_i - \Delta V_{i+1}|$ reach the maximum). Hence, if we only consider RC effects, the worst-case switching pattern that incurs the largest delay is when adjacent wires simultaneously switch in the opposite transition direction of a victim wire. A simulation results in [3] show the effects on the transition delay when the neighboring wires have different transitional directions. In this figure, $fcc = 2$ implies that neighbors make opposite transitions, $fcc = 0$ implies that they are switching in the same direction, while $fcc = 1$ implies that they are static. We can observe that the simulation result in **Figure 3** complies with our theoretical explanations when considering RC effects of interconnects. Therefore, if the neighboring wires transit inversely in the direction of the victim wire, the victim wire will have the largest transition delay (worst-case switching delay).

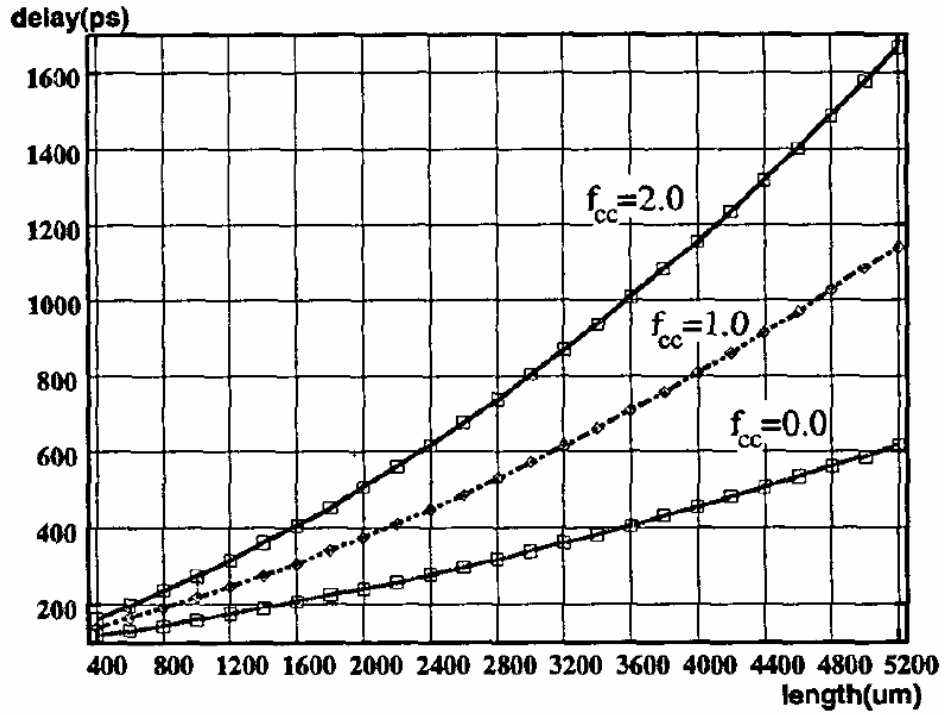


Figure 3: Interconnect delay for various coupling factors (fcc) [3].

Inductive coupling effects occur when a switching signal causes a current loop and changes the magnetic field in the nearby space. We give an example in **Figure 4** [4] to explain the long range inductive coupling effect. From *Faraday's Law*, as shown in Equation (2), the electromotive force induced in a closed circuit is equal to the negative rate of increase of the magnetic flux linking the circuit. We have

$$V_j = - \frac{d\Phi_{ij}}{dt} \text{ with } \Phi_{ij} = \int_{S_j} \vec{B}_i \cdot d\vec{s}_j, \quad (2)$$

where V_j is the electromotive force induced in a circuit loop j due to the time-varying current I_i in a circuit loop i . Here, Φ_{ij} is the magnetic flux in the loop j due to the current I_i in the loop i , \vec{B}_i is the magnetic flux density arising

from the current I_i in the loop i , and S_j represents the surface bounded by the loop j . The orientation of \vec{B}_i can be determined from the *right-hand rule*: if the thumb of the right hand points at the current direction, the other fingers point at the direction of the magnetic field. Therefore, as shown in **Figure 4**, the time-varying (increasing) current of the leftmost aggressor wire induces a downward time-varying (increasing) magnetic field on the victim wire. For simplicity, the aggressor and the victim loops are also shown in **Figure 4** (dotted loops). Therefore, the mutual flux Φ is positive and also increases with time because of the same direction of the resulting magnetic flux density \vec{B} and the victim loop \vec{s} ; in other words, $\vec{B} \cdot d\vec{s}$ is also positive. In conclusion, from Equation (2), the induced voltage on victim loop is negative; that is, the induced current on the victim wire flows in the reverse direction of the original victim current. Hence, while all neighboring wires simultaneously switch in the same direction as the victim wire does, they all induce a reverse current on the victim wire as shown in **Figure 4**. As a result, the charge current of the victim wire is reduced. This implies that the charging time (delay) increases due to the long-range coupling. We can conclude that as the inductance effects on wires become more significant than the capacitance effects, the worst-case switching pattern with the maximum delay is when all wires simultaneously switch in the same direction. Meanwhile, these patterns also result in the largest voltage over/under swing on wires.

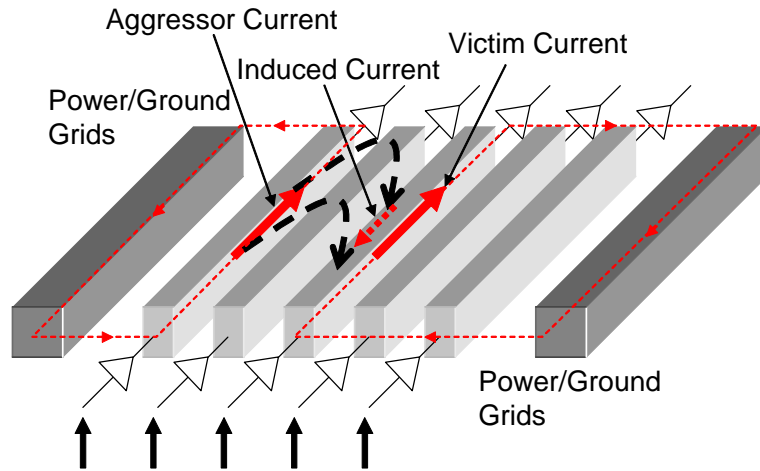


Figure 4: An example of the worst-case switching pattern that incurs the largest delay while considering only inductance coupling effects.

In the following, some simulations are conducted to verify our theoretical explanations. In **Table 1**, we show the complete simulation results for all switching patterns on the 5-bit bus structure considering the *RLC* effects when inductive coupling effects are dominant [4]. From **Table 1**, the worst-case switching pattern that causes the largest transition delay of the center wire is $\uparrow\uparrow\uparrow\uparrow\uparrow$. However, the authors also conduct simulations on the same bus structure when considering only *RC* effects [4]. The simulation results are listed in **Table 2**. The worst-case switching pattern of the center wire when considering only *RC* effects is $\downarrow\downarrow\uparrow\downarrow\downarrow$. From **Table 1** and **Table 2**, we find out that, for the center wire, the worst-case switching pattern changes from $\uparrow\uparrow\uparrow\uparrow\uparrow$ (in **Table 1**) to $\downarrow\downarrow\uparrow\downarrow\downarrow$ (in **Table 2**) and the best-case switching pattern changes from $\downarrow\downarrow\uparrow\downarrow\downarrow$ (in **Table 1**) to $\uparrow\uparrow\uparrow\uparrow\uparrow$ (in **Table 2**). Thus, the worst-case and best-case switching patterns are completely different when considering *RC* and *RLC* effects. In conclusion, the inductive effects can not be neglected especially for high performance circuit designs. Otherwise, designers might gain no improvement on the global

interconnect delay when doing timing optimization due to strong inductive coupling effects.

Table 1: The simulation results of 5-bit bus considering RLC effects when inductive coupling effects are dominant [4].

Switching pattern	50% delay of the central wire (ps)	delay comparison with that of 00↑00	max swing (V)	noise (% of Vdd)
↑↑↑↑↑	97	51.56%	1.71	42.50%
↑↓↑↑↑	88	37.50%	1.47	22.50%
↑↓↑↑↓	63	-1.56%	1.31	9.20%
↑↓↑↓↑	75	17.19%	1.17	-2.50%
↑↓↑↓↓	51	-20.31%	1.04	-13.33%
↓↑↑↑↑	78	21.88%	1.56	30.00%
↓↑↑↑↓	55	-14.06%	1.4	16.67%
↓↑↑↓↓	45	-29.69%	1.13	-5.83%
↓↓↑↑↑	66	3.13%	1.33	10.83%
↓↓↑↓↓	37	-42.19%	0.903	-24.75%
00↑00	64	0.00%	1.31	9.17%

(Supply voltage = 1.2V) (0: no switch)

Table 2: The simulation results of 5-bit bus considering RC effects [4].

switching pattern	50% delay of the central wire (ps)	delay comparison with that of 00↑00
↑↑↑↑↑	33	-35.29%
↑↓↑↑↑	49	-3.92%
↑↓↑↑↓	51	0.00%
↑↓↑↓↑	73	43.14%
↑↓↑↓↓	75	47.06%
↓↑↑↑↑	35	-31.37%
↓↑↑↑↓	37	-27.45%
↓↑↑↓↓	54	5.88%
↓↓↑↑↑	52	1.96%
↓↓↑↓↓	76	49.02%
00↑00	51	0.00%

(Supply voltage = 1.2V)

Recently, many methods have been proposed to reduce the coupling effects such as the shield insertion [5, 6, 7], the wire sizing [6, 8], the wire spacing, the buffer insertion [6, 9], the net reordering [7], and the bus encoding [10, 11, 12]. All of the above methods require some area and power overhead to reduce crosstalk. In this thesis, we focus on improving the bus encoding techniques to reduce the *LC* coupling effects and enhance the performance of buses. The basic idea is that by adding some logic at the transmitter and receiver, the highly coupled data pattern can be transformed to the less coupled codeword pattern. **Figure 5** demonstrates the basic bus encoding architecture. Some relative works on bus encoding for reducing delay will be introduced in the following section.

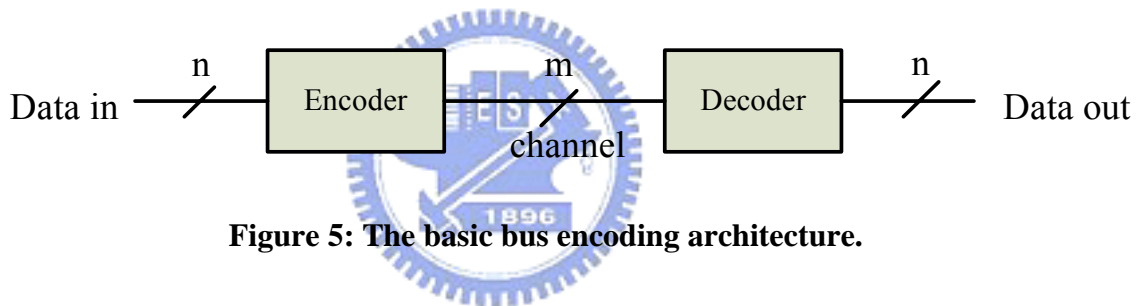


Figure 5: The basic bus encoding architecture.

1.1 Related Works on Bus Encoding

Victor and Keutzer propose some data encoding techniques to eliminate crosstalk delay due to the capacitive coupling effects [10]. Their basic idea is transfer the data patterns to “self-shielding” codes and transmit them on bus. Self-shielding code means that adjacent signal wires will not transit in the opposite directions. They classify their encoding technique as memory and memoryless code. Memory code states that the transmitter and receiver have many different codebooks. When different codewords are transmitted on the

channel, there are different codebooks at the receiver which decide the mapping between codewords and data patterns. On the other side, memoryless code is that there is only one codebook on both the transmitter and the receiver. Hence, there is only one mapping between codewords and data patterns. The advantage of the memoryless code is that this codebook can be implemented by some simple logic gates. On the other hand, the advantage of memory code is that this method has more than one mapping ways and will need fewer extra bus bits to build the self-shielding codes comparing with memoryless code. However, the overhead of memory code is that both transmitter and receiver need a ROM (or some memory unit) to store the large number of codebooks.

Baek et al. propose a bus encoding scheme called LESS (Low Energy Set Scheme) [11] to minimize the coupling effects which result in the significant delay and power consumption on the on-chip interconnect. Their goal is removing the type C transitions and minimizing the number of transitions at the same time. The three types of transitions are shown in **Figure 6**. It is obvious that by considering the coupling delay and power due to the capacitive coupling effects, the worst-case among three examples is type C and the best one is type A.

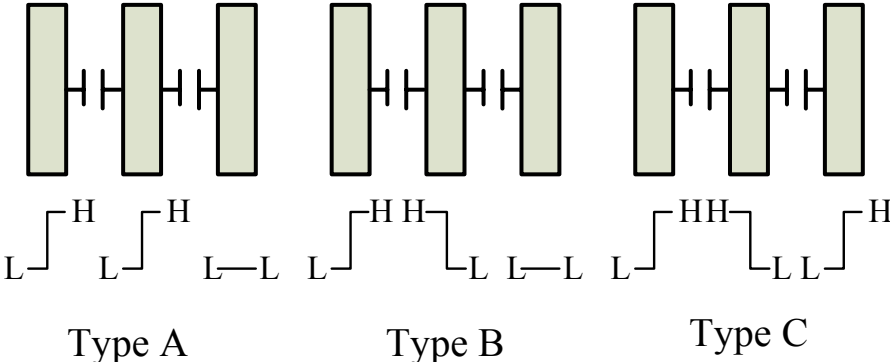


Figure 6: Examples of capacitive coupling between adjacent wires [11].

They use XOR-XNOR or XNOR-XOR to transmit the next cycle data sets which generate lower power-delay product. The block diagram of transmitter is shown in **Figure 7**. Let $y_i(n)$ denotes the encoded pattern which is currently on a bus at cycle time n , and $x_i(n+1)$ represents the data pattern which will be transmitted at the next cycle. Thus, we have four types of patterns (XON type, XNO type, all zero type, and all one type) to choose at SEL. The decision is made by a slim controller. The experimental result in **Table 3** shows that the LESS scheme can significantly reduce the power and the delay comparing with the bus invert method.

