行政院國家科學委員會專題研究計畫　期中進度報告

硬體描述語言低功率架構設計(2/3)

計畫類別：個別型計畫
計畫編號：NSC91-2213-E-002-038-
執行期間：91 年 08 月 01 日至 92 年 07 月 31 日
執行單位：國立臺灣大學電機工程學系暨研究所

計畫主持人：賴飛羆

計畫參與人員：蔡坤霖

報告類型：精簡報告
報告附件：出席國際會議研究心得報告及發表論文
處理方式：本計畫可公開查詢

中　華　民　國 92 年 5 月 28 日

行政院國家科學委員會專題研究計畫期中報告

硬體描述語言低功率架構設計(2/3)

Low power design in hardware description language

計畫編號：NSC 90-2213-E-002-038-

執行期限：91 年 8 月 1 日至 92 年 7 月 31 日

主持人：賴飛羆　國立台灣大學電機工程系

計畫參與人員：蔡坤霖　國立台灣大學電機工程系

一、中文摘要

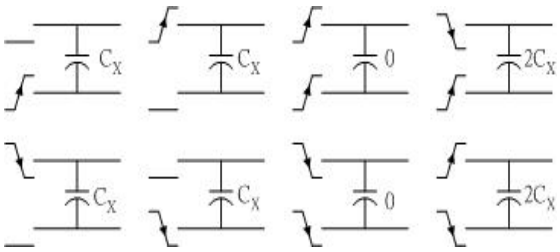
隨著奈米製程的的開發，單位面積所必須容納的邏輯閘數大幅增加，在要求效能的情形下，晶片工作頻率亦不斷向上攀升。然而面臨到的問題卻是在單位面積所呈現的功率消耗極度龐大，因而造成了散熱與系統穩定性等問題。若晶片使用於需電池供應電源的設備上，則更直接關係到操作時間的長短。大多數在電路上的低功率研究都著眼於邏輯區塊的最佳化。然而，由於傳輸資料大量運作在晶片內部的匯流排上，使得匯流排的功率消耗在整體晶片的功率比率上一直居高不下。在匯流排的問題中，由於鄰近的兩條傳輸線會有互相干擾的情形，使得傳輸的資料發生錯誤，造成訊號必須重新傳輸，使功率消耗及時間延遲上都增加了額外的負擔。因此，本計畫便針對此問題，提出了一個可以有效解決匯流排上因傳輸線太過靠近而造成的相互干擾的方法。此方法是基於事先研究資料傳輸的轉態行為特性，再依照所得到的訊息，透過一個包含編碼、排列、與反置的架構來最佳化功率消耗及雜訊干擾。在所提出來的方法中，我們改進了以往必須使用大量額外控制電路的缺點，以較低的延遲及較小的面積負擔來進行控制。依照研究出的成果，大約可以改進38.7%的相互干擾。

關鍵詞：低功率、匯流排、相互干擾、基因演算法

Abstract

With the development of nanometer, more and more gate counts need to be placed in one chip. Under the performance requirement, the working frequency of a chip also increases continually. The following problems are the power consumption of that chip, the heat problem and the reliability of the chip. It is also related to the operating time of the portable device, while using such chip on a battery operating system. Most low power researches on the circuits focus on optimization the logic blocks. However, there are lots of data need to be transmitted on the on-chip buses, and lead to the power consumption of buses have a high percentage among total power consumption. Among the bus problems, there exists a crosstalk

problem while two adjacent wires are too close. It will result in error when data transit, and also increase the loading of power consumption and transition delay. Thus, we proposed a method based on profiling the switching behavior to solve this problem efficiently. This method, based on the profiling information, applying an architecture that encodes pairs of bus wires, permutes the wires and assigns an inversion level to each wire together. Unlike the previous technique, it can reduce the control circuit with small delay and area overhead.



As the research results show, it can be improved about 38.7% of the crosstalk problem.

