逢　　甲　　大　　學

資訊工程學系專題報告

DMA DESIGNED BY VHDL

學生:林俊緯

指導教授:陳德生 教授

中華民國九十三年四月

目錄

圖表目錄

摘要

第一章【導論】，首先介紹為什麼專題要做”DMA Design BY VHDL”的動機，並說明研究目的。

第二章是對PLD、CPLD、FPGA、ASIC等作相關的研究探討。

第三章是針對本專題中使用的硬體電路語言—VHDL,及使用的工具—MaxPlus II做一介紹。

第四章、第五章則是對DMA作一簡介以及對其內部架構、規格、作業週期、傳輸模式等作說明。

第六章為DMA Controller設計與實做的方法及機制。

第七章是在專題製作期間的心得與感想。

第一章　　導論

1.1 前言 & 動機

近年來，數位邏輯電路由於半導體微電子積體電路技術之進步快速，使得上萬個電晶體可以集成在一小片的矽晶片上。積體電路的技術從 LSI(Large Scale Intergrated Circuits)，約數千個 gate，進步到超大型積體電路(VLSI，Very Large Scale Intergrated Circuits)，在單晶片上的電晶體數目已可達百萬個以上。而 TTL 邏輯足 IC 是屬於 MSI 的範疇，它提供標準的邏輯元件以供設計者使用。在設計簡單的邏輯電路時，非常的方便，此常見於一般的電路系統上，但當所設計的邏輯電路月來越龐大時，如果再適用 TTL 邏輯足 IC 就會有下列的缺點：

1. 電路板的面積很大，因此系統與電路板的成本，相對的提高了很多。
2. 電路板上的 IC 元件數相當多，要處理的電池干擾問題(EMI)，相對的複雜且困難，不易控制。
3. 所使用的 IC 元件數相當多，使得設計上的整合性降低，教部亦得的一個可靠的設計。

拜微電子記體電路技術快速發展之賜，使得單一元件邏輯電路成為可能。例如微處理機(Micro-Processor)、微控制器(Micro-Controller)、ASIC、PLD、FPGA 等的設計。這些可編程邏輯元件提供使用者可以針對自己設計上的需要，選用一種積體電路 IC，來使自己的邏輯電路能工作在最佳的狀況下，使成本控制在合理的價位上。這些超大型積體電路解決了 TTL 邏輯族在設計複雜邏輯電路上的問題。而且一些可程式的 PLD 或 FPGA 等，由於其具有可程式規劃的特點，用來做設計的 Protpye 驗證非常有用。再者由於使用特用積體電路所設計的電路系統具有保密的優點，因此仿冒者想要去方茂整個系統也就較為困難，具有保護智慧財產權的功能。

在數位機體電路的設計方法上，近代以來也有了格性的變革，在早期的電路較為簡單，且電路合成的工具較不完整，所以早期都是使用 gate-level 的方式，用圖形介面，就所需的邏輯閘、正反器，一個一個的畫成電路圖。此方式在設計超大型積體電路上，也碰上了瓶頸。主要是因為：

1. 使用 Gate-Level Design 曠日費時，需花相當長的時間在做設計之輸入。
2. 由於邏輯閘數相當多，會增加出錯的機會及將來維護上的困難。

為了解決這樣的問題，於是整個設計的趨勢轉向使用語言的方式，讓電腦來自動依所寫的硬體描述語言，合成其所對應的功能電路，並作電路的最佳化合成，如此可將設計的時程大幅度減短，並增加了系統設計的可維護性，稱之為高階設計自動化。即是利用 VHDL 或 Verilog 等高階語言設計所需要的功能，再利用邏輯合成工具，(Synthesis Tools)將其轉為線路圖，此類的可合成電路描述語言，一般稱為**高階硬體電路描述語言**，目前常用的有 VHDL 及 Verilog 等。而我這個專題主要便是用 VHDL 來做電路的設計。

現今，VHDL 已經成為業界發展數位電路、系統的標準。VHDL 的最大優點就是可以藉由高階語言的行為描述方式來設計複雜的電路，只要再經過 Synthesis Tools 就可以自動產生 Gate Level 的電路圖，可以大幅提升系統發展的速度及競爭力。

利用硬體電路描述語言來設計硬體似乎已成為了一種趨勢，於是我這個專題便選擇用 VHDL 來設計一個 DMA Controller。

1.2 研究目的

在這次的專題製作中，希望能夠利用 VHDL 硬體電路描述語言，以高階語言的方式，設計一個 DMA Controller，主要的目的在於：

1. 了解何謂硬體電路描述語言。
2. 如何利用 VHDL 來設計電路。
3. 學習開發工具的使用(本專題中是使用 Altera 公司的 MaxPlus II 軟體)。
4. DMA Controller 中的架構為何，是由那些單體所組成，以及每個單體的特性。
5. 希望藉由 VHDL 電路描述語言在設計硬體時，能更進一步了解其功用、設計原理及動作。

第二章 相關研究探討

2.1 PLD 簡介

可程式邏輯元件 PLD(Programmable Logic Device)，為 IC 設計的材料之一，以可程式 AND 閘陣列與固定 OR 閘陣列所組成的邏輯陣列為設計內容，可程式邏輯元件 PLD 依架構及密度大小可略分為以下三類：

1. PAL、GAL
2. CPLD、HCPLD
3. FPGA

目前市場的狀況 PAL 及 GAL 應用已漸漸減少，CPLD、HCPLD、EPLD 及 FPGA 則為應用的主流商品。

‧市場發展

PLD 的應用產品從 1975 年 Signetics 的 82S100 開始至今，已發展了 20 多年，從 1980 年代末期開始，因 CMOS PLD 的普及、單價成本減少，市場因而獲的更大的成長，由於技術陸續出現突破性的發展，因此比 PLD 有更高密度 Logic Cell 的 Complex PLD(CPLD)，在 90 年之後也被發表而成為市場應用的主流，20 年來整個市場的發展一直維持在相當穩定的成長，直到 96 年全球 PLD 約有 20 億美元的市場 。

另外，在 1985 年也出現以 SRAM 為應用 Base，使用 Anti Fuse 技術的 Field Programmable Gate Array(FPGA)，與 PLD 一樣，使用者可利用 Programming 的方式應用。在這幾年之間，PLD 及 FPGA Logic Cell 的密度不斷的往上成長，1992 年市場上大多僅在 10K~20K Gate 之間，到了 97 年單一元件便到達了 100KGATE 的規模。由於相信往後對 PLD 及 FPGA 市場規模發展有著極高的可期待性，在 80 年代末陸陸續續出現了許多在這相關領域發展的業者，在 1992 年時，全球可供應 PLD 及 FPGA 產品的業者便多達 30 家，之後 TI、 NS 及 Intel 等大公司因改變經營策略的考量，前後的退出此一市場，而較專業的小公司，因在市場上出現共同利益因素及發展策略，而相互的合併或收購。

目前全球 PLD 以及 FPGA 的供應商數量只剩不到當初的一半，因為這樣的市場發展，現在 Xnlinx、Altera 與從 AMD 分出來的子公司 Vantis，已成為 PLD 及 FPGA 市場最主要的供應商之一，據估計這三家業者的出貨量，總共幾乎佔了全球 PLD 及 FPGA 市場的 70%，目前這樣的市場結構下，新加入者相信是不太容易加入而從中獲得太多的市場佔有率。

- **未來發展的趨勢**

PLD 市場因利基點不斷出現的狀況下，競爭越來越激烈，越來越

好用的設計工具也就不段的推陳出新，近來不斷出現的新商品，有些共同的發展趨勢，即為：

1. 速度越來越快
2. Gate Count 越來越大，功能也就越來越強。
3. 越來越容易使用，與以往推出產品的相容性越來越好，因配合使用的 EDA 設計工具環境日漸成熟。

2.2 CPLD & FPGA

2.2.1 CPLD (Complex Programmable Logical Devices)

所謂 CPLD 即複雜之可程式化邏輯裝置，其主要是將 PLD 概念延伸、擴充、提高大階層之積體電路以改善系統之使用執行性，同時使用較少之晶圓空間而改善了可靠度並降低成本，不把 PLD 加大但增加輸入端、乘績項及微元件數。一個 CPLD 含有多個邏輯元件方塊(PAL Block)，每一個 PAL Block 如同一個小的 PLD 如 GAL16V8 等，而每個邏輯方塊間之接線昰利用可規劃之連接器作訊號繞線，此種架構安排技巧使其更有效的應用在可用之矽晶圓區間，因而導致有更好的使用執行性並降低成本。

・基本架構及特性

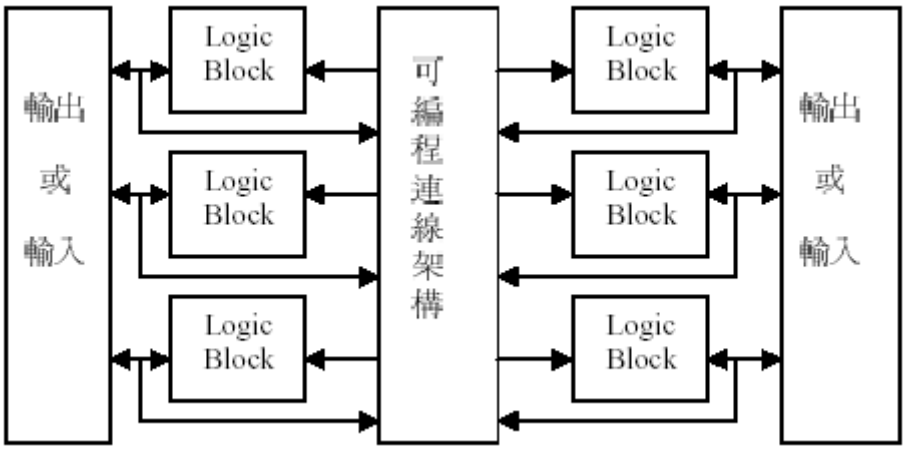

圖 2-1 CPLD 基本架構圖

CPLD 的基本架構(如上圖)，其特性為：

1. 由多個 PAL 及結而成。

2. 可使用的邏輯閘數為 500~5000+

3. 基本單位架構 PLD：輸入接腳為 15~35 個輸入、正反器(Flip Flops)為 2~4 個、輸出接腳為 2~8 個輸出。

4. 內部連結資源為 EPROM 或 EEPROM。

5. 在 PAL 區塊內**可固定時間延遲**。

6. PAL 區塊之間的相連接將增加時間延遲。

2.2.2 FPGA

FPGA 的全名為 Field Programmable Gate Array，它是由許多個邏輯元件(Logic Cell)，經由可程式的垂直通道及水平通道的連線所構成，其基本架構如下圖：