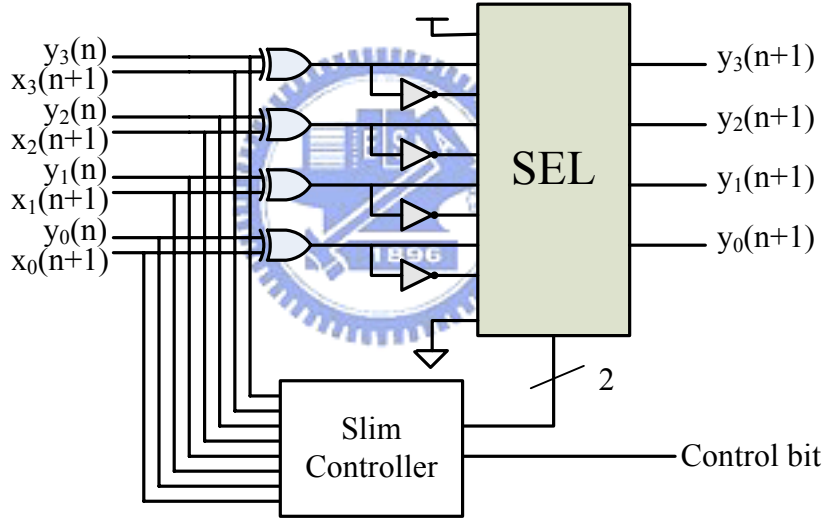


Figure 7: Slim encoder circuit block diagram of the LESS [11].

Table 3: The reduction (%) of LESS and bus invert method [11].

Item	Bus-invert	LESS
Power	13.1%	15.6%
Worst-case Delay	0%	31.4%
Energy	13.1%	42.1%

Sotiriadis and Chandrakasan [12] present a method to calculate delay. By using this delay model, they propose a technique to speed up the communication through a data bus using coding. They use Elmore delay method to calculate the delay function of transitions. **Figure 8** demonstrates the delay function of signal wires. For an m -bit bus, Δ_k is the change of the voltage of the k -th wire ($\Delta_k = \{-1, 0, 1\}$), C_L is the total capacitance between a wire and the ground, r_T is the total resistance of the signal wire, and λ is a normalized factor. By using the delay function, they can find out what transition patterns result in long delay. By avoiding these transitions, they can reduce the bus delay and achieve the goal.

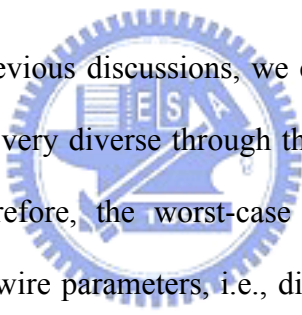
$$\frac{T_k(\mathbf{u}^0, \mathbf{u}^N)}{r_T * C_L} = \begin{cases} (1 + \lambda)\Delta_1^2 - \lambda\Delta_1\Delta_2 & k = 1 \\ (1 + 2\lambda)\Delta_k^2 - \lambda\Delta_k(\Delta_{k-1} + \Delta_{k+1}) & 1 < k < m \\ (1 + \lambda)\Delta_m^2 - \lambda\Delta_m\Delta_{m-1} & k = m \end{cases}$$

Figure 8: Delay function of signal wires [12].

1.2 Our Approach

As the process technology advances and the clock frequency increases over GHz, the inductance effects on on-chip interconnect structures have become increasingly significant [1]. Most existing works focus on reducing the effect resulting from the coupling capacitance on the bus structure. There is not much work in the literature considering the inductance effects on the bus structure to

develop encoding schemes to reduce the bus delay. However, considering the *RLC* circuit model for the bus structure, we find out that when the inductance effect dominates, the worst-case switching pattern with the largest on-chip bus delay is when all wires simultaneously switch in the same direction [4]. Furthermore, in [4], the authors indicate that while considering the *RLC* effect of interconnects, the worst-case switching pattern will change under different levels of interconnect (local, medium, or global wire) and different working frequency. Hence, as inductance cannot be neglected in today high-performance circuit design, it is very important to consider the *RLC* effect while developing the bus encoding schemes.



From [4] and the previous discussions, we can understand that the impacts caused by aggressors are very diverse through the mixture of the capacitive and inductive coupling. Therefore, the worst-case delay patterns could be very different for various bus wire parameters, i.e., different inductive and capacitive coupling conditions. Moreover, the capacitive and inductive coupling effects vary with different working frequencies since the impedance of capacitance $(j\omega C)^{-1}$ (where $\omega = 2\pi f$) decreases with the frequency but the impedance of inductance $j\omega L$ increases. Therefore, when considering *RLC* effects, it is crucial to take design parameters into account to derive a better bus encoding scheme.

Thus, with the concept that the worst-case switching pattern varies with given design parameters while consider the *RLC* effect of interconnects, we propose a flexible encoding scheme for on-chip buses with given parameters. The key idea is that the coupling effect should be alleviated by transforming the data

sequences transmitting through on-chip buses. However, the architectures of the encoder and decoder should be of low complexity so that the power and delay overheads due to the codec circuitry can be compensated by the significant reduction of the bus delay.

The thesis is organized as follows. Chapter 2 describes our bus encoding flow for reducing the delay time. Further more, we propose a bus encoding flow to lengthen signal propagation in chapter 3. Finally, chapter 4 concludes this thesis.



Chapter 2

Bus Encoding for Reducing the Delay Time



2.1 Preliminary

2.1.1 Assumption and Problem Input

In this thesis, we consider the coplanar bus structure shown in **Figure 9** to build the encoding flow. In this bus structure, we assume that each driver (receiver) has a uniform size and each signal wire has a uniform width, pitch, length and height. Given the parameters of wires (width, height and pitch), the delay constraint, the working frequency (or slew rate) and the number of data bits,

the encoding flow will generate a valid code set that meet the delay constraint with considering the *LC* coupling effects. The valid code set states that any transition between patterns within this code set is guaranteed to meet the delay constraint. This code set can be used for one-to-one mapping with data patterns.

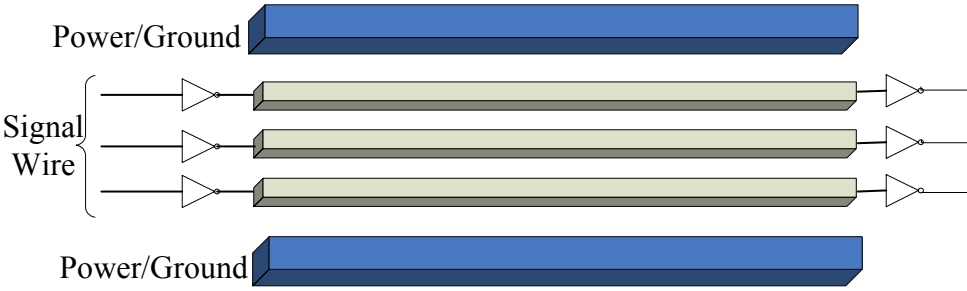


Figure 9: The coplanar bus structure.

2.1.2 The Bus Structure

Assume the number of data bit is n , the overall bus structure is shown in **Figure 10**. The valid code set of the global bus contains only 2^n out of 2^m possible codes. The specific 2^n codes are selected to minimize the coupling effects between any two of them. In addition, the transition delay between any two patterns in the specific 2^n codes will meet the delay constraint which is given by user. The encoding and decoding process are straightforward and are not discussed in this thesis.

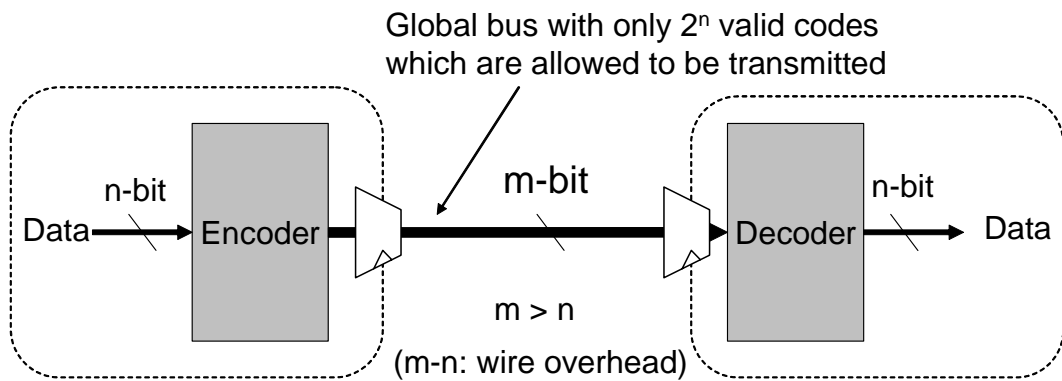


Figure 10: The overall bus structure including the encoder and decoder.

Since transistors mainly operate in the linear region during transitions, it is assumed that all drivers' output resistances are linear throughout the simulations. Therefore, the drivers will be modeled as simple linear resistances. In addition, the receivers will be replaced by equivalent gate capacitances in the circuit model and the wires are replaced by equivalent *RLC* circuit models. With the models, the built circuit model for the coplanar bus structure will be constructed by only linear elements (linear *R*, *L*, and *C*). In other words, the built circuit is a *LTI* (linear time invariant) system.

In the simulation, we assume that synchronous latches are located at the transmit side and receive side. Hence, all the signals switch at the same time on the buses, which is a very common assumption for busses [13].

2.2 Overall Encoding Flow

Figure 11 illustrates our overall bus encoding flow. With the given parameters, the first step is to build the 3D bus structure and then extract the resistances, capacitances, and inductances of bus wires. After extraction, the equivalent RLC circuit will be built. Next, the built circuit will be simulated using HSPICE with the *basis vectors* which will be defined later. By applying superposition theorem [14], we can establish the transition graph efficiently.

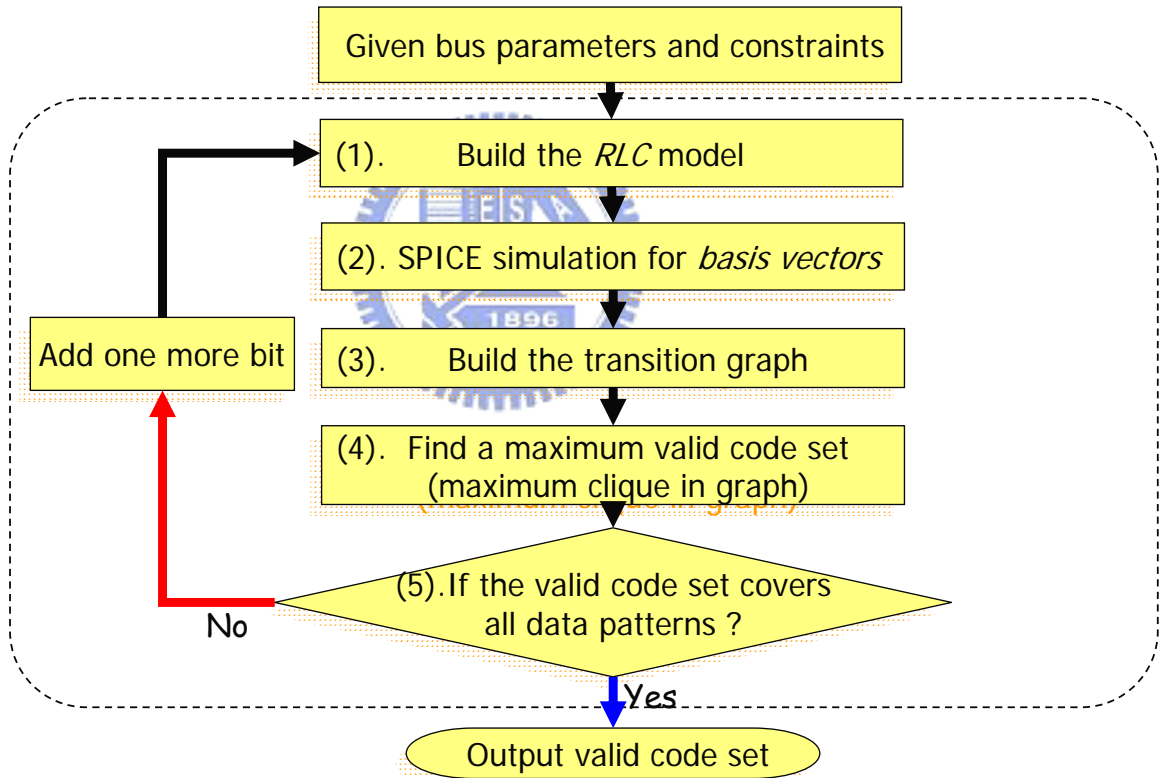


Figure 11: The overall encoding flow.

With the information of the transition graph, the greedy algorithm is applied to find a valid code set in which all transitions between any code pair will meet the delay constraint. In step (5), we will check whether the code set covers all

data patterns or not. If so, the valid code set will be outputted to map to the data patterns. Otherwise, we will add one more bit line to the bus structure and redo from step (1). In the following, we give an example to demonstrate the delay reduction by using our encoding flow with the generated valid code set. Given a 2-bit bus structure (4 data patterns 00, 01, 10, 11), and the delay constraint is set to 30 picoseconds. The transition delays of all transition patterns of the 2-bit bus are listed in **Table 4**. In **Table 4**, since the transition pattern $\uparrow\downarrow$ (i.e. $01 \rightarrow 10$ or $10 \rightarrow 01$) violate the delay constraint, we will use our encoding method to improve the worst-case switching delay to meet the delay constraint for the bus. By using our encoding flow, a valid code set which contains 4 patterns 000, 001, 100, and 101 will be generated. The one to one mapping between the data patterns and the valid codes and the new bus structure with the generated valid code set are shown in **Figure 12**. By using our encoding technique, the worst-case switching pattern $\uparrow\downarrow$ is removed from the bus since only the valid codes (i.e. 000, 001, 100, and 101) are allowed to be transmitted on the bus. The transition delays of all transition patterns of the encoded 3-bit bus are listed in **Table 5**. From **Table 5**, we can observe that all transitions between any two codes in the valid code set meet the delay constraint.

Table 4: The transition delays of a 2-bit bus without encoding.

Transition patterns	Transition delay (picoseconds)
—↑	27.20840
↑—	27.19817
↑↓	38.72782
↑↑	11.96549

(— : stable wire, ↑: switch from low to high, ↓: switch from high to low)

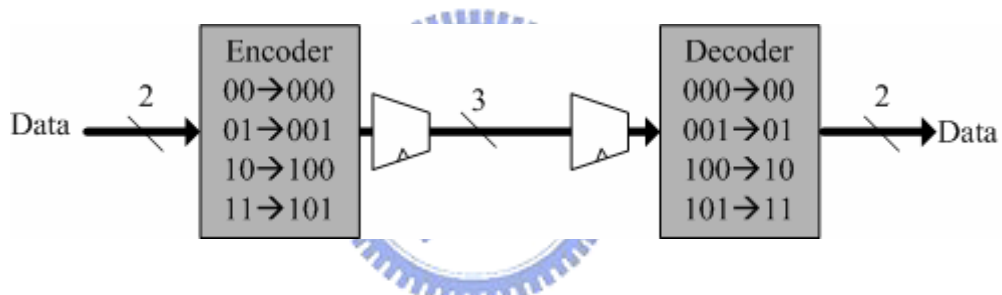


Figure 12: The encoded 3-bit bus with the generated valid code set.

Table 5: The transition delays of the generated valid code set.