Keywords: low power, bus, crosstalk, genetic algorithm

二、緣由與目的

隨著可攜式裝置的普及化，如手機、手提式電腦、個人數位助理等等，這些

圖一:兩條鄰近的傳輸線轉態影響

裝置對我們日常生活的影響也越來越大。然而、在不插電的狀況下，如何延長這些裝置的使用時間是非常重要的考量點。更重要的是，在下一代的中央處理器中，由於時脈不斷的提升，高速運算消耗大量功率引發中央處理器過熱的現象，也嚴重影響了電腦系統的穩定性及效能。

在電路設計方面，系統單晶片上匯流排的設計對於效能或功率消耗一直佔有很大的影響。舉例而言，晶片匯流排的功率消耗在 Alpha 21064 與 Intel 80386 中便分別佔了 15%及 30%。因此如何減低晶片上匯流排的功率消耗及傳輸線之間的互相干擾影響便成了在設計過程中一個相當重要的考量。

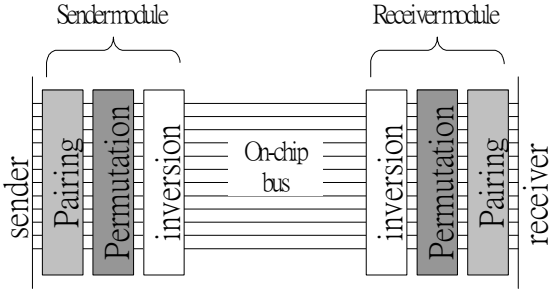
以往在匯流排功率消耗的研究上大都著重於如何利用編碼技術來降低功率消耗。然而，類似這樣的研究通常只考慮了單一條傳輸線的轉態次數，卻忽略了多條傳輸線之間的相互影響。因此，他們並沒有直接面對相互干擾的問題。更進一步地說，這些方法中有些只能適合位址匯流排，或者藉由增加匯流排的傳輸線來降低干擾，這對我們目前的設計上並不合用。當然也有靜態降低晶片上匯流排結構耦合功率的方法被提出來。然而，為了要編碼而產生的額外電路時常跟時序及面積條件相衝突，並且也無法跟已減少的干擾取得平衡。

在這一期的計畫中，我們利用基因演算法提出了一個讓匯流排功率及相互干擾問題最佳化的新方法。我們的方法包含了三個技術：配對、排列及反相。基於一個給定的應用來研究匯流排，我們首先採用一個特別的匯流排編碼技巧，我們稱之為配對。詳細地說，我們配對匯流排的傳輸線來減低匯流排活動。儘管匯流排行為並非與干擾有直接的關係，然而降低轉態活動卻也通常降低了干擾。我們的實驗顯示這樣的假設在我們大多數的測試案例中確實是可行的。接下來，我們

結合排列與反相的技術來獲得進一步的耦合功率降低。在排列的過程當中，為了要使具有相似短暫傳輸行為的訊號相鄰在一起，匯流排的傳輸線被重新安排。反相則靜態地決定每一條匯流排傳輸線在傳輸資料時使用正向或反向邏輯。本計畫的目的在於提出一個有效控制能量消耗的架構與方法。因此在本年度計畫中，我們針對匯流排架構提出一個簡單且有效的方法，利用三個程序來降低功率消耗及控制相互干擾問題。並將之運用於系統單晶片(SoC)上。

三、研究方法與演算法

在提出本計畫所研究的方法之前，我們先對匯流排上傳輸線互相干擾所產生的耦合電容問題做一個簡單的說明。考慮一對匯流排傳輸線，當這對傳輸線在傳送資料時，共有八種可能的型態，假設我們只針對在兩條傳輸線之間的動態轉變分佈，如圖一所示。在圖一中的左邊四種狀態是只有一條傳輸線改變狀態時，其耦合電容會被充電至 C_xV，而在圖一右邊的四種狀態則是當兩條傳輸線皆發生變化時所產生的耦合電容。當兩條傳輸線同時往某一電位轉換時，其耦合電容的充放電並不會產生任何變化，但若兩條傳輸線朝向不同電位轉態，其間的電容改變則為 $2C_xV$。