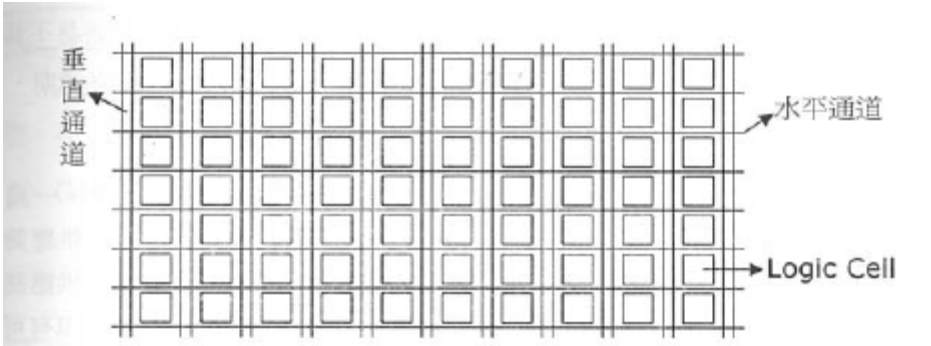

圖 2-2 FPGA 基本架構圖

‧ FPGA 之特性

FPGA 的基本架構如 圖 2-2，其特性為：

1. 基本單位稱為邏輯細胞(logic cell)。
2. 可使用的邏輯閘數為：1000~50000+(Usable Gate Count)。
3. 可使用的輸入/輸出腳數為：60~300+。
4. 邏輯細胞架構：輸入腳數為：4~16、正反器(Flip Flops)：1~2、輸出接腳為：2~50。
5. 內部連接資源(Interconnect)為 SRAM 或是 Anti-Fuse(反熔絲)。
5. 影響速度的因素有：內部連接資源的多寡(Interconnect)、邏輯的佈製(Place)、繞線的路徑(Route)、路徑的負載(Fan out)。

FPGA 的架構與半導體製程中的 Gate Array 方式類似，所不同點只有兩個地方：

1. 性能的需求：

   越來越多的應用場合需求操作在更高的頻率，如此將使 FPGA 之操作頻率深受挑戰。

2. 密度或閘數的需求：越來越多的應用場合，希望將所有的邏輯放在單一晶片內，此稱為 System in Chip，因此高密度的 FPGA 與高閘數的 FPGA 的市場要求也越來越殷切。系統高度整合的結果，將使得產品的成本為之降低。

3. ISP 及 ICR 的功能需求：

ISP 是 In System Programmability 的縮寫，而 ICR 是 In Circuit Reprogrammability 的縮寫。ISP 與 ICR 的設計，使得設計者能在系統上來編程所想要的邏輯電路，也使得產品在銷售之後，可以隨時在顧客的手上來作昇級(Upgrade)。

4. 設計工具的方便性：

方便與實用的設計工具(Design Tools)可以縮短產品開發之週期，使得產品能夠快速的打入市場，增加產品之利潤。

2.2.3 CPLD 與 FPGA 之比較

一般人對 CPLD 和 FPGA 的特性，較難區分其差異，所以先在此作一個簡單的描述。

- CPLD 的優點有：
  1. 較可預測延遲時間。
  2. 較少開發工具的投資。
  3. 在 State Machine&Address 上有較好的應用。

- CPLD 的缺點有：
  1. 限制設計上的彈性運用。
  2. 邏輯閘使用率較差(特別在 Data Path)。
  3. 高消耗功率。
  4. 較大的 macro cell 架構限制了正反器的數目。

- FPGA 的優點有：
  1. 有較高層級的彈性設計。
  2. 較多的正反器。
  3. 適合大部分的設計應用(State Machine & Decoder 除外)。

4. 較低的消耗功率。

- FPGA 的缺點有：

1. 需要先執行 Place & Route 程序。

2. 無法正確預測延遲時間(不同的邏輯佈置、繞線路徑、及負載將有相當大的差異結果)。

3. 需要投資一筆開發工具的費用(開發軟體及燒錄器)。

4. 需使用軟體模擬器作功能測試。

2.3 ASIC 簡介

ASIC 乃 Application Specific Integrated Circuits 的縮寫中文叫做特用積體電路或專用積體電路。顧名思義，此積體電路設設計的需求，即是提供一個在特殊應用場合上所使用的積體電路元件。例如許多電腦週邊設備之產品(如硬碟機、掃瞄器等)。在其電路板上均可發現有一顆各家所特有的 ASIC。此 ASIC 的目的，一方面可使系統之電路整合更有效率，並使成本下降，提升產品的競爭力，另一方面，又可以使電路之設計增加保密性，使得設計不易被盜拷。

ASIC 在半導體的製程上，屬於邏輯類的製程，有別於一般記憶體的製程。在 ASIC 電路製程與製作方式上，可分為下列四種方式：

1. Gate Array：

Gate Array 的 ASIC 製作方式，乃是由積體電路廠商，提供已部分完成之電晶體佈局，由應用者根據此母體，來加上數曾光罩(通常為三層到四層)，來構成各個電晶體之間的連線關係，以達成所需的邏輯電路設計需求。此種設計方式的特點為更動之光罩數少，故 NRE 費用較少，製作之時程也相對較短，但因受限於母體之限制故單價成本較高，設計的整合亦較低。

2. Standaed Cell：

Standard Cell 的製作方式，是以積體電路廠商所擁有的標準 Cell 為基礎，將所設計的邏輯電路，由這些 cell 來合成。因其結構之可變性加大，故其所需設計之光罩層數，較 Gate Array 的方式為多(約 13 層以上)。Standard Cell 的設計製作方式，可使單價之成本降低(因其整合性較高)，但相對地要付出較高的 NRE 費用及較長的製作時程。通常在設計中，如有整合入 RAM 或 ROM 在 ASIC 中時，即需要使用 Standard Cell 的方式來降低電路所要佔用的晶圓面積。

3. Cell Base Array(CBA)：

CBA 的製作方式，乃綜合 Gate Array 及 Standard Cell 的特點所構成。Cell Base Array 的製作方式，將設計中的邏輯電路中的不同模組，均以 Cell 的方式來建立，如此則要改變某一個模組中的電路設計時，只需改變那一個相對應 cell 的光罩即可。所以以 CBA 的方式來製作 ASIC 時，可將一些固定不變的 cell，以固定的光罩來製成，而將可變化的電路設計部份，以某一些另外的光罩來構成。如此一來，則僅需在 ASIC 第一次製作時，需要每一層的光罩費用，而在往後的變更設計時，則僅需更改所需

更動的光罩即可。

4. Full Customize：

Full Customize 的 ASIC 製作方式，是完全以客戶所委託的電路設計為考量，將電路作成最佳的整合，以得到一個單價最便宜、性能最優越的解決方法。但其相對付出的是更高的 NRE 費用與更長的開發製作週期。

ASIC	Gate Array	CBA	Standard Cell	Full Customuze
單價成本	高	適中	適中	低
速度性能	低	適中	適中	高
NRE 費用	極低	低	適中	高
開發時程	極短	短	中	長
閘數	中~低	高~適中	高~適中	適中~低

表 2-1 各種 ASIC 製作方式之比較

第三章 使用工具介紹

3.1 VHDL 電路描述語言

3.1.1 VHDL 簡介

VHDL 是 Very High Speed Integrated Circuit Hardware Description 的英文縮寫。VHDL 電路設計語言規範的目的，即是要提供一個高階且快速的設計工具，它含蓋了電路的描述，電路之合成與電路之模擬等三大電路設計工作。

在 1970 年到 1980 年間，美國國防部為了電子電路的設計意涵，以文件方式將其儲存起來，以便其他人能輕易了解電路的設計意義，此為 VHDL 電路設計孕育的由來。

由於 VHDL 電路設計語言所能函蓋的範圍相當廣，它能適應於各種不同階層的設計工程師的需求。從 ASIC 的設計到 PCB 系統的設計，VHDL 電路描述語言，均能派上用場。在今天，VHDL 電路描述語言的市場，以每年約 30%的比例在成長，於 1993 年時，VHDL 電路描述語言首度超過了 Verilog 的市場。而 VHDL 電路描述語言之所以能快速成長，除了靠半導體製造技術快速進步之外，VHDL 電路描述語言所能提供的高階電路描述語言的方式，使得複雜的電路可以透過 VHDL 編譯器的電路合

成方式輕易且快速的達成設計的規格。所以 VHDL 電路設計語言實為硬體設計工程師之必備工具。

3.1.2 VHDL 的優點

VHDL 電路設計語言當作電路設計的工具，則可以由下面的幾點來分別予以說明：

【一】功能強大：

它包含了電路描述,電路合成與電路模擬等功能。它自 ASIC 設計到電路板設計，甚或大型系統之設計 VHDL 電路設計語言均能派上用場。

【二】設計靈活：

它將電子電路的設計方式，改便程以行為化描述的方式來設計。因此具有設計快速、更改容易、犯錯機率低和除錯容易等優點。

【三】各種不同的描述風格：

VHDL 提供許多不同的描述風格，來適應大小，複雜性不同的電路設計。

以一個 2 bit 的比較器為例：

・順序性敘述（sequential statements）:

```
IF a = b THEN
  eq<=’1’;
ELSE
  eq<=’0’;
END IF;
```

・共時行敘述（concurrent statements）：

```
eq<=’1’WHEN a=b ELSE ‘0’;
```

・布林方程式（booleen equaton）：

```
eq <= (a(0) XOR b(0) NOR (a(1) XOR b(1));
```

・連線關係描述：(netlist)：

```
u1:XOR2 PORT MAP(a(0),b(0),x(0))
u2:XOR2 PORT MAP(a(1),b(1),x(1)));
u:NOR2 PORT MAP(x(0),X(1),eq);
```

【四】可流通性或可攜性：

因為 VHDL 是一種工業界標準的電路設計語言，它所提供的

可攜性能力，可以使你的設計可以利用不同編譯軟體來編譯，可以利用不同模擬器來模擬以及適用於任何種類的邏輯元件與半導體製程技術。

3.1.3 VHDL中的重要詞彙與解釋

- ***單體(ENTITY)***

單體是一個設計中最基本的一部份，也是不可缺少的一部份，單體通常描述電路中的輸出入宣告。

- ***架構(ARCHITECTURE)***

架構是描述電路行為特性的主要部份。相當於電路圖設計中的schematics，用來描述各個元件間的互相連結情形。

- ***套件(PACKAGE)***

我們將一些副程式、程序及一些常用的資料型態，放在一個套件中，而形成一個工具箱，如果在設計中有應用到套件中的程式時，只要將此套件名稱寫在程式的最開頭部份，則在程式中任何地方，均可以使用子套件裡面包含的東西，套件程式部份可以單獨形成一個檔案分別單獨來編譯，以利整合設計的進行。

・*屬性(ATTRIBUTE)*

用來描述個種資料物件(data object)的特性，如陣列的位元數。

・*傳遞(GENERIC)*

VHDL 用來傳遞一些特性參數給單體電路使用。

・*資料型態(DATA TYPES)*

VHDL 語言中所使用的任何的資料物件，如常數、變數..等，均要定義其資料型別之後才能使用。

・*副程式(SUB-PROGRAMS)*

在 VHDL 語言中，有兩種副程式來使主程式的結構更加的簡潔清楚，他們是函式(FUNCTIONS)與程序(PROCEDURES)，函式使用在其只有一個輸出變數時，而程序則無此限制。