Transition patterns	Transition delay (picoseconds)
— — ↑	24.6111
↑ — —	24.5803
↑ — ↓	29.2374
↑ — ↑	19.6909

The details of each step in our encoding flow will be described in the following sections.

2.3 Build the *RLC* Model

In step (1), with the given feasible parameters, FastCap [15] and FastHenry [16] are used to extract the *RLC* parameters of the bus and construct the SPICE model. The detailed flow of step (1) is shown in **Figure 13**. FastCap can extract the self and coupling capacitance of wires, while FastHenry is developed to extract the resistance, self inductance, and coupling inductance. With these extracted *RLC* parameters, the equivalent *RLC* circuit models will be constructed. The circuit models are constructed as π -segments using series resistances and inductances and shunt capacitances. The circuit model will be outputted as a SPICE file.

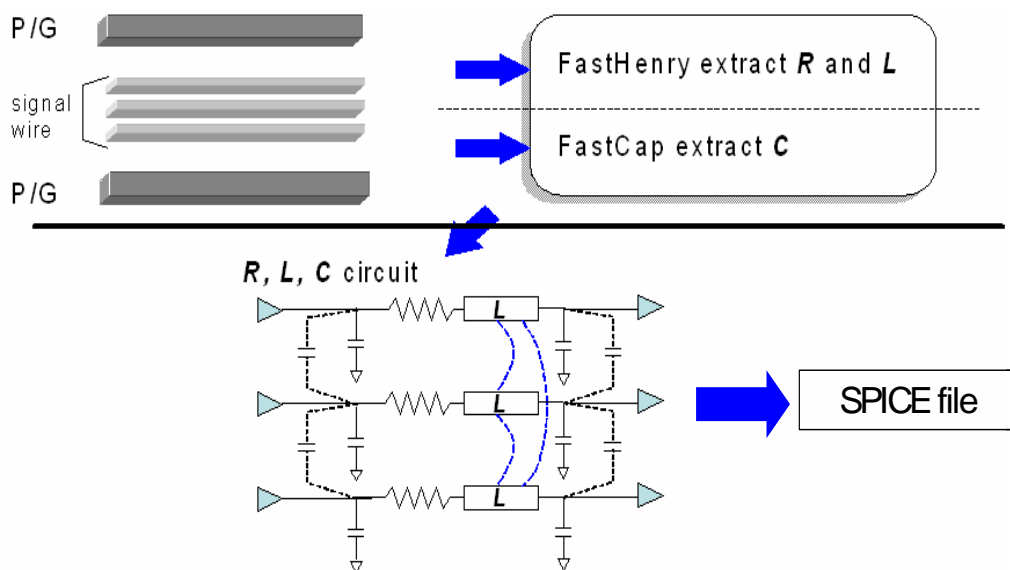


Figure 13: Extract *RLC* and generate the corresponding SPICE file.

2.4 SPICE Simulation for *Basis Vectors*

After building the equivalent *RLC* circuit model of the bus, one can obtain the transition delay by simply using HSPICE simulation. However, for an n -bit bus, there are 2^n input patterns and total 4^n kinds of transition patterns. If we simulate all transition patterns using only HSPICE, it will be very time-consuming. The complexity of simulation time will be $4^n \cdot (\text{HSPICE simulation time for a transition pattern})$. Therefore, we develop a method based on superposition theorem [14] to speed up the simulation time. The superposition theorem states that, for an *LTI* circuit, the resulting effects (currents and voltage differences) of the independent current (or voltage) sources in the circuit can be considered (calculated) separately, and then summed up to obtain the overall results. Based on this idea, we first simulate the *basis vectors* which are independent sources in the built *RLC* circuit. Then we can obtain the real delay of each transition pattern by superposing the results of the *basis vectors*.

What are the *basis vectors* of a bus? We define them as all independent transitions of a bus inputs. The *basis vectors* of the 3-bit bus are $--\uparrow$, $-\uparrow-$, $\uparrow--$, $--\downarrow$, $-\downarrow-$ and $\downarrow--$. Throughout the thesis, “ $-$ ” stands for stable input (i.e. $0 \rightarrow 0$ or $1 \rightarrow 1$), “ \uparrow ” stands for input changing from low to high, and “ \downarrow ” stands for input changing from high to low. There are two properties of *basis vectors* shown in the following.

Property 1: In the equivalent linear *RLC* circuits, given a *basis vector* (e.g. $--\uparrow$), the transition delay of switching inputs are the same whether other stable

inputs are stable at '0' or '1' (e.g. 000→001 and 110→111 have the same transition delay).

Proof: Coupling effects result from the voltage changing ($I_{induce} = C_{couple} \times (dV/dt)$) and current changing ($V_{induce} = L_{couple} \times (dI/dt)$), stable inputs will not contribute noise to neighbor wires. Therefore, the transition delays of switching inputs are the same whether other stable inputs are stable at '0' or '1'. □

Property 2: In the equivalent linear **RLC** circuits, $(--\uparrow, --\downarrow)$, $(-\uparrow-, -\downarrow-)$, and $(\downarrow--, \uparrow--)$ are called *dual basis vector pairs*. The voltage waveforms resulting from a *dual basis vector pair* such as $(--\uparrow, --\downarrow)$ will be equal in magnitude but opposite in directions.

Proof: Since the extracted **RLC** circuits are composed by linear elements, the built circuit is a **LTI** (linear time invariant) system. Therefore, if we input signals with the same magnitude but opposite directions, then the output waveforms will be in equal magnitude but in opposite directions. □

Based on the two properties, we define a *minimum basis vector set* as that all transition patterns can be obtained by superposing the *basis vectors* of this set. Please note that the *basis vector sets* of a 3-bus are $(\{--\uparrow, --\downarrow\}, \{-\uparrow-, -\downarrow-\}, \{\uparrow--, \downarrow--\})$. Therefore, there are eight choices of the *minimum basis vector set* for a 3-bit bus (e.g. $(--\uparrow, -\uparrow-, \uparrow--)$, $(--\uparrow, -\uparrow-, \downarrow--)$, $(--\uparrow, -\downarrow-, \uparrow--)$, $(--\uparrow, -\downarrow-, \downarrow--)$, $(--\downarrow, -\uparrow-, \uparrow--)$, $(--\downarrow, -\uparrow-, \downarrow--)$, $(--\downarrow, -\downarrow-, \uparrow--)$ and $(--\downarrow, -\downarrow-, \downarrow--)$). The *minimum basis vector set* for a 3-bit bus can be arbitrary one of the eight choices. Hence, we only need to simulate the

basis vectors of the chosen *minimum basis vector set*. Then we can use the simulation results to obtain the delays of all possible transition patterns by applying the superposition theorem. The details and examples of step (2) are shown in **Figure 14**. The example in **Figure 14** demonstrates the *minimum basis vector* simulation flow for a 3-bit bus. First, we choose one *minimum basis vector* from the *minimum basis vector set* at a time as the input transition pattern on the bus. Second, we perform SPICE simulation and record the voltage waveform of each signal wire. Then, we repeat the first step for another *minimum basis vector* until whole *minimum basis vector set* is simulated.

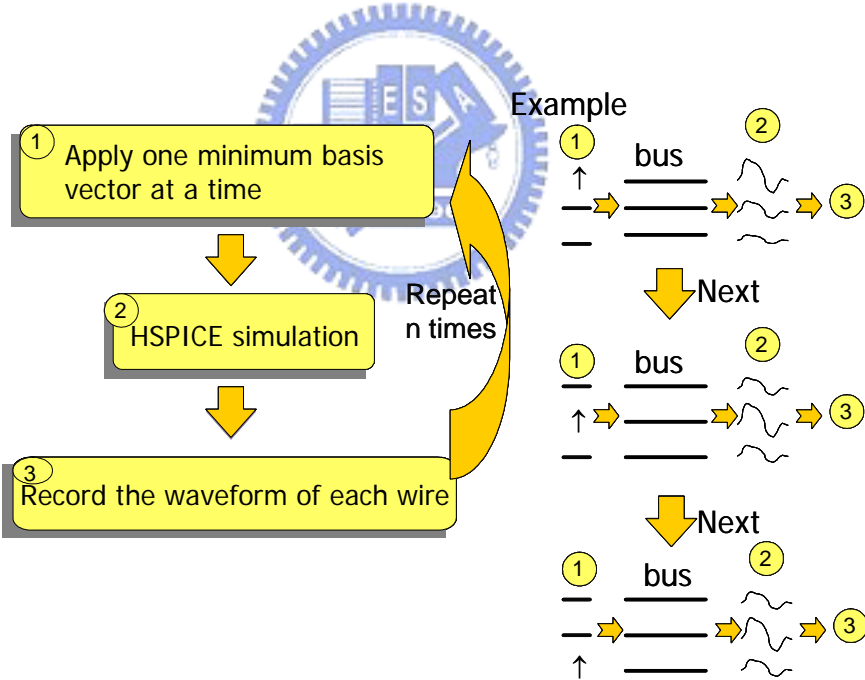


Figure 14: HSPICE simulations with the *minimum basis vectors* of the *minimum basis vector set*.

2.5 Build the Transition Graph

In step (3), we apply the superposition theorem to calculate the real transition delay of each transition pattern by using the simulation results of the *basis vectors* in step (2). **Figure 15** illustrates how to obtain the real delay of a transition pattern by using the simulation results of the *basis vectors*. First, we decompose the transition pattern into some *basis vectors*. Then, by looking up the simulation results of the *basis vectors* that have been simulated in step (2) and superposing them, we can obtain the real voltage waveform of the transition pattern. Finally, the transition delay of this transition can be obtained. To verify the accuracy of this method, we conduct various simulations and compare the voltage waveforms obtained by using the superposition method and by HSPICE simulation. The simulation results show that both methods have exactly the same voltage waveforms in various simulations. We give an example in **Figure 16** to demonstrate the accuracy of the superposition method. For a 3-bit bus structure, **Figure 16 (a)** shows the voltage waveform of each signal wires at the receiver side by directly conducting HSPICE simulation. The transition pattern is $-\uparrow\uparrow$ and the supply voltage is 1.2V. The voltage waveforms obtained by using the superposition method is as shown in **Figure 16 (b)**. These voltage waveforms of the transition pattern (i.e. $-\uparrow\uparrow$) are obtained by superposing the pre-simulation results of the two basis vectors, $-\uparrow$ and $-\uparrow-$. Comparing **Figure 16 (a)** and **Figure 16 (b)**, the voltage waveforms obtained by using two methods are exactly the same. In addition, to demonstrate the accuracy of the superposition method more clearly, we also show some check points on the voltage wave forms. The voltage values of the check points in **Figure 16 (b)** are exactly the same with

those in **Figure 16 (a)**. Thus, we can conclude that by using the superposition method, we can obtain voltage waveforms exactly equivalent to those obtained by directly conducting HSPICE simulation.

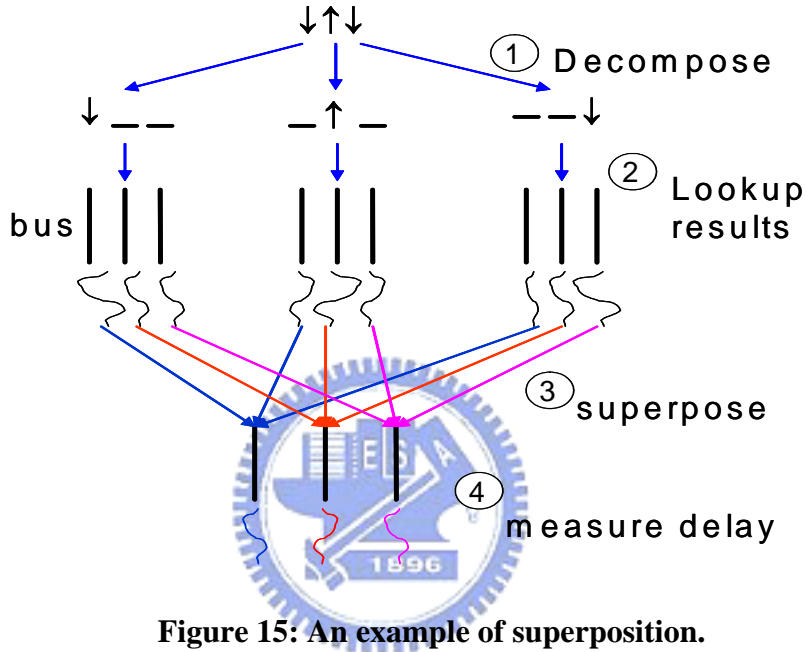
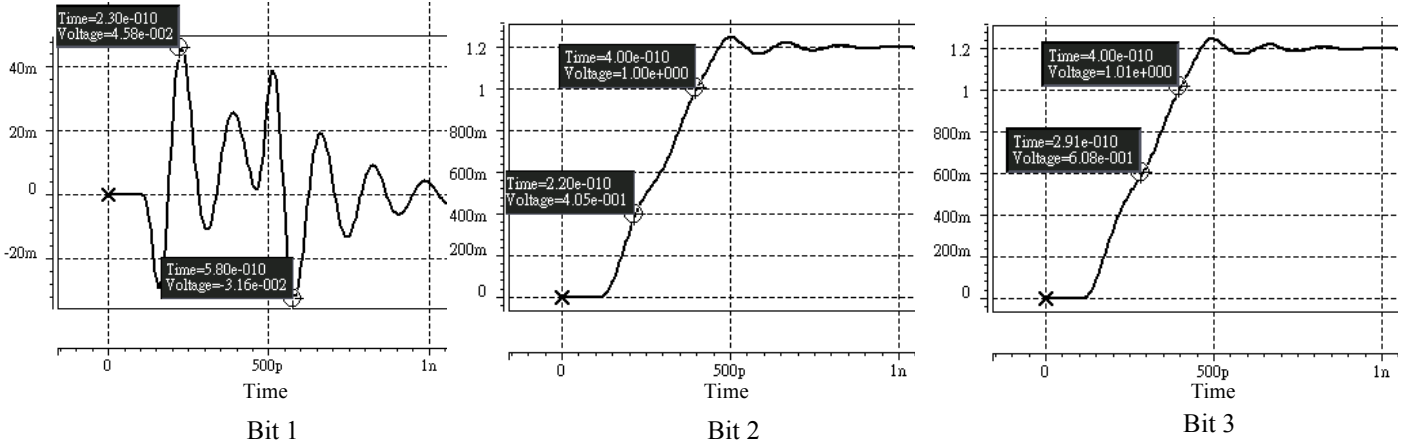


Figure 15: An example of superposition.