為了分析某一個給定的應用系統在匯流排上相互干擾的影響，我們必須事先研究其一般的傳輸順序，對於每對匯流排傳輸線，我們決定每對傳輸線在下個週期從 k 轉變為 l 的機率($k, l \in \{00, 01, 10, 11\}$)。更詳細地說，我們對於每對傳輸線 k,l 以定義了絕對機率 $p_{k,l}(i, j)$，例如：$p_{10,01}(i, j)$為傳輸對(i, j)從 10 到 01 的機率。

定義了傳輸機率之後，我們便可藉此建立一份傳輸線干擾表 $xtalk(i, i_p, j, j_p)$，其中 i_p, j_p 定義為傳輸線 i 與 j 的反轉層。如果反轉層的值為 1，則在傳輸線上的

圖二：降低干擾的架構

資料必須與原欲傳輸之邏輯值相反。使用了干擾表之後，我們便可決定一組匯流排中 n 條傳輸線所產生的干擾，如下：

$$crosstalk_{bus} = \sum_{i=0}^{n-2} xtalk(w_i, v_i, w_{i+1}, v_{i+1})$$

我們的目標是在一個給定的傳輸序列中使 $crosstalk_{bus}$ 最小化。為了達到這個目標，必須在匯流排的傳送端與接收端各加上一個模組。如圖二所示，此模組包含了三個部分：

- Pairing:為了減少匯流排轉態的動作次數，我們必須預先處理匯流排的資料。我們使用「位元配對」的方式來將傳輸線群組化成一群不交疊的傳輸線對。這些線對在稍後會經過重新編碼以降低訊號的轉態動作。
- Permutation:重新排列匯流排的順序使得相似且短暫的行為得以排列在一起。
- Inversion:更進一步地，一個反轉層被指定給每一個匯流排傳輸線，也就是說，在匯流排傳輸線上的資料也許會被轉變成與原資料不同的邏輯。

在這個方法中，若不要增加資料路徑額外的延遲，則 pairing 是可以被省略的。但 permutation 與 inversion 則會在電路合成時決定，無法在電路運作中進行改變。而這些方法通常不會增加任何重大的面積或延遲負擔。

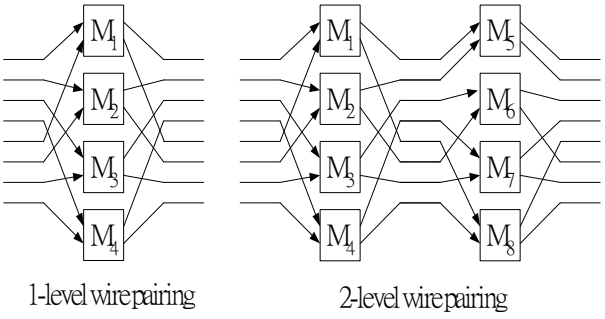
接下來，我們對這三個步驟做進一步說明：

(1) 配對(Pairing)：相互干擾的問題事實上與傳輸線本身的傳輸行為及鄰近的傳輸線動作有關。因此，若減少傳輸的動作亦能夠有效地減低相互干擾的影響。然而，必須注意的是，較少的傳輸動作，並非可以自動得到較好的干擾影響結果。

曾經有許多種方法都被使用來降低匯流排上的傳輸動作。他們有些是特別針對位址匯流排與增加資料路徑的延遲而設計，然而，更多的應用會限制這些在資料路徑上的延遲，以便能有效地控制延遲符合剛好的時間限制。因此，我們發展了一個減少匯流排傳輸行為的方法，我們稱之為「傳輸線配對」，其不止可以運用在各式各樣的匯流排上，還可以幫助設計者簡易地去控制資料路徑上的最大延遲。

更詳細地說，我們使用了如圖三所顯示的編碼架構。圖三左側為 1 層編碼架構。它使用編碼區塊 M_i 來配對匯流排上的訊號傳輸線。也就是說，輸入的訊號線被分群成一對一對，再將每一對送進編碼區塊進行編