・*行程(PROCESS)*

用來描述行為化電路描述的最基本單元，而在 process 內所寫的程式，是有先後次序關係的，我們稱這些為 sequential statements。

・*構形(CONFIGURATIONS)*

可以容許一個單體電路內能有一個以上的構成情形。而程式設計者可以使用 CONFIGURATION 指令來說明該單體電路的構造情形。

3.1.4 VHDL的程式架構與語法

・VHDL的程式架構

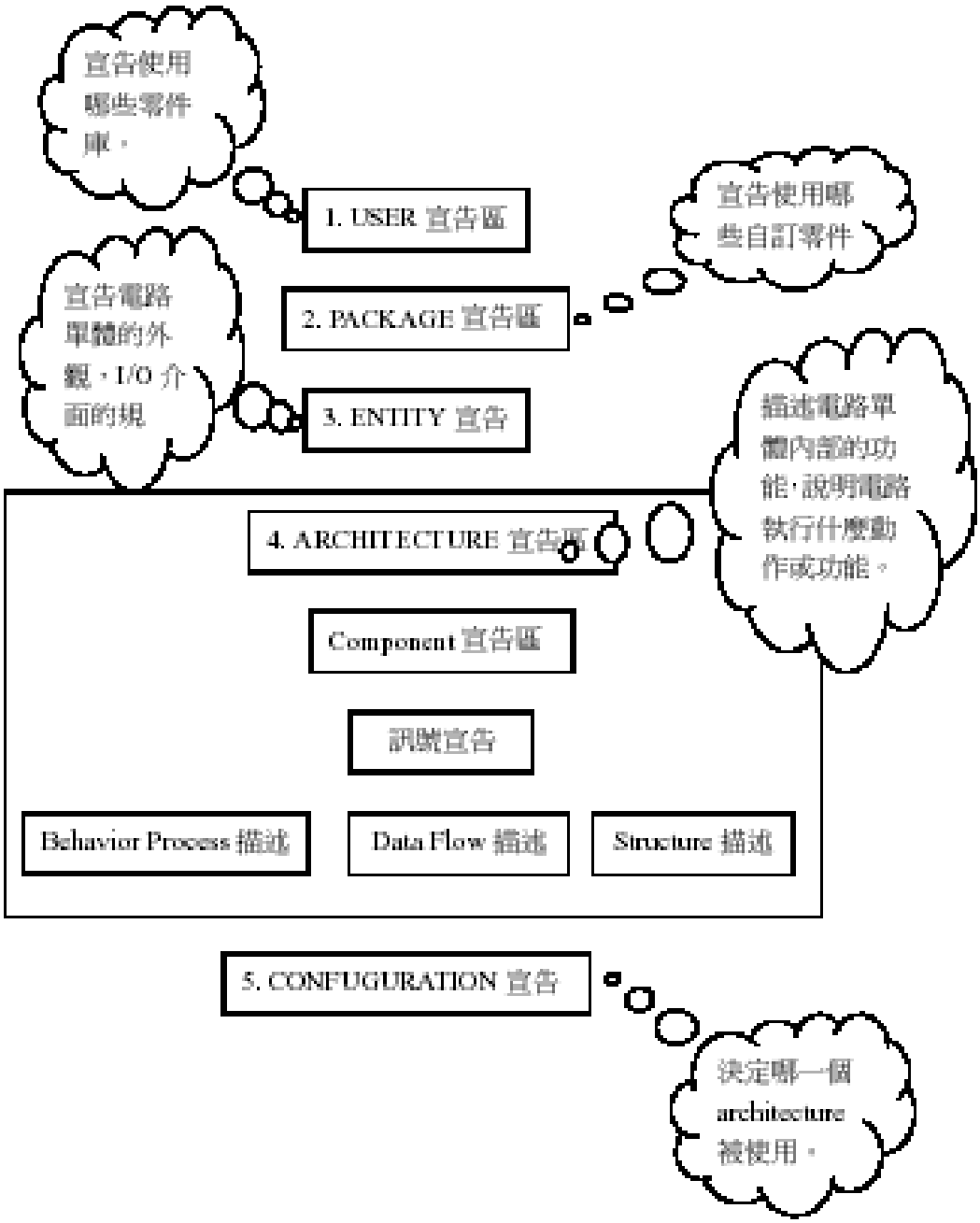

圖 3-1 VHDL的程式架構

• VHDL 的語法

• *Entity*：

Entity 在VHDL 語言中只是用來描述階層式方塊的界面，無需描述其他的行為。其語法如下：

```
ENTITY NAME IS
PORT(
A ：IN STD_LOGIC；
B ：OUT STD_LOGIC)；
END NAME;
```

在PORT 裡一共定義四種輸出入模式，IN、OUT、BUFFER、INOUT。

a. IN ：代表輸入模式，即此訊號需由電路單體外的電路來驅動之。

b. OUT：代表輸出模式，即僅能輸出電路單體之外去驅動外圍電路，不可回授回單體之內驅動內部電路。

c. BUFFER：代表緩衝模式，是一種可以回授驅動單體內電路的輸出模式。

d. INOUT：代表雙向模式PORT 訊號，可以將訊號回授至單體內驅動電路，也可以同時代表IN、OUT 和BUFFER 等模式訊號。

- ***架構描述（Architecture Description）***

架構描述主要描述電路單體的內部結構。架構描述有三個主要描述方式：架構（structure）、資料流（dataflow）、行為描述。

架構描述的語法：

```
ARCHITECTURE 架構名稱OF 電路單體名稱IS
--區域訊號宣告
BEGIN
--架構描述
END 架構名稱；
```

- ***處理程序（Processes）***

處理程序是一段VHDL 程式描述，其內的敘述依法執行之。處理程序只能在架構描述主體之內。同一個架構描述主體可以有數個互相溝通的處理程序，這些處理程序在同時工作。

```
格式一
標記名稱：PROCESS（感測訊號清單）
BEGIN
--PROCESS 敘述主體
END PROCESS 標記名稱；

格式二
PROCESS（感測訊號清單）
BEGIN
--PROCESS 敘述主體
```

```
END PROCESS ;
```

- ***零件盒（packages）***

零件盒實際上就是許多設計都可以參考使用的定義集合。將設計間常常用到的定義放在一起，將有助於團隊工作的一致性。在VHDL 語言將零件盒分成兩個部分：零件盒宣告（package declaration）和零件盒主體（package body）。零件盒內的宣告部分用來宣告的項目，包含常數（constants）、資料型態（types）、元件（components）、訊號（signals）、共用變數宣告（shared variables）、函數（functions）與程序（procedure）等項目。在宣告部分所宣告的函數與程序，必須在零件盒主體中將他們的實際內容定義清楚。

- ***編譯與程式庫（Compilations and Libraries）***

VHDL 程式的最小編譯項目就是單一設計單元。一旦VHDL 原始碼編譯之後，所產生的二進位檔會存入指定磁碟目錄裡。通常給目錄一個邏輯名稱“程式庫”（library），當使用的零件名稱不在目前工作程式庫內，我們就需以LIBRARY 字句來指明欲使用的程式庫。當VHDL 程式欲使用某一零件盒所定義的資料型態、元件、函數或程序時，僅需用USE 字句將零件包括在VHDL

程式中就可以了。

LIBRARY 程式庫名稱
例如：LIBRARY IEEE；

USE 程式庫名稱. . 零件盒名稱 . 指定項目；
例如：USE IEEE.STD_LOGIC_1164.ALL；
USE WORK.DANIEL.REG12；

在1990 年早期，VHDL 主要使用在複雜的ASIC 設計上，使用電路合成器自動地產生及最佳化ASIC 設計。一直到1990 年後半期，VHDL 與電路合成才逐漸在PLD（Programmable Logic Design 可規劃邏輯設計）的領域上。以VHDL 作規格塑造（modeling specifications）也有逐漸增加的趨勢。

目前VHDL 在類比領域上的運用有相當大的限制，不過在VHDL 的類比標準已有相當工作正在努力中，也有業者投入研究，來開發VHDL 在類比方面的應用技術，相信在不久的將來，必能得到豐碩的成果，屆時，VHDL電路描述語言將可以應用在類比/數位電路的開發與設計上。

3.2 Altera MaxPlus II

MAX+PLUS II 是由Programmable Logic Device 的提供商ALTERA Inc 所提供的設計、驗證與燒錄的軟體。在安裝好MAX+PLUS II 軟體後，我們就能在WINDOWS 的程式集中看到。