The voltage waveform by directly conducting HSPICE simulation:

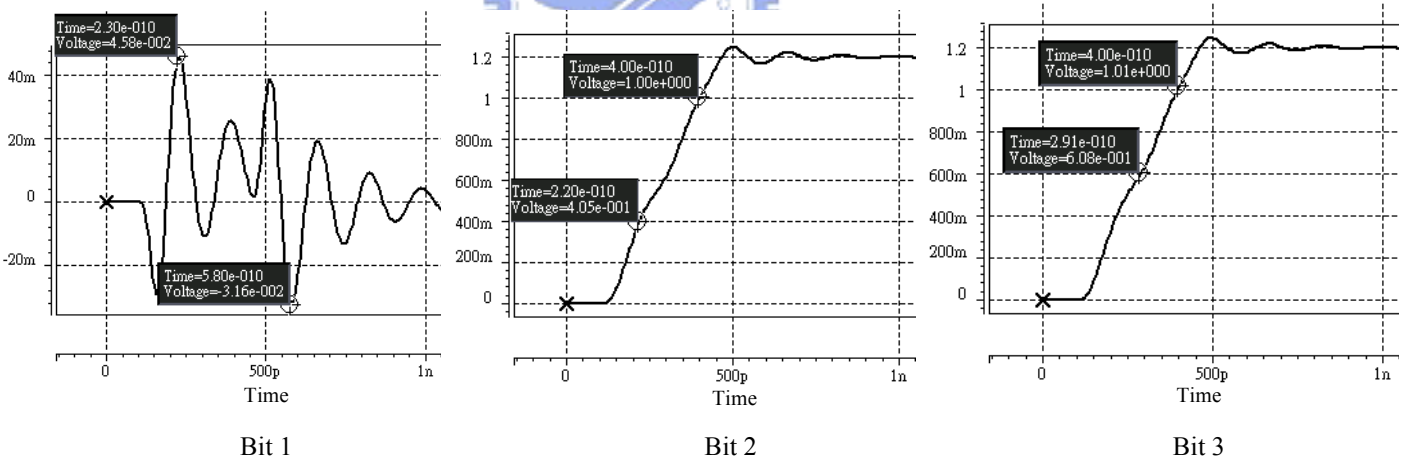
(- ↑ ↑)



(a)

The voltage waveform by using superposition method:

(-- ↑ + - ↑ - = - ↑ ↑)



(b)

Figure 16: The voltage waveforms of the signal wires obtained (a) by directly conducting HSPICE simulation and (b) by using superposition method.

With the use of the superposition method, we can effectively calculate the transition delay between any two input patterns by looking up and superposing the pre-simulation results of the *minimum basis vector set*. In next step, we build a transition graph to record if the transition delay between input patterns meets the delay constraint or not. **Figure 17** illustrates an example of a transition graph of 3-bit bus where vertices represent input patterns and edges indicate that the transition delay between two input patterns meets the delay constraint.

With the use of the superposition theorem on an n -bit bus, the complexity of obtaining all transition patterns' delays will be reduced from $(4^n) \cdot (\text{HSPICE simulation time for a transition pattern})$ to $n \cdot (\text{HSPICE simulation time for a transition pattern}) + (4^n/2) \cdot (\text{superposition time})$. The first term, $n \cdot (\text{HSPICE simulation time for a transition pattern})$, is for simulating the *minimum basis vector set*, and the second term, $4^n/2 \cdot (\text{superposition time})$, is for mathematical computations which superpose the *minimum basis vectors*' simulation results.

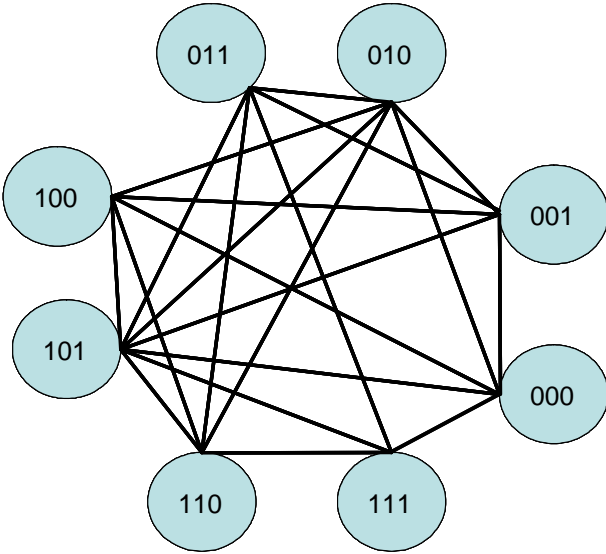


Figure 17: The transition graph.

2.6 Find a Maximum Valid Code Set

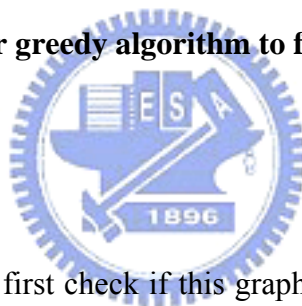
After building the transition graph, we can obtain a maximum valid code set by finding a maximum clique in the graph. The reason to find a clique in the transition graph is that we want to obtain a valid code set within which any transition between two codes is guaranteed to meet the delay constraint. Besides, the reason to find a maximum clique is because the number of found codes (vertices) should be greater or equal to the number of required data patterns (e.g. 2^n in **Figure 10**) under the given wire overhead (e.g. (m-n) in **Figure 10** (b)). If the number of found valid codes is greater than that of required data patterns, it indicates that we could possibly use less extra wires to generate the valid code set that meet the delay constraint. Therefore, we are always eager to maximize the number of found codes to minimize the wire overhead. However, since finding a maximum clique is an NP-complete problem, we use a greedy algorithm to solve this problem within reasonable time. The pseudo code of the greedy algorithm is shown in **Figure 18**.


```

INPUT: A transition graph
OUTPUT: A clique
If the graph is a clique, output this graph and exit
REPEAT
    Find a vertex in the graph whose degree is the smallest
    Remove that vertex and update the graph
UNTILE All remained vertices in the graph form a clique
All vertices in the clique form a set CLIQUE
All vertices that deleted from the graph form a set DELETE
REPEAT
    Find a vertex in the set DELETE
    IF the vertex and the set CLIQUE can form a larger clique THEN
        Renew the set CLIQUE to the larger one
UNTILE All vertices in the set DELETE have
        been tried to add in the set CLIQUE
Output the set CLIQUE

```

Figure 18: Our greedy algorithm to find a maximum clique.



In the first loop, we first check if this graph is a clique. If it is a clique, we output this graph directly. Otherwise, we delete the vertex whose edge degree is the smallest and update the graph. Following, we redo above steps until the remaining vertices form a clique. Take **Figure 17** as an example, the edge degree of each vertex in **Figure 17** is $\{5, 5, 6, 5, 5, 7, 5, 4\}$ from vertex (000) to vertex (111) counterclockwise. Obviously, this input graph is not a clique. Hence, in the first loop, we first remove (111) since it has smallest edge degree and then update the graph. Next, we pick (000) as the next vertex to be removed. Then (001) and (011) are selected to be removed subsequently. The removed vertices will be stored in the set DELETE for the further check in the second loop of **Figure 18**. This loop will be repeated until the remained vertices in the graph can form a

clique. Finally, the remaining vertices (010, 100, 101, 110) will form a clique as shown in **Figure 19**.

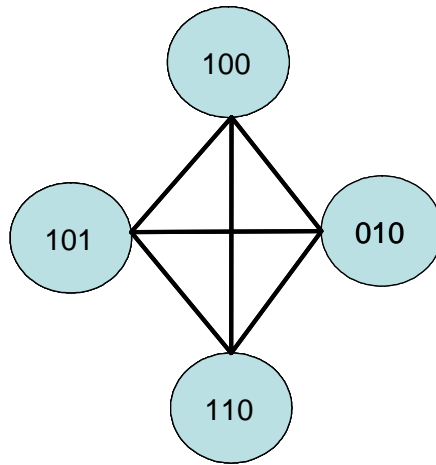


Figure 19: The greedy search result of Figure 17.

Since the greedy method in the first loop of the **Figure 18** is a simple heuristic algorithm, it could not guarantee to find an optimal solution. In order to further improve the result, we add extra steps in the second loop in **Figure 18** to improve the outcome.

The second loop tries to add vertices that are stored in the DELETE set back into the identified clique. Each attempt is checked if the newly added vertex can form a larger clique. If this is the case, the clique is updated to a larger one. By using this method, we could find a larger clique without consuming significant computing power. From the simulation results, we observe that the second loop can really enlarge the clique found in the first loop.

To demonstrate the effectiveness of the second loop in **Figure 18**, we also conduct various simulations. The simulation results are as shown in **Figure 20**. In

Figure 20, the x-axis represents the number of the found clique size using the complete greedy algorithm in **Figure 18**, while the y-axis represents the number of nodes added back from the DELETE set in the second loop of the greedy algorithm. From the simulation results in **Figure 20**, we observe that the second loop could enlarge the clique size that is found in the first loop. In average, the second loop can enlarge about 8% of the clique size obtained in the first loop in **Figure 18**.

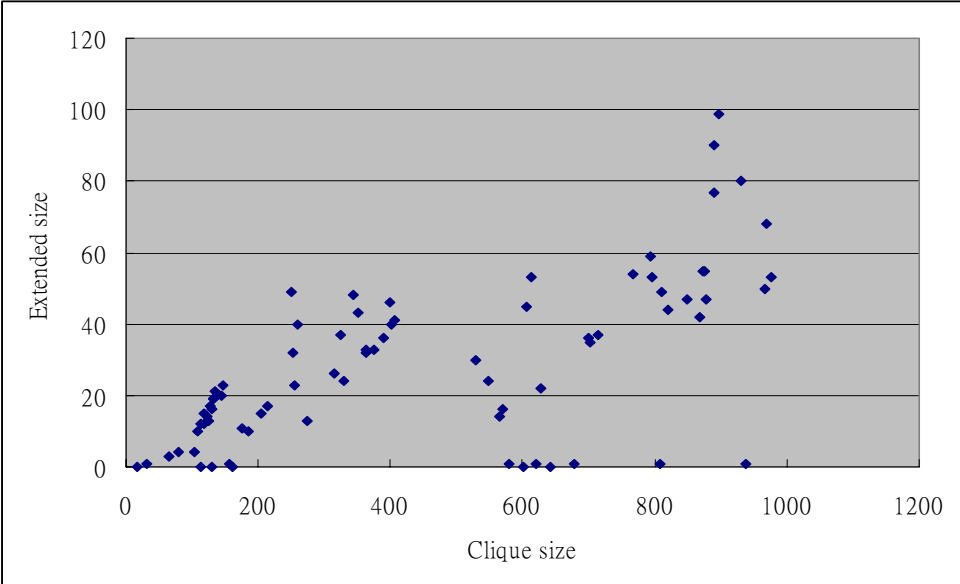


Figure 20: The number of extended clique size by using the second loop in Figure 18.

2.7 Reduce Simulation Time by Using Superposition Theorem

In this section, we compare the computation time of building the transition graph using our superposition method and pure HSPICE simulations. Our

simulation environment is described as follows. The length, width, height and pitch of signal wires are $2000\mu\text{m}$, $2\mu\text{m}$, $2\mu\text{m}$ and $4\mu\text{m}$, respectively. The bus working frequency is 1GHz and the supply voltage is 1.2V. The bit number of the bus width varies from 3 to 18. The simulation results are shown in **Figure 21** and **Table 6**. The speedup ratio is calculated as $((\text{Pure HSPICE}) / (\text{Our Method})) - 1$. Obviously, our approach is much more effective than pure HSPICE simulations especially for the wider bus. **Table 6** illustrates that our method provides a dramatic speedup against pure HSPICE simulation. Moreover, the transition graphs obtained by our method and pure HSPICE simulations are exactly the same. In other words, as a result of the superposition theorem, the transition delays that obtained by our method and pure HSPICE are exactly identical.

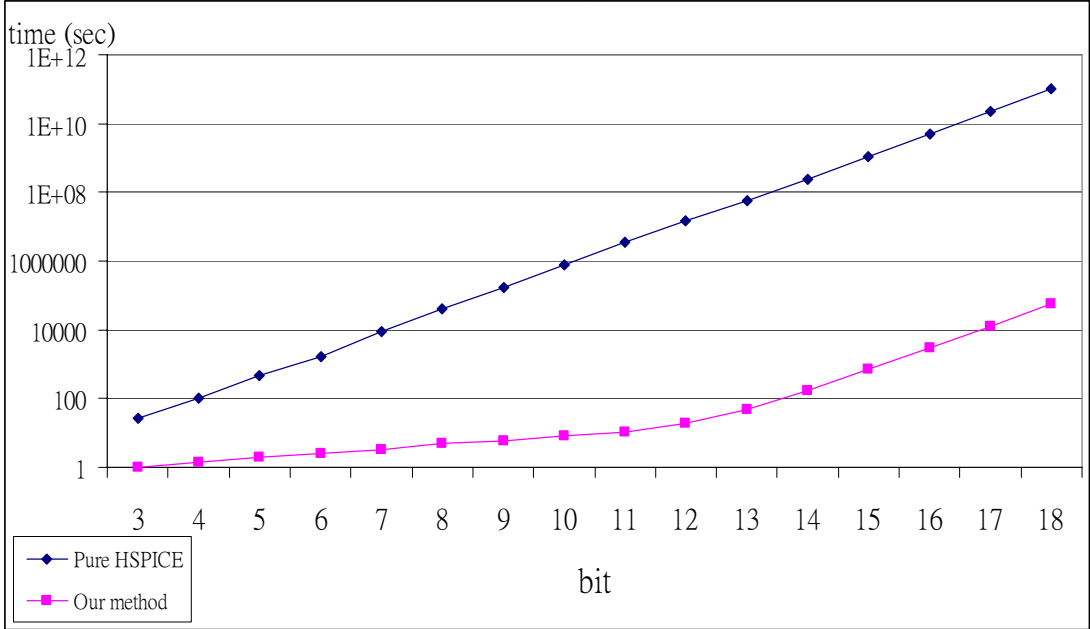


Figure 21: The run time of our method and pure HSPICE simulation.

Table 6: The run time improvement using our method.

Bit	Pure HSPICE (sec)	Our Method (sec)	Speedup
3	27.0	1	26.0
4	104.2	1.4	72.9
5	463.9	1.9	243.2
6	1728.5	2.6	663.8
7	8699.9	3.3	2635.3
8	39977.0	4.9	8157.6
9	176160.8	6.0	29359.1
10	802160.6	7.9	101538.3
11	3405774.8	11.0	309614.9
12	14931722.0	19.4	769675.4
13	57646516.0	48.1	1198471.3
14	247497488.0	176.2	1404638.5
15	1124207744.0	717.8	1566184.2
16	4964982272.0	3002.9	1653394.8
17	22282291200.0	12536.8	1777349.8
18	98749890560.0	57361.5	1721535.1

2.8 Simulation Results of Delay Reduction

In the following, we give an example to demonstrate the delay reduction using our encoding flow. Given a 6-bit bus (i.e. 64 data patterns), and the delay constraint is set to 56 picoseconds. The transition delays of all transition patterns on the 6-bit bus are shown in **Figure 22**. Each point on the graph represents a transition delay. Obviously, some transitions violate the delay constraint. By using our encoding flow for the 6-bit bus, the size of the found maximum valid code set is 38, which is smaller than 64 (less than the number of data patterns). Thus, we will add one more bit to the 6-bit bus and redo our encoding flow. After using our encoding flow for the 7-bit bus, a maximum valid code set of size 73 is found. All transition delays between any two patterns in this valid code set meet the delay constraint as shown in **Figure 23**. Therefore, we output the valid code set with the 7-bit bus as the final result.