碼。每一個編碼區塊依照表一中的對應值從 T_0 到 T_5 將兩條輸入線編碼後輸出



圖三：信號傳輸線配對

兩條傳輸線。我們可以容易地觀察出，合法的編碼數目共有 24 個。而表一只顯示了其中 6 個，剩餘的 18 組可以藉由反轉一個或兩個輸出欄位獲得。然而，反轉會在稍後討論，目前我們只考慮這最基本的六種編碼方式。

如同每個編碼表將每對位元對應到輸出位元對一樣，對應的函數可以使用兩層的 2 輸入邏輯閘所組合而成。因此，在最糟的情形下，兩層邏輯閘的延遲會被加進整體資料路徑中。而相同的情形也發生在當已經編碼過的資料在接收端必須被還原成原始資料時所產生的延遲。

為了在使用傳輸線配對後的匯流排動作能夠盡可能的少，我們必須去解決傳輸線配對的問題：

- 將傳輸線分群成一些配對的集合 $M=(M_1, \ldots, M_{n/2})$，其中 $M_i=(w_x, w_y)$，每個傳輸線 w_x 只能夠出現在 M 中一次，以及
- 找出一個對應 $EM: M \rightarrow T$，其中 $T=\{T_1, \ldots, T_6\}$，每個編碼過後的 T_i 會分配給每對在 M 中的傳輸線對，

使得一個給定的傳輸序列能夠將匯流排的動作最小化，也就是說盡可能地去減少傳輸的總量。

基於一個從研究匯流排動作的機率資訊，我們可以應用一個基因演算法來解決匯流排的配對問題。為了更進一步地減少匯流排動作，我們同樣考慮一個多層次的方法，其中，匯流排配對的方式被重複使用了好幾次。

Input	Output					
	T0	T1	T2	T3	T4	T5
00	00	01	10	11	00	01
01	01	00	01	01	11	11
10	10	10	00	10	01	10
11	11	11	11	00	10	00

在圖三的右側顯示了一個使用

2 層配對架構的例子。然而，在最糟糕的情形之下，每一層的配對都會造成兩個邏輯閘的延遲。

(2) 排列與反轉：對於一個有 n 條傳輸線的匯流排而言，要去找出排列與反轉的規則，可以描述成去找出一個

- 傳輸線順序 $bus_{opt}=(w_0, w_1, \ldots, w_{n-1})$，其中 $w_i \in \{0, 1, \ldots, n-1\}$，$w_i \neq w_j$，與一個
- 反轉向量 $v_{opt}=(v_0, v_1, \ldots, v_{n-1})$，其中 $v_i \in \{0, 1\}$