以下我將簡要的說明 MAX+PLUS II 軟體所支援的功能選項：

Hierarchy Display ：

用來顯示一個大型電路設計的階層化結構(Hierarchical Structure)。

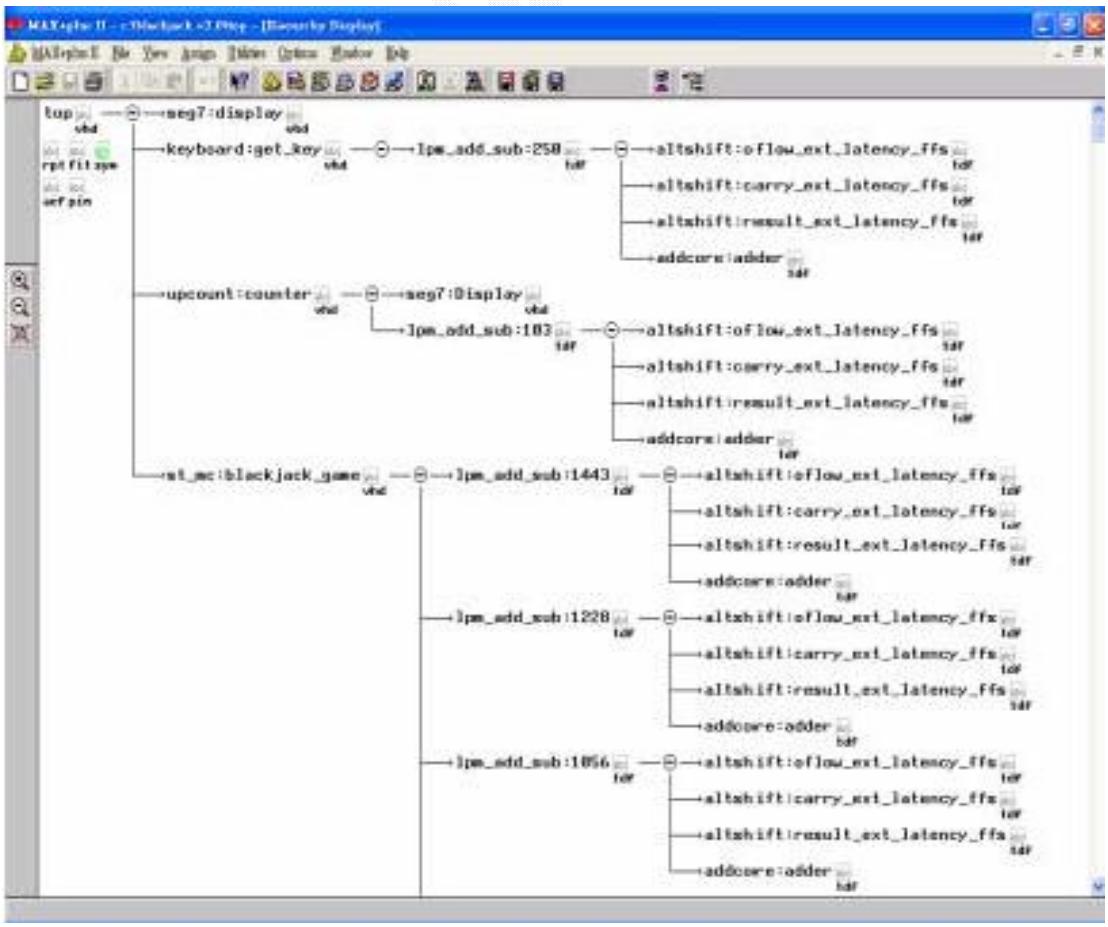

圖 3-2 Hierarchy Display 操作環境

Graphic Editor ：

這是用來提供電路圖輸入的介面。使用者可利用此圖形編輯器，利用繪製電路圖(schematics)的方法，來作為 電路功能的設計輸入(design entry)。

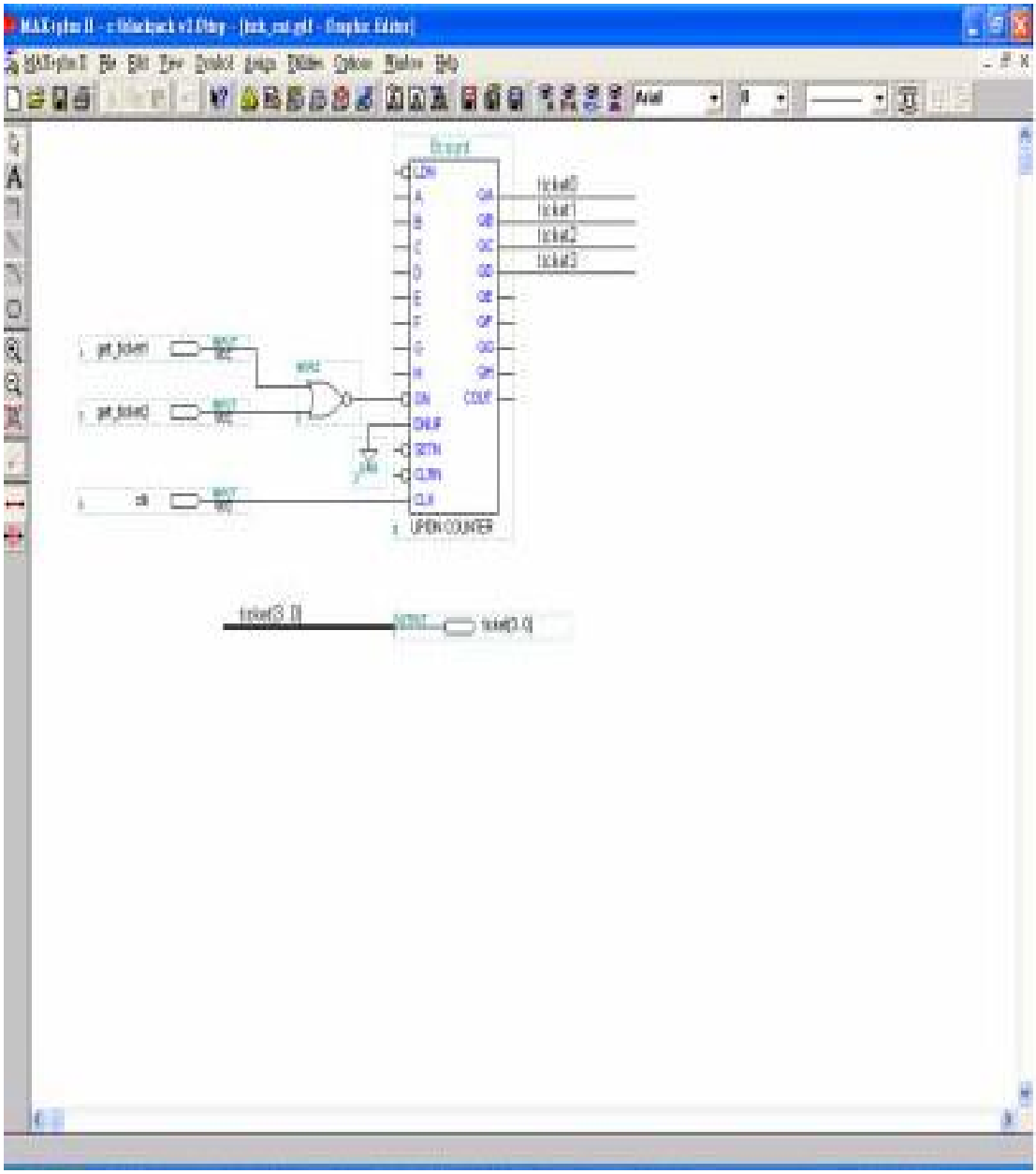

圖 3-3 Graphic Editor 操作環境

Symbol Editor ：

在利用電路圖的方法來當作電路的設計輸入時，要用一些浮號(symbols)來代表一些合成一組的電路功能。Symboleditor 即是用來編輯電路圖上的元件符號。

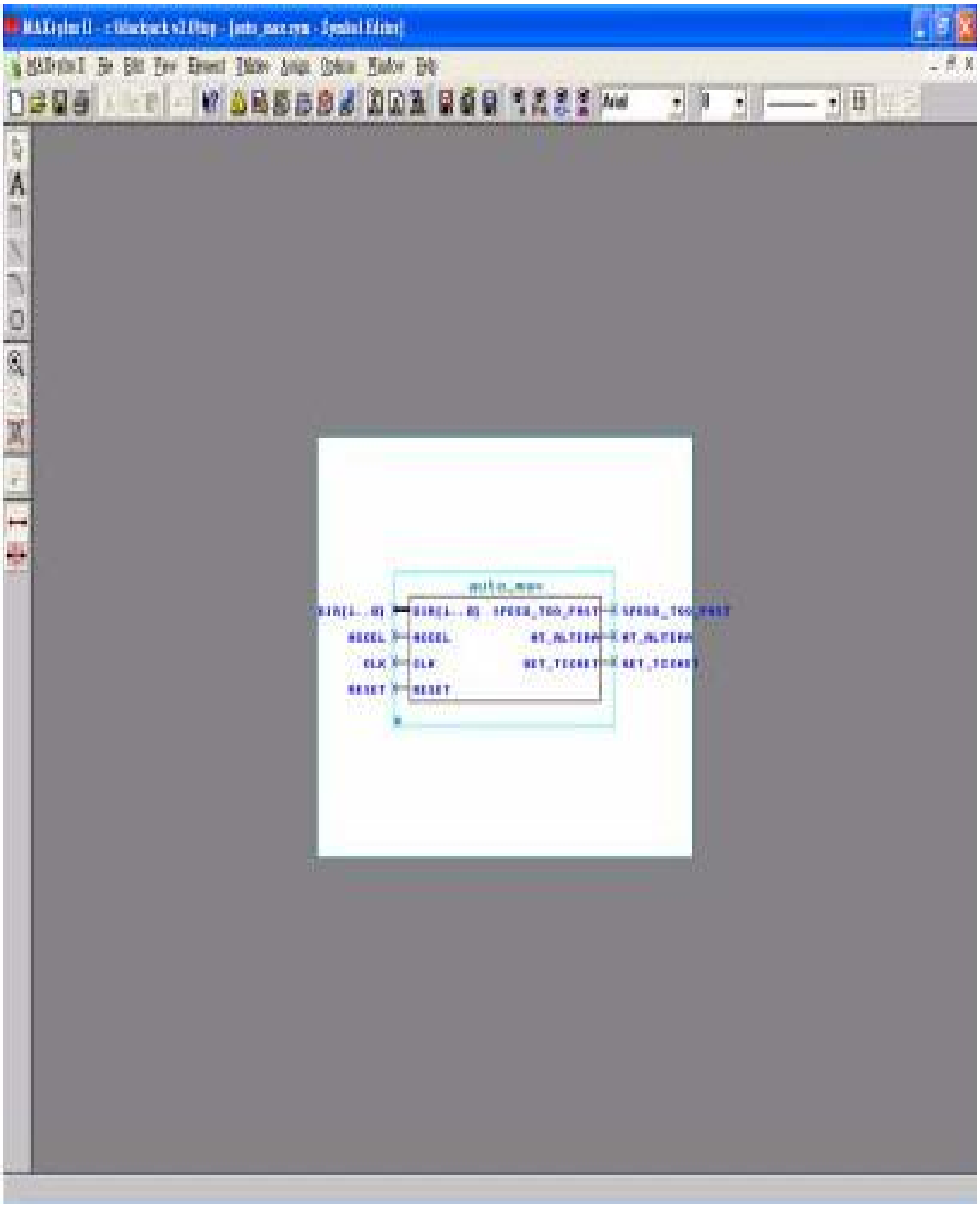

圖 3-4 Symbol Editor 操作環境

Text Editor ：

為 maxplus 所內建的文字編輯器，其可用來當作 vhdl 語言的輸入。

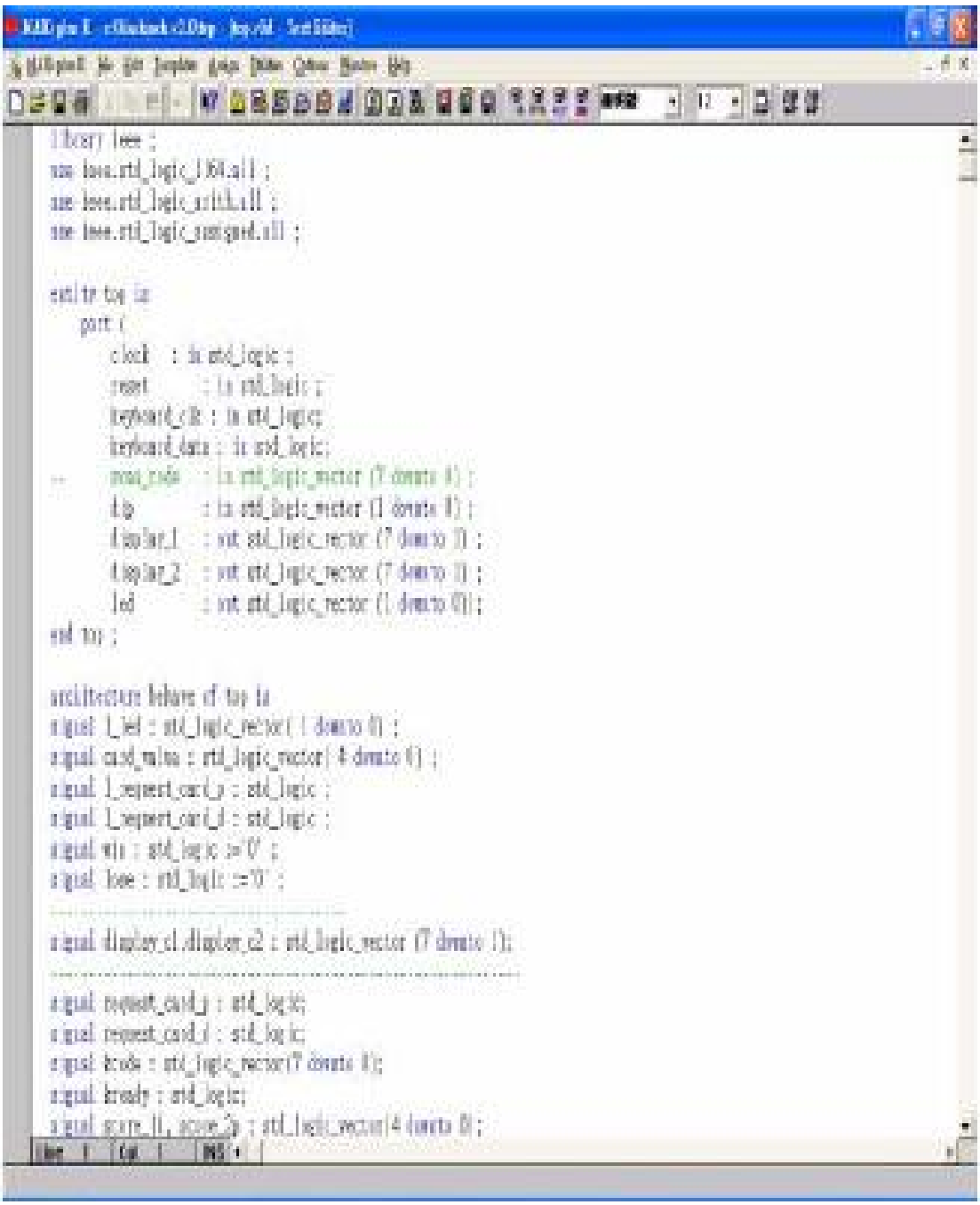

圖 3-5 Text Editor 操作環境

Waveform Editor ：

用來編輯訊號的波形圖可分成兩種。一種是用來模擬電路時序及驗證其正確性的波形模擬，另一種是為用輸入訊號波形的相對關係當作電路功能的設計輸入。

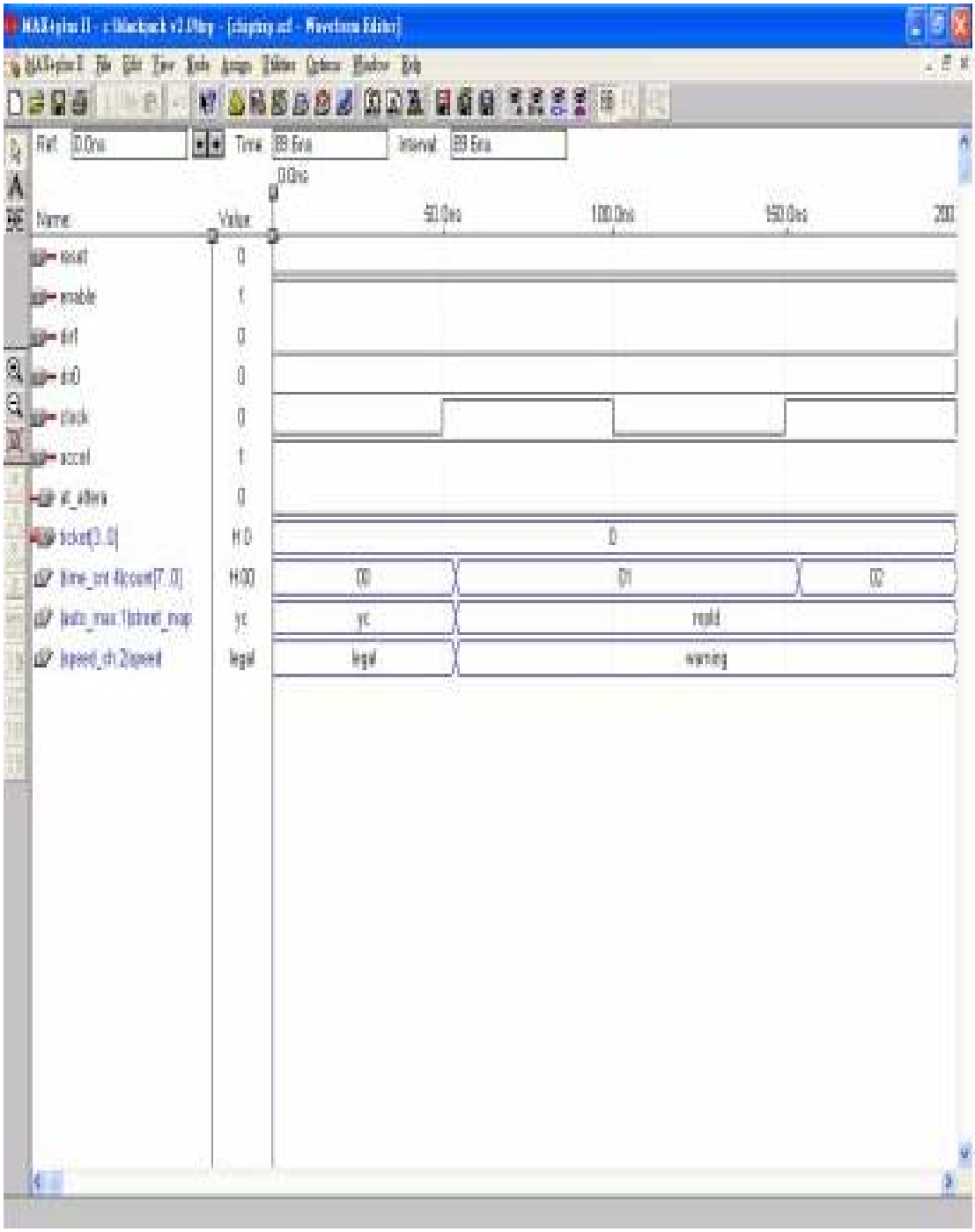

圖 3-6 Waveform Editor 操作環境

Floorplan Editor ：

用來直接在 FPGA 或 PLD 的佈局圖上編輯 cell 的功能與連線關係。

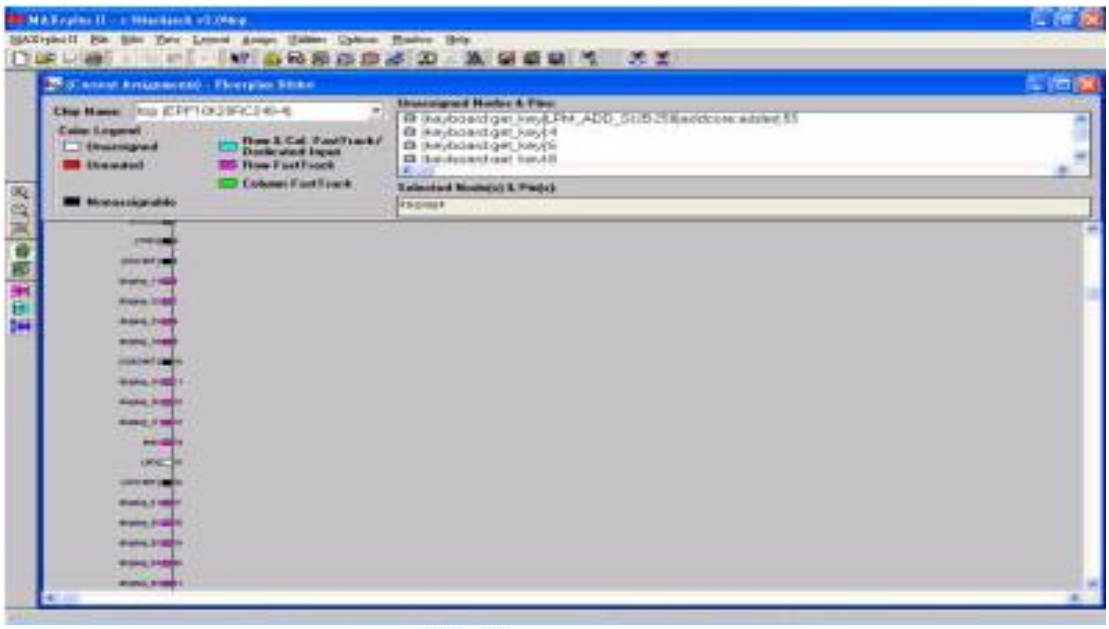

圖 3-7 Floorplan Editor 操作環境（一）

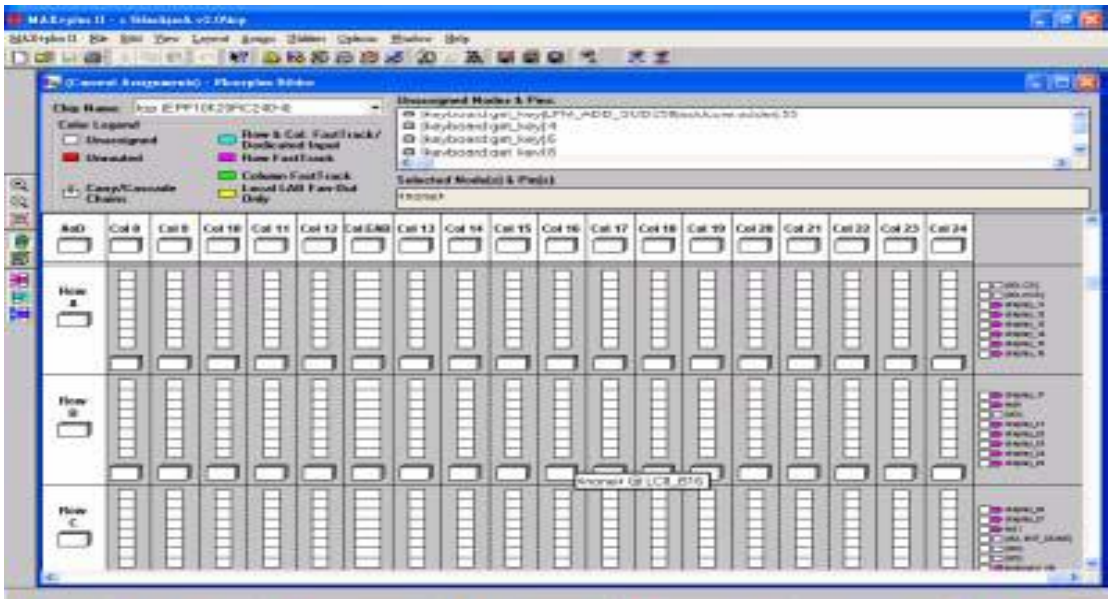

圖 3-8 Floorplan Editor 操作環境（二）

Compiler ：

此為電路功能的編輯器，用來將設計輸入，經過編譯後，產生實際電路功能的可燒錄程式檔。

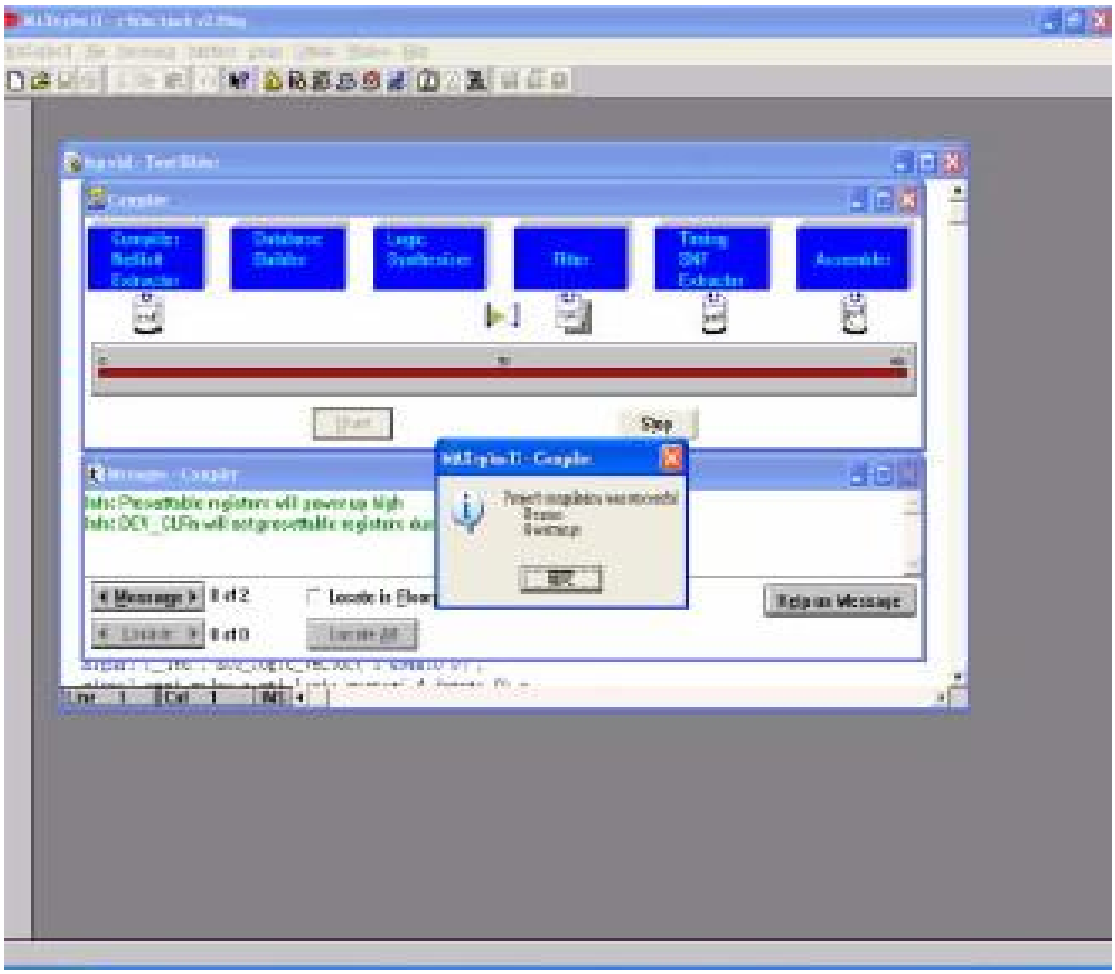

圖 3-9 Compiler 操作環境