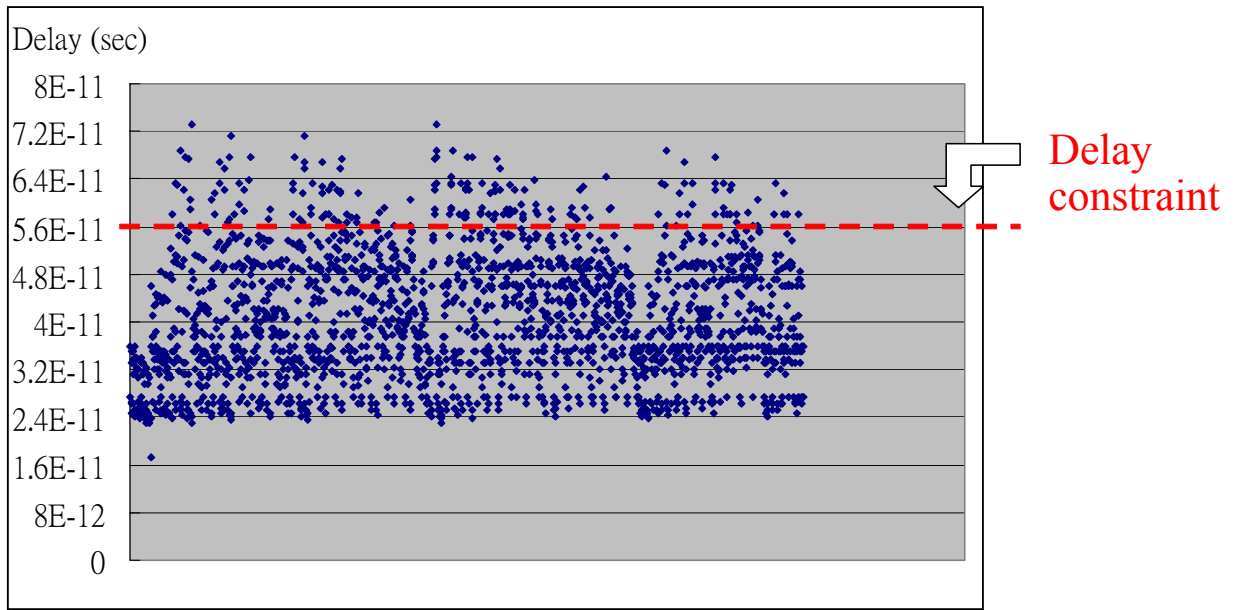


Figure 22: Transition delay of the bus (6 bit) before encoding.

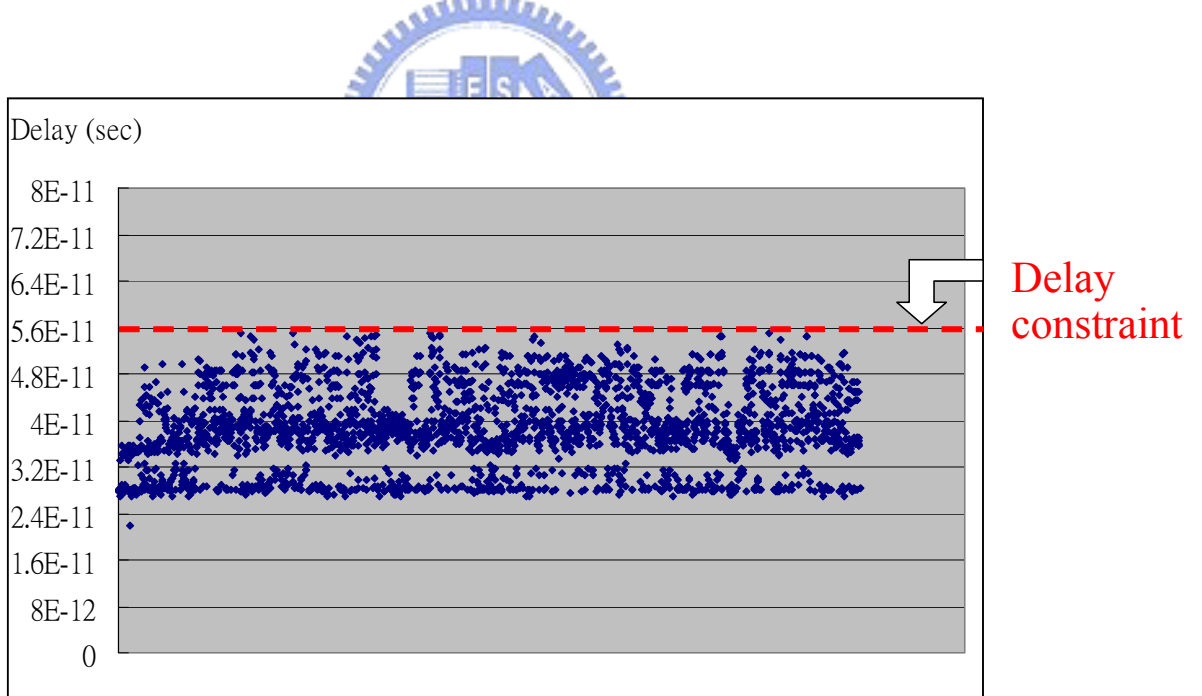


Figure 23: Transition delay of the bus (7 bit) after encoding.

2.9 Delay Reduction under Different Working Conditions

In this sub-section, we will show the flexibility of our encoding flow on reducing the delay with various working frequencies, user-given delay constraints, and wire dimensions. First, we will discuss the results of our encoding flow with considering different delay constraints and then demonstrate the flexibility with considering different working frequencies. Finally, different wire dimensions will be considered and demonstrated as well.

The delay constraint directly decides that the types of the transitions which are allowed to transmit on the bus (i.e. meet the delay constraint). When the delay constraint is tight, few types of transitions meet the delay constraint. Therefore, we need more extra wires to encode the data patterns. On the other hand, if the delay constraint is loose, many types of transitions are allowed to be transmitted on the bus. Hence, only few extra wires are needed to encode the data patterns. We give an example to demonstrate this phenomenon by using our encoding flow with different delay constraints. Give a bus parameter which the length, width, height and pitch of signal wires are $1000\mu\text{m}$, $2\mu\text{m}$, $2\mu\text{m}$ and $4\mu\text{m}$, respectively. The number of data bits is 4 (16 data patterns). The simulation results of needed extra wires with different user-given delay constraints are listed in **Table 7**. The second column in **Table 7** (i.e. m-bit bus) represents the bit number needed to generate the valid code set using our encoding flow. The last column represents the wire overhead ratio which is calculated as $(m - 4) / 4$. From **Table 7**, we can observe that when the delay constraint is loose (i.e. larger than 26 picoseconds),

all transitions of the 4-bit bus meet the delay constraint. This means all transitions are allowed to be transmitted on the bus. Thus, no additional wires are needed to encode the data patterns and the wire overhead ratio is 0. When the delay constraint is tight (e.g. delay constraint = 25 picoseconds), some transitions of 4-bit bus are invalid for the delay constraint. Hence, an additional wire is needed to encode and find a valid code set that can map the 16 data patterns. In other word, we need an additional wire to enlarge the size of generated valid code set that is large or equal to the number of the data patterns. As the delay constraint is tighter, fewer types of transitions are allowed to be transmitted on the bus. Thus, we will need more extra wires to encode the data patterns. In other words, the wire overhead ratio becomes very large when the delay constraint is very tight. From **Table 7**, we can observe that when the delay constraint is set to 17 picoseconds, 5 extra wires are needed to meet the delay constraint by using our encoding method.

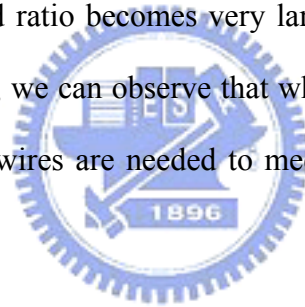


Table 7: The wire overheads versus different delay constraints for a 4-bit data bus.

Delay constraint (ps)	m-bit bus	Wire overhead
30	4	0.00
29	4	0.00
28	4	0.00
27	4	0.00
26	4	0.00
25	5	0.25
24	5	0.25
23	5	0.25
22	5	0.25
21	6	0.50
20	6	0.50
19	6	0.50
18	7	0.75
17	9	1.25

Next, we consider different bus working frequencies in our simulations and discuss the results. Since the wire impedance $Z = R + (j\omega C)^{-1} + j\omega L$, the wire impedance and behavior changes for different bus working frequencies. As mentioned in the introduction, the portion of $(j\omega C)^{-1}$ (where $\omega = 2\pi f$) decreases with the increase of the working frequency, while the portion of $j\omega L$ increases. Therefore, the wire behavior of a bus might be capacitive, inductive, or mixed depending on the working frequency for the same wire dimensions. Thus, the worst-case switching patterns will change for different working frequency as well.

Hence, different encoding results will be generated for different working frequencies.

In the following, we will give an example to demonstrate the phenomenon for different working frequencies. Given a bus structure with length, width, height and pitch of signal wires are $2000\mu\text{m}$, $2\mu\text{m}$, $2\mu\text{m}$ and $4\mu\text{m}$, respectively. The delay constraint is set to 50 picoseconds. For a 3-bit bus (8 data patterns), the simulation results for different working frequencies are listed in **Table 8**. The second column in **Table 8** represents the worst-case switching patterns for the 3-bit bus. The third column shows the bit number needed to generate the valid code set using our encoding flow, and the last column represents the wire overhead ratio which is calculated as $(m - 3) / 3$. In **Table 8**, when the working frequency is less than 3.2GHz, the worst-case switching pattern is that the neighboring wires switch in the opposite directions of the victim wire (i.e. $\downarrow\uparrow\downarrow$ when the second bit line is considered as the victim wire). This implies that the interconnect delay is dominated by the capacitive coupling effect in this case. Under this condition, our encoding method needs one extra wire to generate the valid code set to map the 8 data patterns. However, when the working frequency is up to 6.4GHz, the worst-case switching pattern is changed from $\downarrow\uparrow\downarrow$ to $\uparrow\uparrow\uparrow$. This implies that the interconnect delay is now dominated by the inductive coupling effect. Under this condition, we find out that all transitions for 3-bit bus meet the delay constraint and no additional wire is needed for encoding.

Table 8: The encoding results for different working frequencies for a 3-bit bus.

Working frequency (MHz)	Worst-case switching pattern	m-bit bus	Wire overhead
200	↑↓↑	4	0.333
400	↑↓↑	4	0.333
800	↑↓↑	4	0.333
1600	↑↓↑	4	0.333
3200	↑↓↑	4	0.333
6400	↑↑↑	3	0.000

In addition, we also conduct simulations for various number of data bits (i.e. 6-bit data, 8-bit data, and 9-bit data) and list the simulation results in **Table 9, 10, and 11**. From these tables, we can observe that different working frequencies and different number of data bits will result in different worst-case switching patterns (i.e. different capacitive and inductive coupling effects for the interconnect delay). However, our encoding flow is still capable of considering these conditions and generating the respective valid code sets that can map to all data patterns for buses. Hence, from the simulation results listed in **Table 8, 9, 10, and 11**, it is clearly that our encoding flow is also flexible for different working frequencies.

Table 9: The encoding results for different working frequencies for a 6-bit bus.

Working frequency (MHz)	Worst-case switching pattern	m-bit bus	Wire overhead
200	↑↓↑↓↓↓	7	0.167
400	↑↓↑↑↑↑	7	0.167
800	↑↓↑↑↑↑	9	0.500
1600	↑↓↑↓↓↓	8	0.333
3200	↑↑↑↑↑↑	8	0.333
6400	↑↑↑↓↑↓	8	0.333

Table 10: The encoding results for different working frequencies for a 8-bit bus.

Working frequency (MHz)	Worst-case switching pattern	m-bit bus	Wire overhead
200	↑↑↑↑↑↓↑↓	10	0.250
400	↑↑↑↑↑↓↑↓	10	0.250
800	↑↑↑↑↑↓↑	10	0.250
1600	↑↑↑↑↑↓↑↓	12	0.500
3200	↑↑↑↑↑↓↑↓	11	0.375
6400	↑↑↑↑↑↓↑↓	11	0.375

Table 11: The encoding results for different working frequencies for a 9-bit bus.

Working frequency (MHz)	Worst-case switching pattern	m-bit bus	Wire overhead
200	↑↑↑↑↑↓↑↓	11	0.222
400	↑↑↑↑↑↓↑↓	11	0.222
800	↑↑↑↑↑↓↑↑	12	0.333
1600	↑↑↑↑↑↓↑↓	13	0.444
3200	↑↑↑↑↑↓↑↓	13	0.444
6400	↑↑↑↑↑↓↑↓	12	0.333

Finally, we consider the effects of different user-given wire dimensions. Different wire dimensions will result in different capacitive and inductive coupling effects and, hence, different worst-case switching patterns and delays for buses. To demonstrate this phenomenon, we change the wire dimensions in our simulations. Given a bus structure with the wire length, width, height and pitch of signal wires are 2000 μm , 4 μm , 8 μm and 10 μm , respectively. The delay constraint is set to 50 picoseconds. For a 6-bit bus, the simulation results using our encoding method are shown in **Table 12**. Comparing **Table 9** with **Table 12**, we can observe that for the same bus working frequency the worst-case switching patterns are different due to different wire dimensions (e.g. when working frequency is at 200MHz, the worst-case switching pattern in **Table 9** is $\uparrow\downarrow\uparrow\downarrow\downarrow$, but in **Table 12** is $\uparrow\uparrow\uparrow\downarrow\uparrow\downarrow$). Therefore, the coupling effects are very

different for a bus with different wire dimensions. However, our encoding flow is still capable of considering different wire dimensions and generating the respective valid code sets that can map to all data patterns for buses as shown in **Table 12**. Thus, our bus encoding flow is indeed very flexible for considering various design parameters.

Table 12: The encoding results for different wire dimensions for a 6-bit bus.

Working frequency (MHz)	Worst-case switching pattern	m-bit bus	Wire overhead
200	↑↑↑↓↑↓	6	0.000
400	↑↑↑↓↑↑	7	0.167
800	↑↑↓↑↑↑	8	0.333
1600	↑↑↑↓↑↓	7	0.167
3200	↑↓↓↓↓↓	7	0.167
6400	↑↑↑↑↑↑	7	0.167

2.10 The Performance Comparison between Shield Insertion Technique and Our Encoding Method

In this sub-section, we conduct some simulations to compare the improvement of our flow with the conventional shield insertion technique. Given a bus structure, the number of data bit is 4 (16 data patterns), and the wire

overhead is one (only one additional wire is allowed). The bus length, width, height and pitch of signal wires are $2000\mu\text{m}$, $0.8\mu\text{m}$, $2\mu\text{m}$ and $2\mu\text{m}$, respectively. The number of data bits is set to 6 (i.e. 64 data patterns). **Figure 24 (a)** and **(b)** illustrates the buses after using our flow and inserting one shield.