使得

$$crosstalk_{bus} = \sum_{i=0}^{n-2} xtalk(w_i, v_i, w_{i+1}, v_{i+1})$$

為最小。

由於匯流排排列的問題為一個 *NP-hard* 的問題，因此，我們使用了一個近似的方法來找出可行的解法。我們所使用的方式便是基因演算法。

表二：與原來匯流排結構比較之干擾降低百分比

	instruction-cache address bus						data-cache address bus						instruction bus					
program	w/o inv.	w inv.	1 level	2 level	3 level	4 level	w/o inv	w inv.	1 level	2 level	3 level	4 level	w/o inv	w inv	1 level	2 level	3 level	4 level
gzip	39.39	42.42	57.57	57.57	63.63	60.60	16.42	17.69	32.40	40.32	43.23	42.82	21.56	28.58	25.57	29.70	32.16	36.02
vpr	29.71	32.20	45.12	48.27	50.63	52.11	30.20	36.85	39.83	39.65	41.17	41.12	17.99	21.25	16.62	18.58	21.46	25.55
gcc	27.89	28.67	43.23	47.08	48.06	48.97	20.66	22.75	24.13	26.49	25.70	25.55	20.41	23.84	19.52	21.20	22.88	22.77
mcf	40.00	45.00	60.00	60.00	60.00	60.00	33.33	33.33	51.11	55.56	58.06	57.80	16.47	19.18	15.83	14.75	18.74	17.82
parser	29.49	31.42	46.60	50.95	52.79	53.53	29.94	33.20	30.45	29.50	29.63	30.29	21.30	25.46	18.79	21.60	26.57	26.07
perlbmk	27.24	27.59	42.03	46.27	47.93	48.15	20.81	25.42	25.05	25.10	23.95	24.48	17.67	20.66	15.03	17.35	16.56	19.41
vortex	29.59	30.53	45.62	49.72	51.29	51.93	14.85	18.97	21.43	26.88	27.41	29.48	20.21	27.43	22.80	25.61	27.71	28.88
bzip2	30.15	32.90	47.13	51.38	53.07	54.14	41.60	46.09	43.11	49.98	51.62	54.69	21.08	23.57	19.59	25.63	28.17	29.46
mesa	28.10	29.29	45.57	49.58	50.42	52.16	35.05	39.91	40.04	44.50	45.60	46.91	16.57	20.88	12.88	12.54	12.77	12.90
art	32.87	34.83	50.72	54.54	54.40	56.29	59.32	61.26	58.72	58.44	59.08	59.71	38.18	47.46	41.41	47.41	53.57	54.11
equake	29.42	31.38	45.42	49.49	50.42	51.27	31.60	39.55	37.01	41.66	43.86	47.23	19.71	18.36	16.12	15.38	16.63	18.11
ammp	31.67	33.19	46.32	49.53	51.33	51.89	39.91	43.45	44.06	43.21	44.36	44.20	36.96	40.46	31.96	33.82	38.80	43.24
average	31.29	33.29	47.94	51.20	52.83	53.42	31.14	34.87	37.28	40.11	41.14	42.02	22.34	26.43	21.34	23.63	26.34	27.86

對於一個給定的匯流排排列 $w=(w_0, w_1, \ldots, w_{n-1})$以及兩個反轉向量 $v=(v_0, \ldots, v_{n-1})$和 $v'=(v_0, \ldots v_m, v_{m+1}, \ldots, v_{n-1})$，對於整體匯流排的干擾會減低或增加，決定於將換傳輸線 m 與 $m+1$。因此，對於一個給定的排列，其最佳反轉向量可以藉由使用 $m=0, 1, \ldots, n-1$ 與反轉所有的 v_i 數次而得

到。

四、成果與討論

本計畫中，基因演算法利用 C++語言在 Linux 平台上進行實現，並利用 SimpleScalar PISA 處理器的模擬器取得輸入資料，包含指令－快取位址匯流排、資料－快取位址匯流排及指令匯流排等。而資料取得都是藉由執行 SPEC2000 測試程式為主。

表二顯示了在三種匯流排型態中所得到干擾降低的結果。當只有使用排列及反轉時，在指令快取位址匯流排得到約 33%的改善，在資料快取為 35%，在指令匯流排則為 22%。

為了分析匯流排動作對於干擾降低的影響，我們在使用排列及反轉技術之前使用了配對方式。結果顯示在「1 level」到「4 level」的欄位，level 的數字表示被使用的配對層級。而結果也如我們所預期的，配對的確有效改善位址匯流排的干擾問題。但對於指令匯流排的影響卻不盡理想這是由於存在太多隨意的指令匯流排順序。

從結果中我們可以獲得一個結論：對於指令快取位址匯流排來說，至少一層的配對是相當有效的，可以降低干擾達 16%以上。而增加更多層次的配對只是對於效能有部分改進而已。這對於位址匯流排來說更是事實。然而對於指令匯流排則在四層配對之後才會有較好的結果。因此四層配對結合重新排列與傳輸線反轉對於指令匯流排來說會是一個比較好的選擇。這樣的結果是相當有趣的，因為當程式要在處理器上執行之前，指令編碼能夠在指令編譯時完成。也就是說延遲所造成的負擔並不會在這個方法裡出現，卻能夠有效地降低干擾。