Simulator ：

此為訊號波形的模擬器，用來分析訊號時序邏輯的正確性。

Timing Analyzer ：

用來分析電路元件的輸出入接腳間的時序延遲的狀況。

Programmer ：

此為 maxplus 所提供的邏輯元件燒錄器，可支援 Altra 提供的 Byte Blaster 和 Bit Blaster 等的邏輯編輯器。

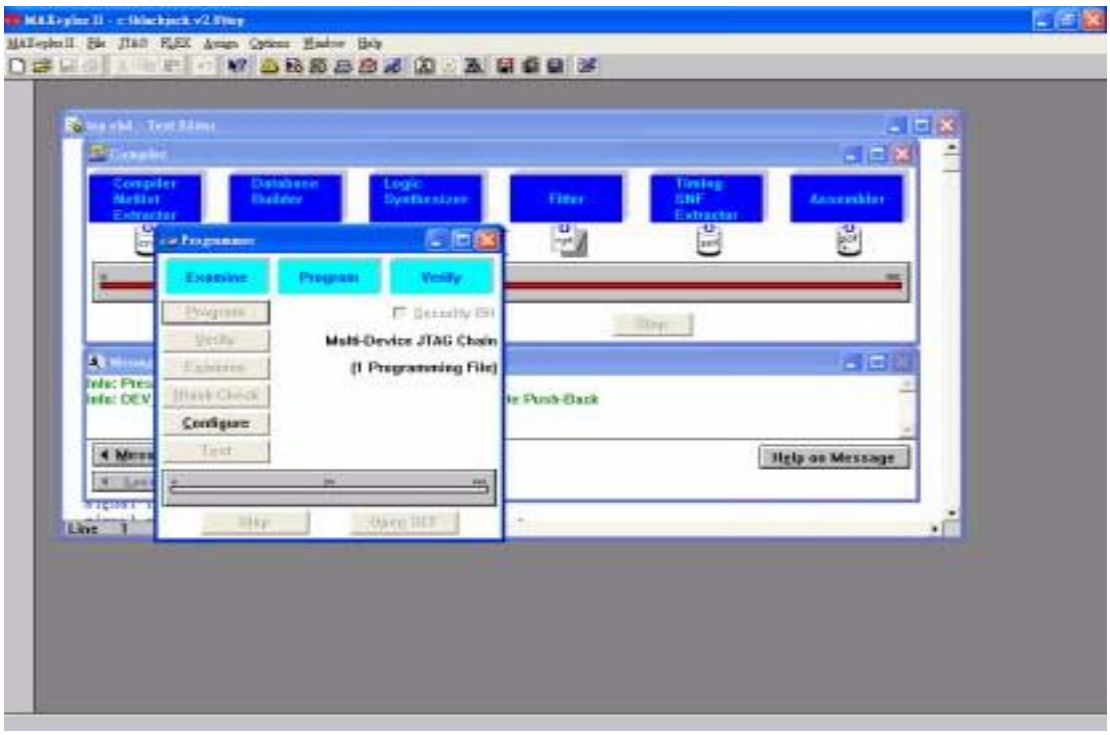

圖 3-10 Programmer 操作環境

Message Processor ：

此為程式的編譯過程或訊息號波形的模擬過程中所產生的一些訊息，如 errors、warnings...等，的訊息處理器，其可以將這些訊息所設計的電路或訊號作一連結。

第四章 DMA 簡介

4.0 三種 I/O 方式說明

・Program controlled I/O 或稱為polling I/O:

此種方式為 CPU 反覆地輪流檢查各I/O設備的 data ready 信號，但浪費過多的CPU 時間在 I/O 查詢上，效率不高。一般應用於單一使用者的環境，或無需握手式(Hand-Shaking)信號溝通的 I/O 設備，如LED，七段數字顯示器、繼電器、指撥開關等，反應慢，資料搬動也慢。

・ Interrupt I/O:

在這個模式下，I/O 設備的 data ready 信號即中斷要求信號。於中斷服務程式中，進行資料之搬動。CPU 可空出較多時間來處理主要事務，效率較高。主要應用於即時(real-time)系統，如自動控制系統、飛彈導航系統、反應快，但資料搬動如 Polling I/O 一般，由 CPU 執行程式進行，是影響效率主因。

・DMA I/O:

於此模式下時，I/O 裝置需要服務時，便發出 DMA 請求信號，系統

認可後，便由 DMA 控制器負責記憶體對 I/O 裝置直接交換資料，因不需 CPU 的讀寫過程，data ready， data transfer 均由硬體處理，高速傳輸資料，效率最高。適用於資料量大，短時間產生速度快的 I/O 設備上，如CRT 畫面之更新，Disk-I/O 等。另外，鍵盤介面(資料產生速度慢)、無須使用DMA；繼電器，七段顯示器，均無 data ready 信號，只能使用program controlled。

- **三種 I/O 方式比較表**

I/O 名稱	Data ready 的偵測	Data transfer 的控制
Program controlled I/O	由軟體	由軟體