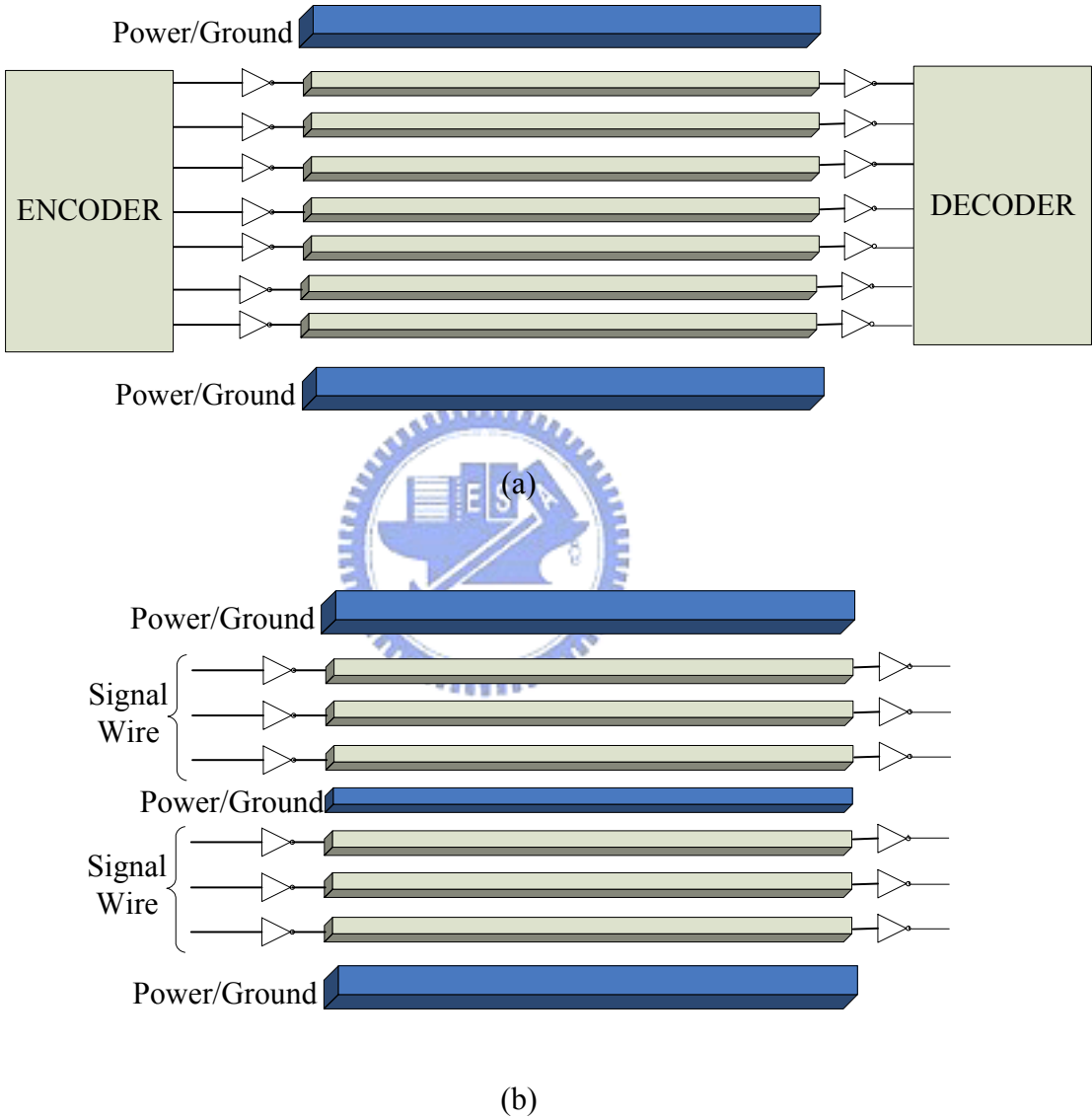


Figure 24: The bus structure (a) with the use of our encoding method and (b) with the use of shield insertion technique.

The simulation results under different working frequencies and bus lengths are conducted and the worst-case transition delays are shown in **Figure 25**.

Besides, the worst-case transition delay of the original 6-bit bus is also shown in **Figure 25**. Although both our flow and the shield insertion technique can reduce the worst-case transition delay of buses, our flow always outperforms the shield insertion technique under different working frequencies and bus lengths as shown in **Figure 25**. Therefore, our flow is effective in reducing the coupling delay under different working frequencies and bus lengths comparing to the conventional shield insertion technique.

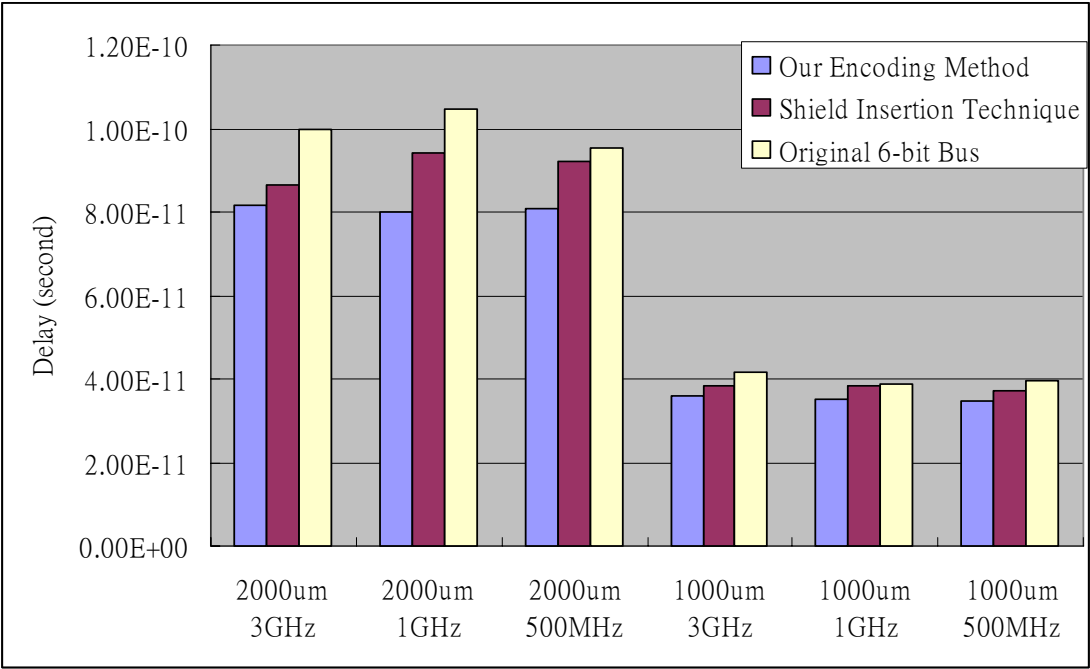
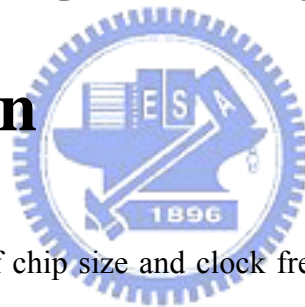


Figure 25: The worst-case transition delay by using our method and the shield insertion technique.

Chapter 3

Bus Encoding to Lengthen Signal

Propagation



With the increase of chip size and clock frequency, the global interconnect delay is likely to be larger than one clock period. Hence, on-chip signals can no longer reach the entire die in one clock cycle [17]. As shown in **Figure 26**, the percentage of the reachable chip area in one clock cycle continuously decreases as the technology advances. When the process technology enters 0.1- μm , only 16 percent of the die is reachable within one clock period (at 1.2 GHz). In other words, signals need eight pipeline stages to propagate through the entire die. Recently, due to strong noise coupling effects in DSM, the one cycle signal propagation length is further decreased especially for global interconnects. Thus, how to increase the signal propagation length in one clock cycle becomes an important issue in today high performance circuit designs.

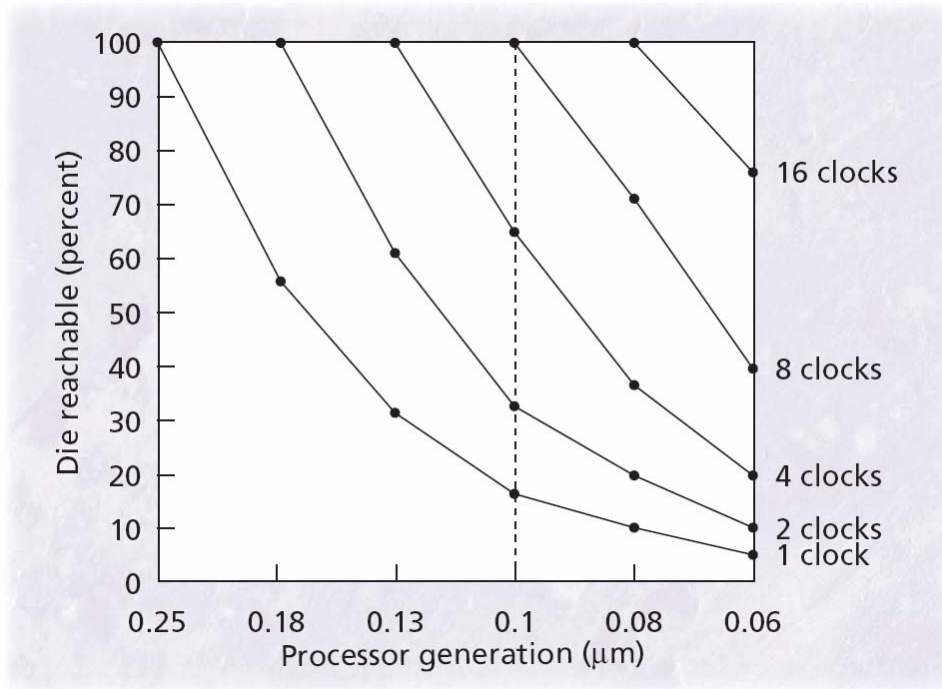


Figure 26: Trends for the clock locality metric [17].

The conventional bus encoding problem focuses on how to reduce the delay on a fixed length bus with considering only *RC* effects. In our work, we extend the conventional one to a new problem, i.e., how to increase the signal propagation length of a bus by using bus encoding methods with considering *RLC* effects under given parameters and constraints. Therefore, the conventional bus encoding problem becomes a sub-problem of our extended problem. We also propose a flexible maximum signal propagation length estimation flow to solve the extended problem. The proposed flow combines a bus encoding scheme and a curve fitting method. The bus encoding scheme can effectively reduce the *LC* coupling effects on on-chip buses, and hence, improve the worst-case switching delay. The curve fitting method can keep the proposed flow very efficient.

3.1 Problem Formulation

Again, the coplanar bus structure shown in **Figure 9** is considered in this chapter. The inputs of the problem in this chapter are the parameters of wires (i.e. wire width, high, and pitch), the working frequency (or slew rate), the delay constraint, the number of data bits, the initial bus length, and user-required precision. With these inputs, our goal is generating a valid code set that achieves the maximum propagation length. The definition the valid code set is the same as that in Chapter 2.

3.2 Maximum Signal Propagation Length Estimation Flow

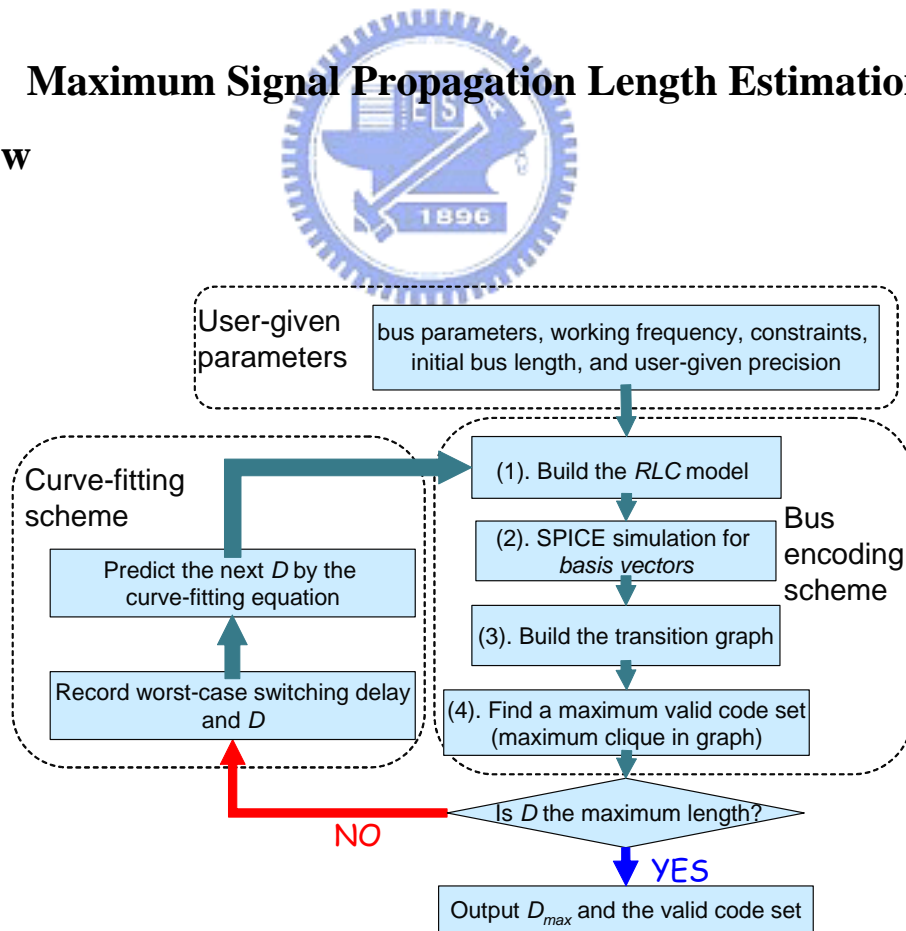


Figure 26: The maximum signal propagation length estimation flow.

To achieve the maximum propagation length, we build the maximum signal propagation length estimation flow shown in **Figure 26**. It mainly comprises two parts – the bus encoding scheme and the curve fitting method. At first, users should give the bus parameters (n-bit data, initial wire length D , wire height, wire width, wire pitch, Power/Ground wire width, Power/Ground-to-signal pitch), the user-required precision ($\triangle D_{required}$), the working frequency, and constraints (delay constraint and wire overhead constraint). Then with these parameters, we perform our encoding scheme and check if the delay constraint is still met under the wire overhead constraint (e.g., (m-n) in **Figure 1** (b)). If the delay constraint is met, the bus length is increased and predicted by the curve fitting method for the next iteration. This process is repeated until the maximum signal propagation length is obtained. When this is the case, the last length which meets the delay constraint is reported as the maximum propagation length D_{max} , and a valid code set is generated at the same time. The details of our bus encoding method and curve fitting method are discussed as follows.