在本年度的計畫中，我們針對電路中較為耗電的匯流排提出了一個可以改進功率消耗及雜訊的方法。在這個方法當中，藉由對匯流排傳輸線的配對編碼、決定傳輸線的順序以及對每條匯流排傳輸線的反轉層設計來降低因互相干擾所產生的問題。跟以往不同的地方是，在我們所提出來的方法中，對於延遲與面積多了一項評估的重點。如果延遲是主要的考量，則步驟中的配對可以被省略。但如果主要的考量是互相干擾的問題，則多層次的配對方式便可有效地去控制干擾。

在計畫進度方面，我們將一般電路中最耗電的部分進行討論與改進，這將有助於我們對硬體描述語言低功率架構設計進行更進一步的研究。未來，只需將此一方法與結果運用在硬體描述語言，便可針對硬體描述語言所設計出之電路進行

匯流排重新設計，以便能達到更佳的省電的效果，以及減少干擾所產生之問題。這可以幫助使用硬體描述語言的晶片設計者更快速且有效的達到低功率的目的。

五、成果發表

1. E. Naroska, S.-J Ruan, F. Lai, U. Schwiegelshohn, L.-C. Liu, “On Optimizing Power and Crosstalk for Bus Coupling Capacitance Using Genetic Algorithms,” *Proceedings of IEEE International Symposium on Circuits and Systems, May, 2003.*

六、參考文獻

[1] S. Ramprasad, N. R. Shanbhag, and I. N. Hajj, “A Coding Framework for Low-Power Address and Data Busses,” *IEEE Trans. on VLSI Systems, June, 1998.*

[2] L. Benini, G. D. Micheli, E. Macii, M. Poncino, and S. Quer, “Powe r Optimization of Core-Based Systems by Address Bus Encoding,” *IE EE Trans. on VLSI Systems, vol. 6, pp. 551-562, Dec, 1998.*

[3] L. Benini, G. D. Micheli, E. Macii, D. Sciuto, and C. Silvano, “Address bus encoding techniques for system-level power optimization,” *in Proceedings EuroDAC-97, pp. 861-867, Nov, 1997.*

[4] Y. Shin, S. Chae, and K. Choi, “Partil Bus-Invert Coding for Power Optimization of Application-Specific Systems,” *IEEE Trans. on VLSI Systems, vol. 9, pp. 377-383, Apr, 2001.*

[5] E. Musoll, T. Lang, and J. Cortadella, “Working-zone encoding for reducing the energy in microprocessor address buses,” *IEEE Trans. on VLSI Systems, vol. 6, pp. 568-572, Dec. 1998.*

[6] B. Victor and K. Keutzer, “Bus Encoding to Prevent Crosstalk Delay,” *in Proceedings ICCAD-2001, Nov. 2001.*

[7] P.-P. Sotiriadis and A. Chandrakasan, “Bus Energy Minimization by Transition Pattern Coding (TPC) in Deep Submicron Technologies,” *in in Proceedings ICCAD-2000, pp. 322–327, Nov. 2000.*

[8] K.-W. Kim, K.-H. Baek, N. R. Shanbhag, C. L. Liu, and S. Kang,

"Coupling-driven signal encoding scheme for lowpower interface-desig n," *in Proceedings ICCAD-2000, pp. 318–321, Nov. 2000.*

[9] H. Zhou and D. Wong, "Global Routing with Crosstalk Constraints," *in Proceedings DAC-1998, June 1998.*

[10] I. H.-R. Jiang, Y.-W. Chang, and J.-Y. Jou, "Crosstalk-Driven Inte rconnect optimization by Simultaneous Gate and Wire Sizing," *IEEE Trans. on CAD of Integrated Circuits and Systems, vol. 19, pp. 999–1 010, Sep. 2000.*

[11] L. Macchiarulo, E. Macii, and M. Poncino, "Low-Energy Encoding for Deep-Submicron Address Busses," *in Proceedings ISPLED-01, pp. 176–181, Aug. 2001.*

[12] D. Burger and T. M. Austin, "The SimpleScalar Tool Set, Version 2.0." *http://www.simplescalar.com*, 1997.