中斷 I/O	由硬體	由軟體(ISR,中斷服務程式)
DMA I/O	由硬體	由硬體

表 4-1 三種 I/O 方式比較表

4.1 DMA Controller 概述

所謂 DMA 就是 Direct Memory Access 直接記憶體存取的縮寫，為微電腦/個人電腦/迷你電腦/大型電腦系統與週邊設備之間傳輸大量資料的一種方法。它可以允許某些週邊裝置，直接地去 RAM 中存取資料，而不用經過 CPU，是一種權宜的措施，不但可節省 CPU 的工作時間，又可提高工作的效率。這和一般傳統的 I/O 設備對記憶體之間的資料傳輸方法不同，前者的方法是以硬體的方式來完成二者之間資料的傳輸，至於後者則是以軟體的方法(查詢或中斷法)來傳輸資料，即靠 CPU 執行主程式或中斷副程式以達成 I/O 設備與記憶體之間資料的傳輸。由於 DMA 完全是以硬體的方法來達成 I/O 與記憶體間資料的傳輸，在理論上是比 CPU 利用程式控制來得快(現在 CPU 的性能卓越，在 memory-memory 傳輸時，效能比 DMA 來得好)。

我們簡單說明利用 CPU 執行程式與 DMA 的方法控制 I/O 與記憶體間資料傳輸的差異。

CPU 欲將資料從記憶體送到 I/O 或是由 I/O 送到記憶體，都必須先經由 CPU 處理然後才會傳至目的地。當彼此間傳送的資料量少時還算

可行，但如果資料量過於龐大時，則由於 CPU 必須連續的讀取指令並執行之，以完成 I/O 與記憶體間的傳輸，浪費過多的時間在執行與等待資料的傳輸上，降低了工作效率。

而利用 DMA。如一 I/O 設備欲以 DMA 的方法傳輸資料，它先產生一 DMA 請求信號(DRQ)給 DMA 控制器。當 DMA 控制器收到此一信號時，如果它沒有被 mask 的話，DMA 隨即產生一持有請求信號(HRQ)給 CPU 的 HOLD 輸入接腳，請 CPU 將系統的控制權讓出。此時 CPU 會將控制信號線，位址線，以及資料線變成高阻抗浮接(floating)狀態，然後送出一個讓出所有匯流排控制權的信號(HLDA)給 DMA 控制器。

當 DMA 控制器收到來自 CPU 的 HLDA 信號後，其隨即接管匯流排的控制權，然後送出一個 DMA 認可信號(DACK)給 I/O 設備。緊接著 DMA 控制器會同時令控制信號線(MEMR/MEMW/IOR/IOW)動作，以完成資料的傳輸。若資料是從 I/O 設備寫入記憶體，則控制信號線上的 MEMW 與 IOR 線動作。反之，若資料是從記憶體輸出至 I/O 設備，則 MEMR 與 IOW 線動作。