The bus encoding scheme in the maximum signal propagation length estimation flow is similar to that in **Chapter 2**. Hence, in the following, we give a brief review of the encoding scheme. With user-given bus parameters and constraints, we extract and build the **RLC** model in the first step. Next, the chosen minimum basis vectors will be simulated by HSPICE with the built **RLC** model. Then, the transition graph will be constructed according to the delay constraint. Finally, we apply a greedy algorithm to find the maximum valid code set.

3.3 The Curve Fitting Method

In order to minimize the runtime of our flow, the iterations of our flow should be kept as few as possible. Hence, the maximum propagation length should be successfully predicted within few iterations instead of just incrementally increasing the predicted length for each iteration. However, if we apply a simple binary search to find the maximum length, the number of required iterations depends heavily on the given initial length. Therefore, we adopt a curve fitting method to quickly predict the maximum propagation length and thus reduce the number of required iterations.

To use the curve fitting method, we have to find a suitable fitting equation of the interconnect delay with coupling effects. A closed form delay equation is given for a gate driving an *RLC* wire segment with a gate capacitance load [21]. In [21], two extreme cases need to be considered. For one extreme case where $L \rightarrow 0$, the delay reduces to $0.37RCl^2$ where R is the unit length wire resistance, C is the unit length wire capacitance, and l is the wire length. In this case, the wire delay is squarely dependent on the wire length. For the other extreme case where $R \rightarrow 0$, the delay reduces to $l\sqrt{LC}$ where L is the unit length wire inductance. In this case, the wire delay is linearly dependent on the wire length. Therefore, it is desirable to use a quadratic equation to fit the delay of a single wire when the *LC* effects of the wire are considered. Furthermore, we can also use a quadratic equation to fit the worst-case switching delay of an n -bit parallel coupled bus

because the switching aggressors only change the effective wire capacitance and inductance of a victim wire. We also use various curve fitting methods to fit the worst-case delay curve with respect to the wire length. The simulation results are shown in **Figure 27**. From **Figure 27**, we can observe that the quadratic fitting equation matches the wire delay very well comparing to other fitting equations. Therefore, by using the quadratic curve fitting, the maximum propagation length can be predicted efficiently within few iterations in our flow.

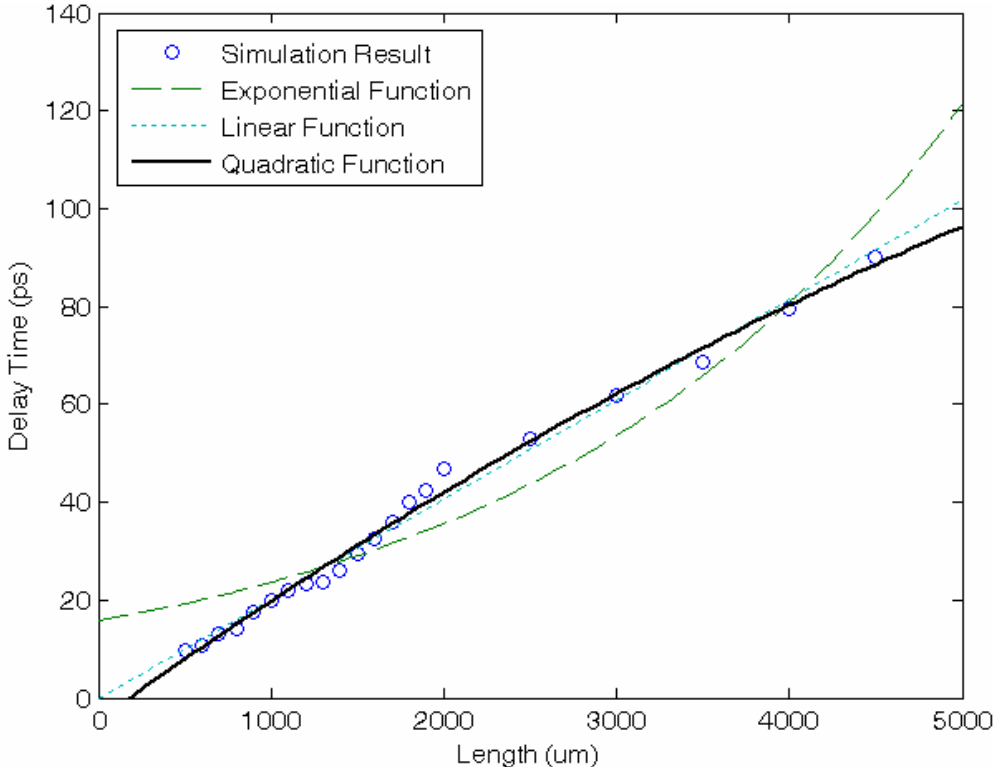


Figure 27: Curve fitting with different functions.

3.4 The Termination Condition

After the valid code set is generated by the bus encoding scheme, two conditions need to be checked before outputting the final valid code set and the maximum propagation length (D_{max}). The two conditions are shown in **Figure 28**. In the first step, we will check if the size of the found valid code set is greater than or equal to the number of the data patterns. If the size is greater than or equal to the number of data patterns, we record the current bus length (D) as the *valid length* and move to the next step for further check. Else, the current bus length will be recorded as the *invalid length* and the flow move to the curve fitting scheme to predict the next D . The *valid length* means that with this bus length, our flow can generate a valid code set whose size is greater than or equal to the number of the data patterns. On the contrary, the *invalid length* means that with this bus length, our flow can not generate a valid code set whose size is greater than or equal to the number of the data patterns.

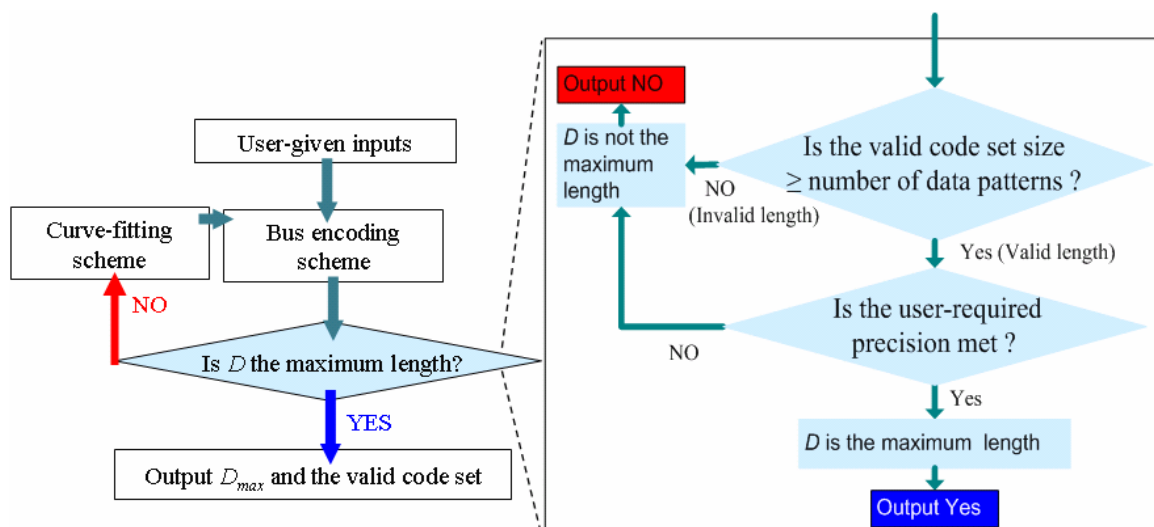
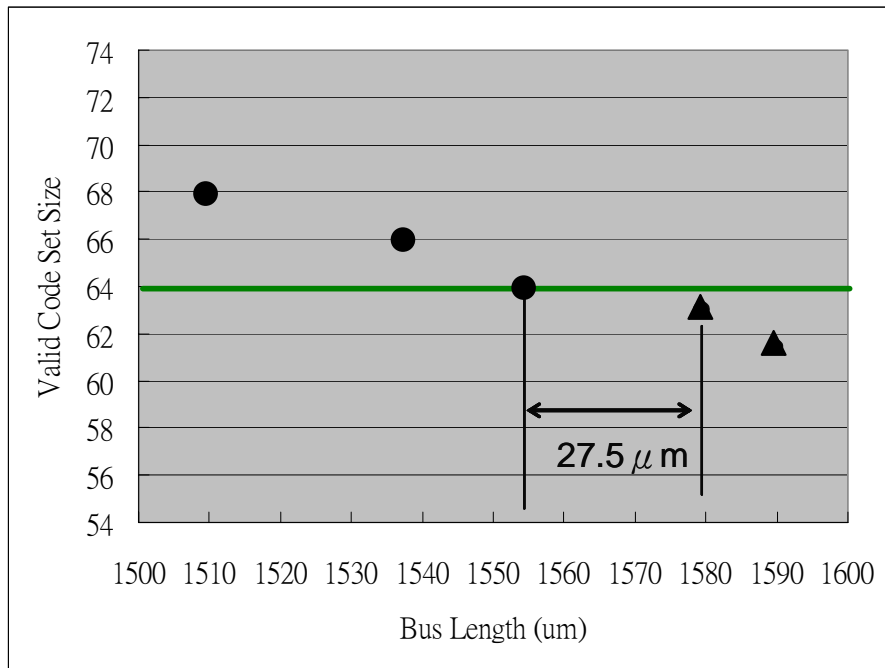
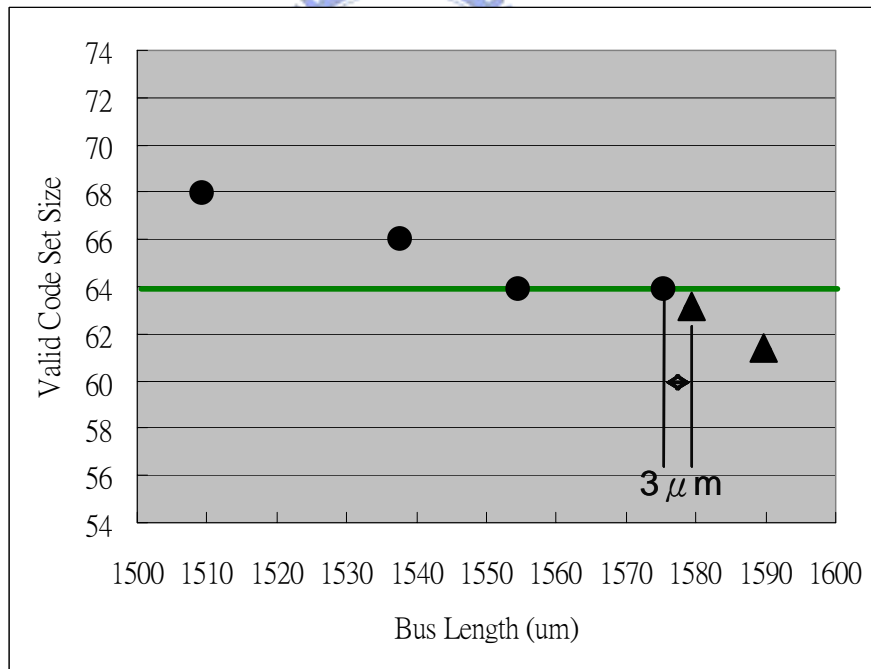


Figure 28: The decision policy of our flow.

If the current bus length (D) is recorded as the *valid length*, we move to the next step to check whether the user-required precision is met or not. In step (2), we will check if the difference of distance ΔD is smaller than user-required precision $\Delta D_{required}$, where $\Delta D = |invalid\ length - valid\ length|$. If ΔD is smaller than user-required precision, the flow will be terminated and the *valid length* will be outputted as the maximum signal propagation length with the valid code set. Else, the flow will move to the curve scheme to predict the next D to redo the bus encoding scheme. In the following, we give an example shown in **Figure 29** to demonstrate the conditions discussed in step (2). Given a bus structure, the user-required precision is set to $10\ \mu\text{m}$ (i.e. $\Delta D_{required} = 10\ \mu\text{m}$) and the number of data patterns is 64. The black points in **Figure 29** represent the *valid lengths* and the black triangle represent the *invalid lengths*. In **Figure 29 (a)**, the distance difference ΔD of the current *valid length* and the *invalid length* is larger than user-required precision $\Delta D_{required}$. On the other hand, since $\Delta D < \Delta D_{required}$ in **Figure 29 (b)**, the current *valid length* ($1576\ \mu\text{m}$) will be outputted with a valid code set as the final result.



(a)



(b)

Figure 29: (a) The distance difference ΔD ($|invalid\ length - valid\ length|$) is larger than user-required precision $\Delta D_{required}$. (b) $\Delta D < \Delta D_{required}$.

3.5 Simulation Results

With the user-given parameters, the following simulation results show that our flow can increase the signal propagation length of buses by reducing coupling effects. For example, if the delay constraint is set to 20ps, and the width, height and pitch of signal wires are given as 2 μ m, 2 μ m and 4 μ m, respectively. The bus working frequency is 1GHz and the supply voltage is 1.2V. For a 4-bit data bus (16 data patterns), the simulation results of the signal propagation length with different numbers of wires are shown in **Figure 30**. From **Figure 30**, we can observe that the improvement of the maximum propagation length is generally better when adding more wires into the bus (more wire overhead). Similar simulation results can be obtained in **Figure 31** for a 6-bit data bus (64 data patterns). To verify whether the estimated maximum propagation length and the valid code set are correct, SPICE simulation is conducted. It is confirmed that all transitions meet the given delay constraint. Thus, our flow can increase the signal propagation length by reducing the coupling effects under the given delay constraint.

However, the improvement of the maximum propagation length becomes less after adding a certain number of wires into the bus as shown in **Figure 30** and **31**. If we continuous adding wires, more added wires may even decrease the improvement of the propagation length. This is because the additional wires increase the bus width (i.e. increase the current loop width of each wire). Therefore, the inductance effects on the buses increase with every additional wire as well. Hence, there is an optimal number of extra wires for a given bus

structure and working frequency. By using our flow, designers can easily obtain the improvement curve of the maximum propagation length regarding to different number of additional wires. With this information, designers can obtain the optimal number of additional wires and estimate the maximum propagation length of buses.