當資料傳送結束後，DMA 控制器隨即將讓出匯流排控制權的信號傳給 CPU 的 HOLD 接腳，請 CPU 重新接管匯流排控制權。當 CPU 拿回控制權之後，其隨即繼續執行原有的程式，一直到 DMA 控制器再送出持有請求的信號為止。

4.2 DMA Controller 與 CPU 間匯流排控制權轉換方式

4.2.1. 週期竊取法(Cycle Stealing Mode) — 像用偷的一樣

此模式利用 CPU 做內部操作或指令解碼等未使用匯流排的瞬間，迅速的偷取匯流排使用權並做一次資料傳輸，然後立即將匯流排使用權交還 CPU。這種方法不會延誤 CPU 時間，效率很高。

如 Motorola 6800 系列 8 位元 CPU 因只在匯流排週期(Bus Cycle)的後半週期使用匯流排(即 E 信號為高電位時)，故其前半週可拿來做 DMA 操作，甚至可以使用另一工作在前半週的 CPU 來共同運作，這種技術叫雙相運作模式(Bi-phase Opera-tion)。Intel 的 CPU 不適用於 Bi-phase 操作。

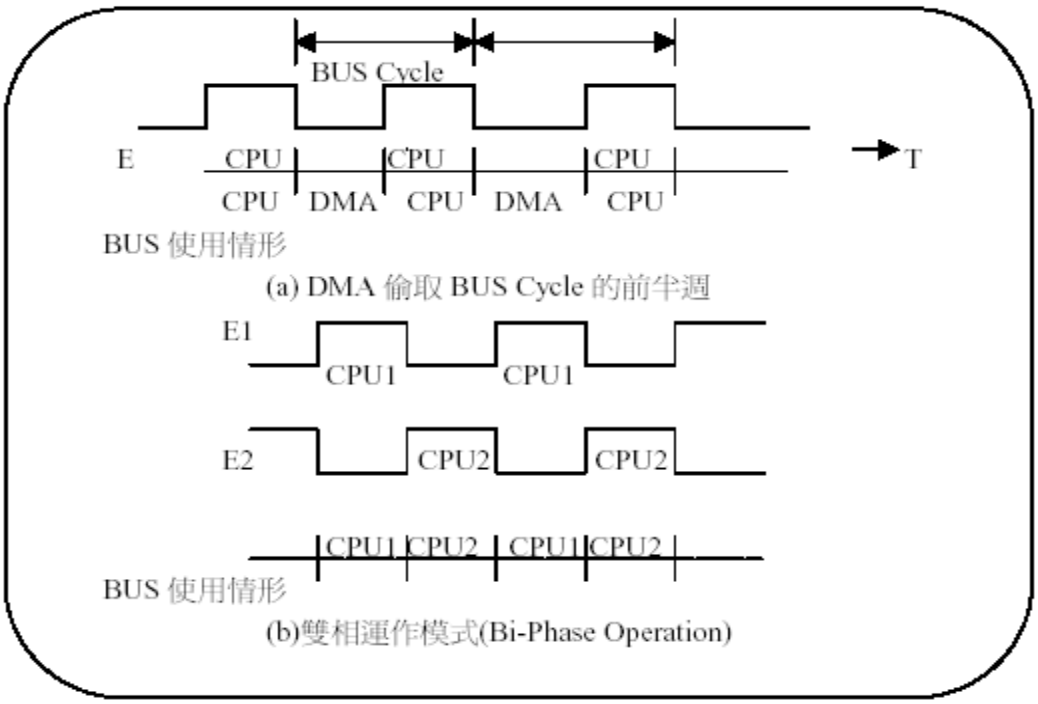

圖 4-1 週期竊取法示意圖 & 雙相運作模式

週期竊取模式常被用來做電腦顯示幕(如 CRT)的掃瞄,Apple 公司的 Apple-II 即採用這種 DMA 方式做螢幕掃瞄控制，這方式的缺點是螢幕掃瞄的位元組(Byte)速率必須與 CPU 時脈同步，若螢幕的解析度(Resolution)要提高時，掃瞄速度須加快，便須改變系統時脈的頻率，變成系統時脈與螢幕控制無法獨立，這並不好，但它的優點是 CPU 存取顯示記憶區時為完全為”零等待”(Zero Wait)狀態，因為 DMA 是用偷取時間的方式，不會影響 CPU 工作。

4.2.2.暫停 CPU 的方式(Suspend CPU Method) — 像用搶的一樣

這種方式也是一次 DMA 週期只傳輸一筆資料，只差它是屬於非同步式的。當 DMA 動作時若恰好遇到 CPU 也正要存取該部分記憶體，則強迫 CPU 暫停，等 DMA 控制器存取完一筆資料後再令 CPU 繼續動作，這種 DMA 方式可說是用搶匯流排的方式。

這種操作方式會讓 CPU 的速度稍為變慢,但 DMA 存取速率卻可以不與 CPU 的時脈同步。

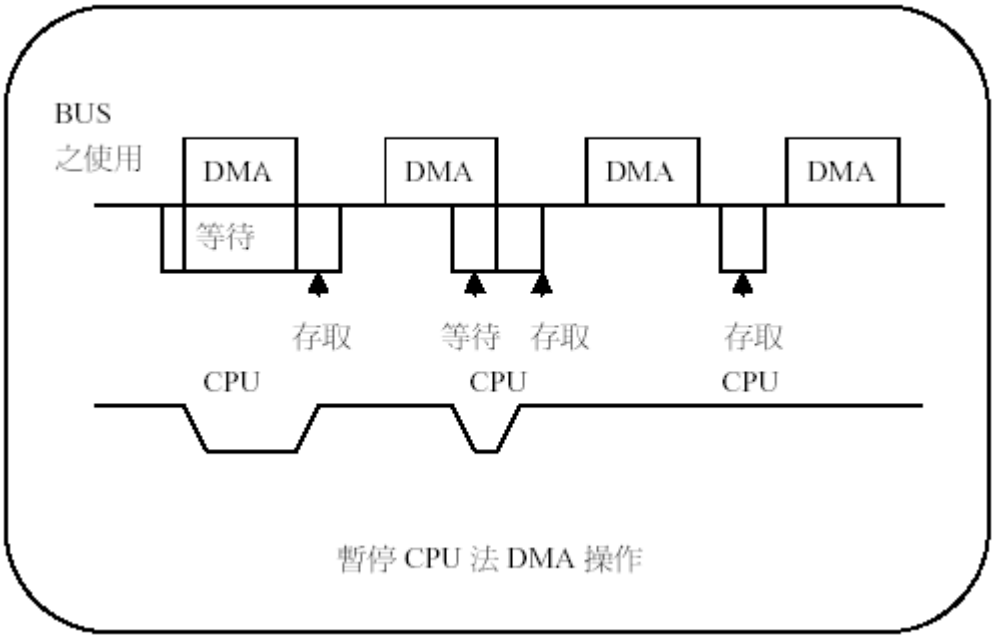

圖 4-2 暫停 cpu 模式示意圖

PC/XT/AT 系統的螢幕掃瞄控制即使用暫停 CPU 方式，它以 MC6845(CRTC)做螢幕顯示控制器.PC 的時脈與螢幕掃瞄頻率是互不相干的，因此螢幕控制卡可以插在各種不同頻率的 PC 系統仍能正常操作，同樣的 PC 系統可以使用單色，CGA、EGA、VGA 等不同頻率的顯示卡，主要是因採用非同步的 DMA 掃瞄控制的緣故。

4.2.3.停止 CPU 方式(Stop CPU Method) — 像用協調的一樣

此種方法可在一次 DMA 週期傳輸一筆資料或一區塊資料(a block of data)，區塊資料傳輸方式又叫突發模式(Burst Mode)，當 I/O 裝置需要 DMA 服務時，透過 DMA 控制器與 CPU 協調,請求做 DMA 操作，在