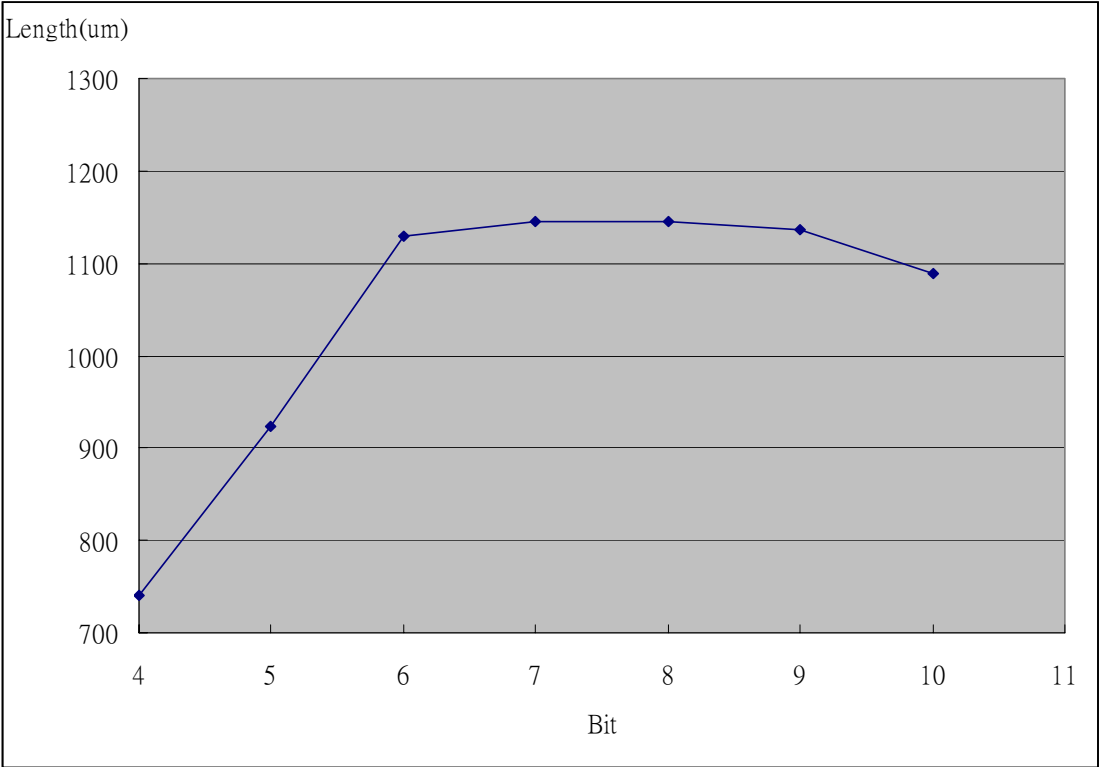


Figure 30: The maximum propagation length vs. different wire overheads for a 4-bit data bus.

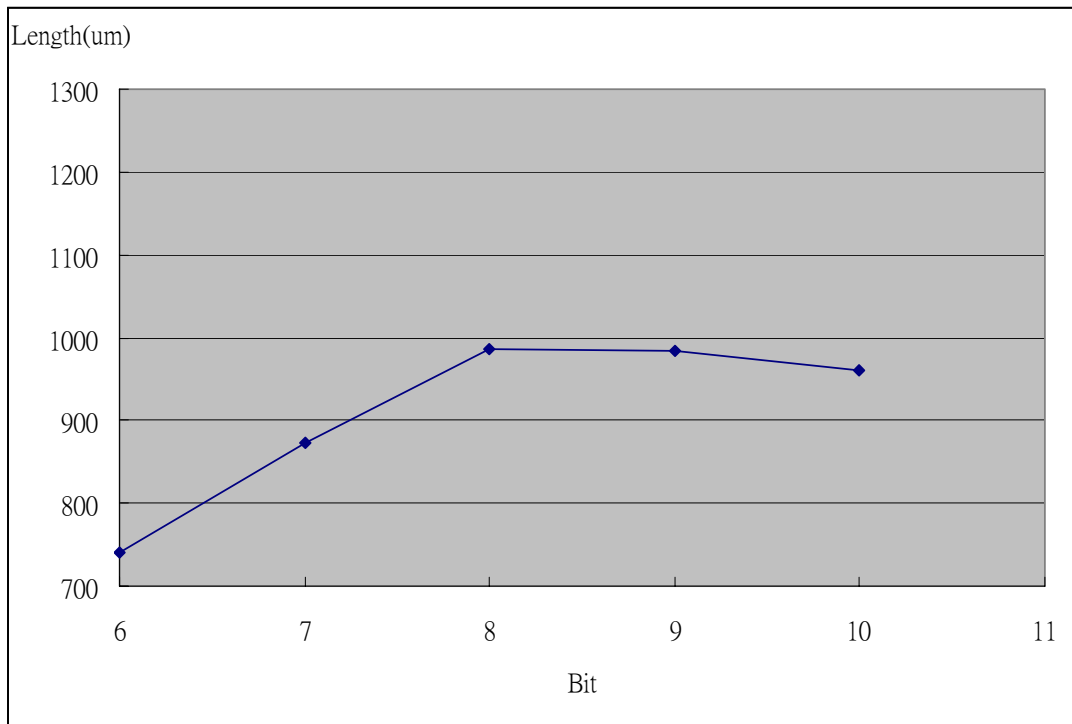


Figure 31: The maximum propagation length vs. different wire overheads for a 6-bit data bus.

In **Figure 32**, we demonstrate more simulation results of our flow. Assume the number of data bits varies from 3 to 10, we use the maximum signal propagation length estimation flow to find the maximum length with one bit wire overhead. In other words, for 3-bit data, we conduct our flow with a 3-bit bus and a 4-bit bus to find the maximum propagation length. The original propagation length in **Figure 32** represents the found maximum propagation length by using our flow without 1-bit wire overhead (i.e. for 3-bit data, the original propagation length represents the found maximum length on 3-bit bus). In addition, the increased propagation length stands for the increased length after adding one bit wire overhead by using our flow. As shown in **Figure 32**, our flow can increase about 25 percent of the original propagation length after adding one more bit as wire overhead. Hence, we can conclude that our flow can indeed increase the

maximum propagation length by using our encoding scheme.

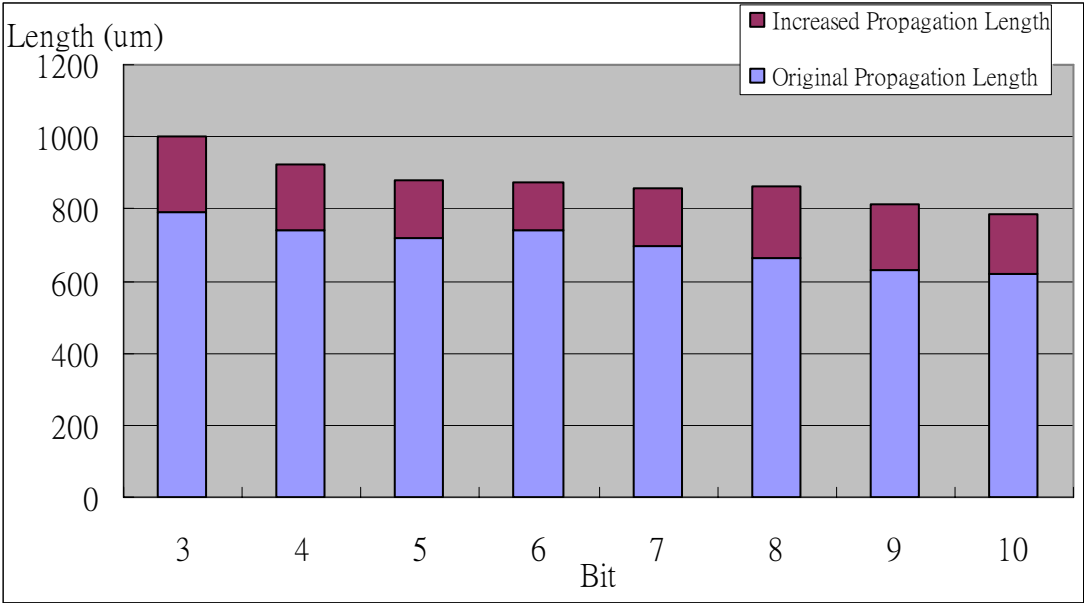


Figure 32: The maximum propagation length after adding one more bit as wire overhead.



Chapter 4

Conclusions



4.1 Bus Encoding for Reducing the Delay Time

In this work, we propose an effective on-chip bus encoding flow by considering *RLC* effects and is usable in any working frequency. With user-given design constraints and wire parameters, our encoding flow will generate a valid code set that meets all constraints. Since all the transitions between two input patterns in the valid code set meet the design constraint, designers can perform one to one mapping between valid code set and data patterns without restrictions. Simulation results show that our encoding method can significantly reduce the coupling delay of a bus with given delay constraint and parameters. In addition, by using superposition theorem, our method can significantly reduce the run time.

4.2 Bus Encoding to Lengthen Signal Propagation

We also propose a maximum signal propagation length estimation flow that can increase the propagation length on buses with the user-given constraints and bus parameters. Our flow combines a proposed bus encoding scheme and a curve fitting method. The proposed bus encoding scheme can effectively reduce the *LC* coupling effects on on-chip buses, and hence, improve the worst-case switching delay. In addition, the signal propagation length can be efficiently predicted by the proposed curve fitting method. Therefore, our flow can produce a valid code set of a bus structure that achieves the maximum signal propagation length under the given constraints with reasonable runtime. The simulation results show that the proposed flow significantly increase the propagation length.

Since this work can estimate the propagation length of a bus, we believe that it is feasible to be integrated into a floorplan or routing tool. Furthermore, we intend to combine our work with some layout techniques such as the shield insertion or the buffer insertion. Therefore, the maximum signal propagation length of a bus could possibly be further increased by a proper mixture of these techniques.

Reference

- [1] Semiconductor Industry Association, International Technology Roadmap for Semiconductors, 2003.
- [2] Mohamed A. Elgamel and Magdy A. Bayoumi, "Interconnect Noise Analysis and Optimization in Deep Submicron Technology," *IEEE circuits and Systems Magazine*, pp.1540-7977, March, 2003.
- [3] Joon-Seo Yim and Chong-Min Kyung, "Reducing Cross-Coupling among Interconnect Wires in Deep-Submicron Data path Design," *Proceedings of IEEE Design Automation Conference*, pp. 21-25 June 1999.
- [4] Shang-Wei Tu; Jing-Yang Jou; Yao-Wen Chang, "RLC Effects on Worst-Case Switching Pattern for On-Chip Buses," *IEEE International Symposium on Circuits and System*, 2004.
- [5] Mohamed A. Elgamel, Ashok Kumar, and Magdy A. Bayoumi, "Efficient Shield Insertion for Inductive Noise Reduction in Nanometer Technologies," *IEEE Transaction Very Large Scale Integration Systems*, vol. 13, issue 3, pp. 401-405, Mar. 2005.
- [6] Yu Cao, Chenming Hu, Xuejue Huang, Andrew B. Kahng, Sudhakar Muddu, Dirk Stroobandt, and Dennis Sylvester, "Effects of Global Interconnect Optimizations on Performance Estimation of Deep Submicron Design," *IEEE/ACM International Conference on Computer Aided Design*, pp. 56 – 61, Nov. 2000.

- [7] Kevin M. Lepak, Irwan Luwandi, and Lei He, "Simultaneous Shield Insertion and Net Ordering under Explicit RLC Noise Constraint," *Proceedings of Design Automation conference*, pp. 199-202, June 2001.
- [8] Magdy A. El-Moursy and Eby G. Friedman, "Optimum Wire Sizing of RLC Interconnect with Repeaters," *Proceedings of the IEEE Great Lakes Symposium on VLSI*, pp. 27-32, April 2003.
- [9] Yehea I. Ismail, Eby G. Friedman, and Jose L. Neves, "Repeater Insertion in Tree Structured Inductive Interconnect," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 48, issue 5, pp. 471-481, May 2001.
- [10] Bret Victor and Kurt Keutzer, "Bus Encoding to Prevent Crosstalk Delay," *International Conference on Computer Aided Design*, pp. 57-63, Nov. 2001.
- [11] Kwang-Hyun Baek; Ki-Wook Kim; and Sung-Mo Kang, "A Low Energy Encoding Technique for Reduction of Coupling Effects in SOC Interconnects," *Proc. of the 43rd IEEE Midwest Symposium on Circuits and Systems*, vol. 1, pp. 80-83, Aug. 2000.
- [12] Paul P. Sotiriadis and Anantha Chandrakasan, "Reducing Bus Delay in Submicron Technology Using Coding," *Proc. of the Asia and South Pacific Design Automation Conference*, pp. 109-114, Feb. 2001.



- [13] Harish Kriplani, Farid Najm, and Ibrahim Hajj, "Pattern Independent Maximum Current Estimation in Power and Ground Buses of CMOS VLSI Circuits: Algorithms, Signal Correlations, and Their Resolution," *IEEE Trans. Computer-Aided Design of Integrated Circuits Syst.*, pp. 998-1012, 1995.
- [14] L. O. Chua, C. A. Desoer, and E. S. Kuh, "Linear and Nonlinear Circuits", McGraw-Hill Inc, 1987.
- [15] Keith Nabors and Jacob White, "FastCap: A Multipole Accelerated 3-D Capacitance Extraction Program," *IEEE Trans. Computer-Aided Design*, vol. 10, No. 11, pp. 1447-1459, Nov. 1991.
- [16] M. Kamon, M. J. Tsuk, and J. K. White, "FastHenry: a Multipole-accelerated 3D Inductance Extraction Program," *IEEE Trans. Computer-Aided Design*, pp. 1750-1758, Sept. 1994.
- [17] Doug Matzke, "Will Physical Scalability Sabotage Performance Gain?" *IEEE Computer*, pp. 37-39, Sept. 1997.
- [18] Ankireddy Nalamalpu, Sriram Srinivasan, and Wayne P. Burleson, "Boosters for Driving Long On-chip Interconnects: Design Issues, Interconnect Synthesis and Comparison with Repeaters," *IEEE Transaction on Computer-Aided Design*, vol. 21, pp. 50-62, Jan. 2002.

- [19] Hui Zhang, Varghese George, and Jan M. Rabaey, "Low-swing On-chip Signaling Techniques: Effectiveness and Robustness," *IEEE Transaction Very Large Scale Integration Systems*, vol. 8, pp. 264-272, June 2000.
- [20] Atul Maheshwari and Wayne Burleson, "Differential Current-sensing for On-chip Interconnects," *IEEE Transaction Very Large Scale Integration Systems*, vol. 12, pp. 1321-1329, Dec. 2004.
- [21] Yehea I. Ismail and Eby G. Friedman, "Effects of Inductance on the Propagation Delay and Repeater Insertion in VLSI Circuits: a Summary," *IEEE Circuits and Systems Magazine*, 2003.



Vita

Jiun-Sheng Huang was born in Taitung on December 29, 1980. He received the M.S. degree in Electrical Engineering from National Chiao Tung University in June 2003 and entered the Institute of Electronics, National Chiao Tung University in September 2003. His major studies were Electronics Design Automation (EDA) and VLSI design. He received the M.S. degree from NCTU in June 2005.