CPU 允許後，即釋放匯流排給 DMA 控制器使用，直到資料傳輸完畢後再交回 CPU，這段期間 CPU 完全處於停止狀態。

DMA 控制器(DMAC)亦可以是另一個 CPU，而兩個 CPU 以匯流排仲裁(Bus Arbitration)信號來互相協調使用匯流排時機。

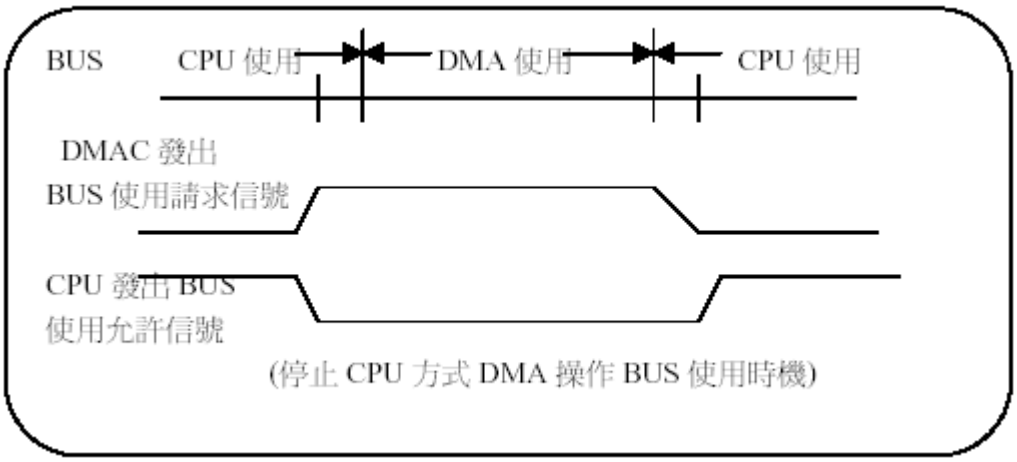

圖 4-3 停止 cpu 模式示意圖

停止 CPU 方式的 DMA 操作，常使用在諸如磁碟機(Disk Drive)的存取控制方面，例如 PC 的磁碟控制卡常有用一單晶片的微控制器來完成這種操作。

4.2.4.拉長時脈週期方式(Stretch Period Method) — 像用騙的一樣

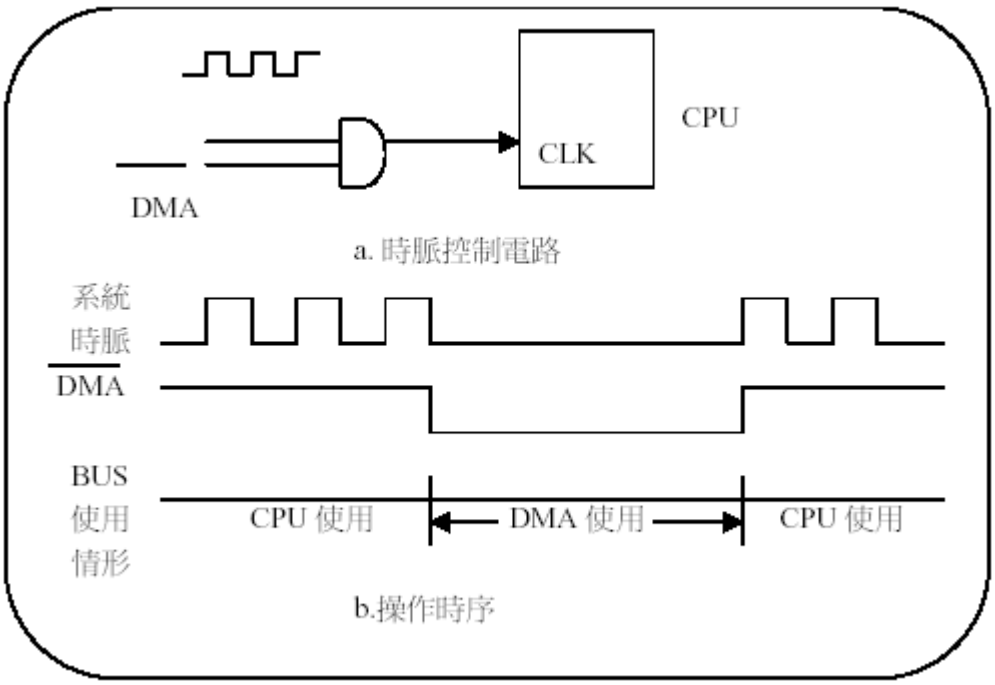

圖 4-4 拉長時脈週期模式示意圖

以欺騙 CPU 的方式在這段延長時間裡取用匯流排做 DMA 操作。其操作時序(如上圖 b)所示。

以上 DMA 控制器與 CPU 分享匯流排使用的方式,都是為了取得有效率的資料傳輸的種種方法，在應用上，要取捨何者來做設計，那就得依用途而仔細評估了。

第五章 DMA 架構、規格及功能定義

5.1 DMA 架構說明

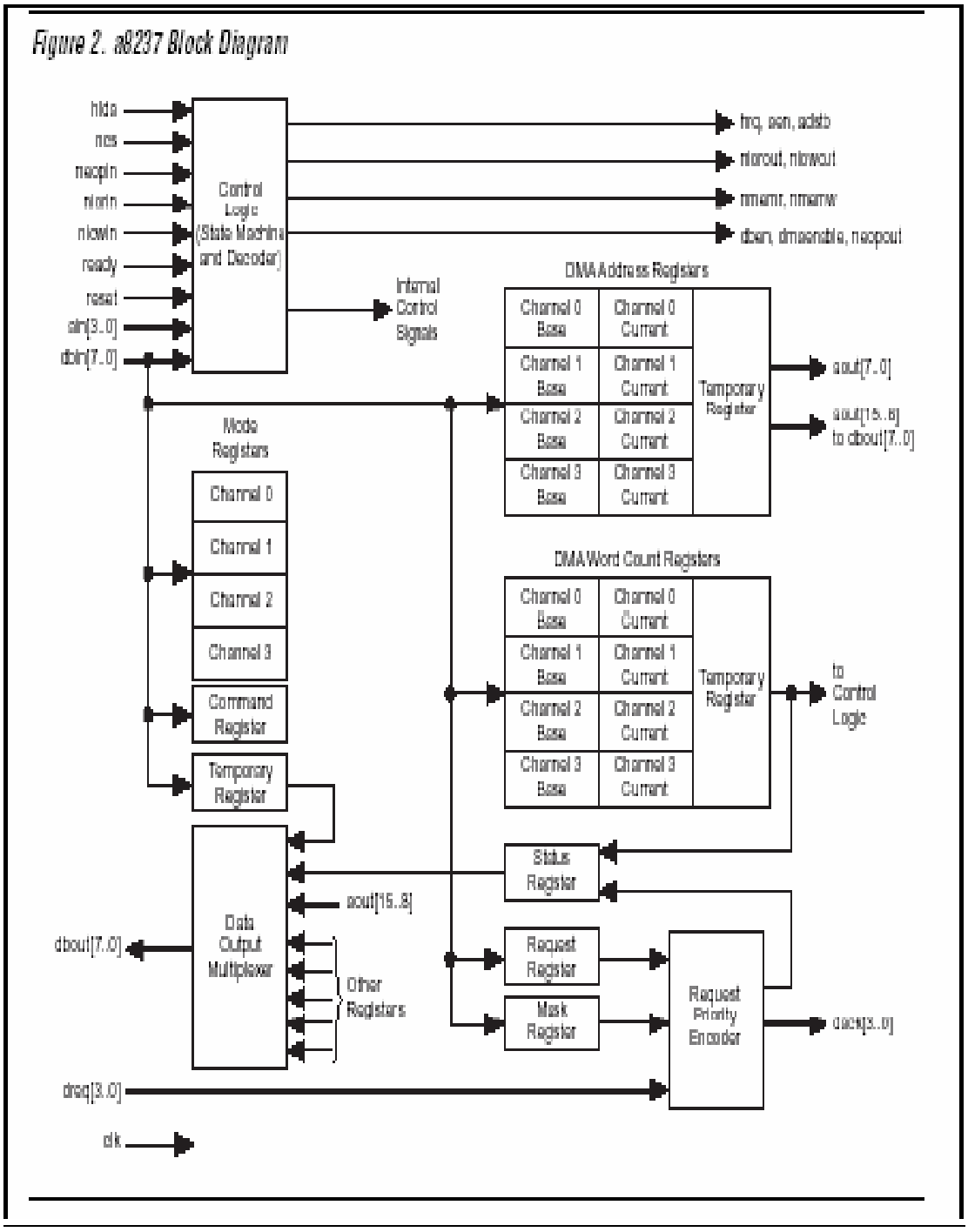

圖 5-1 DMA Controller 架構方塊圖

5.2 DMA 控制器接腳說明

CLK(時序信號線)　(TYPE : INOUT)

此一信號用以控制 DMA 的內部作業以及資料傳輸的速率。在我們所設計的控制器中，FINITE STATE MACHINE 的狀態變化就是利用此一信號的下降緣觸發。

NCS(晶片選取信號)　(TYPE : IN)

此一信號是用來致能或禁能 DMA 之用。當 NCS 為低電位時，DMA 被致能;反之，當 NCS 為高電位時，DMA 被禁能。當 DMA 處於禁能狀態時，CPU 無法將資料寫入或讀自 DMA。反之，則可接受來自 CPU 的資料或將資料傳給微處理器。通常 NCS 都是接至用以選取此一元件的 I/O 解碼電路之輸出端上。

RESET(重置信號線)　(TYPE : IN)

此一信號線在高電位時動作，除了可清除命令(COMMAND)、狀態(STATUS)、請求暫存器(REQUEST)。還可清除前後正反器(FIRST/LAST FLIP-FLOP)，以及設定罩遮(MASK)暫存器。當 DMA 被重置後，其隨即處於閒置週期。

註:因為 DATABUS 為八位元，而位址、計數暫存器則是十六位元。因此電路中有一個 FLIP-FLOP 用於決定寫入暫存器的 most-significant(當值為 1 時)或 least-significant(當值為 0 時)的八個 bits。

當有寫入的動作時，其值隨即變化一次。

READY(備妥信號線)　(TYPE : IN)

為一輸入信號線，低電位時動作，用來令 DMA 插入等候週期，以便讓存取速度較慢的記憶體或 I/O 裝置能來得及完成寫入或是讀出的動作。也就是用來延長 READ 與 WRITE 信號的波寬之用。當 READY 為低電位時，DMA 即進入等候週期直到變為高電位為止。

HRQ(DMA 持有請求信號線)　(TYPE : OUT)

HRQ 為 DMA 請 CPU 讓出系統控制權的信號線。以便讓控制器掌握匯流排及所需信號線(如:memory write…等)的使用。

如果 I/O 有發出 DREQ 或是請求暫存器有 dma request(即有 valid dma request)，則 HRQ 將會產生一高電位的脈衝信號給 CPU，請其讓出系統控制權。

HLDA(持有認知信號線)　(TYPE : IN)

此一信號與 HRQ 是一組的,於高電位時動作。當 HLDA 為高電位時,表示 CPU 已同意將系統控制權讓給 DMA。DMA 則可以開始運作。

DREQ(DMA 請求信號線)　(TYPE : IN)

為週邊設備向控制器要求執行 direct memory access 的請求信號線。它的動作準位可以程式設定為高或是低電位動作，但當 RESET 之後則為高電位動作。

它必須保持到 DACK 信號送出給週邊設備為止。

DACK(DMA 認可信號線)　(TYPE : OUT)

與 DREQ 是相對應的,用以告知週邊設備其所要求的 DMA 請求已被允許。它一樣可以程式設定高或低電位動作，但 RESET 之後則必為低電位動作。

DATABUS[7..0](資料匯流排)　(TYPE : INOUT)

這是八條雙向，三態的資料線，用以連接系統的資料線。當 CPU 與 DMA 做一般的 I/O 讀取或寫入動作時(即 CPU 讀、寫，控制器裡頭的暫存器)，DB0-DB7 為二者之間的資料線。但是當 DMA 取得系統控制權

後，在執行 DMA 動作時，它在某一個 clock cycle 被用來送出 ADDRESS 的八個 most-significant bits(15..8)給外部的 address latch。

IOR(I/O 讀取信號線)　(TYPE : INOUT)

為一雙向三態低電位動作的信號線。當 DMA 處於閒置週期時，它為一輸入信號線，提供 CPU 讀取 DMA 內部暫存器之用。而在 DMA 動作週期時，它為一輸出信號線，供 DMA 發出讀取訊號給週邊設備，讓週邊設備將資料給到資料匯流排。

IOW(I/O 寫入信號線)　(TYPE : INOUT)

為一雙向三態低電位動作的信號線。當 DMA 處於閒置週期時，它為一輸入信號線，CPU 靠其將資料寫至 DMA 內部暫存器中。在 DMA 動作週期時 IOW 為一輸出信號線，供 DMA 將寫入訊號給到週邊設備，通知週邊設備由匯流排抓取資料。

MEMR(記憶體讀取信號線)　(TYPE : OUT)

為低電位動作的信號線。在 DMA 控制器執行 DMA 讀取動作時，它用來致能記憶體以便資料能從指定的記憶體中讀取出來，放到匯流排，準備讓週邊設備抓取。

MEMW(記憶體寫入信號線)　(TYPE : OUT)

為低電位動作的信號線。在 DMA 控制器執行 DMA 寫入動作時，它用來致能記憶體以便匯流排上的資料能寫入至所指定的記憶體中。

ADDBUS[3..0](位址線)　(TYPE : INOUT)

ADDBUS 的 bit3 至 bit0 為雙向三態信號線。在閒置週期時，為 I/O 位址輸入線，有十六個不同的 I/O 位址(4 個 bits 可決定出十六個位址)可供 CPU 讀取/寫入 DMA 內部不同的暫存器。在 DMA 動作週期時，bit3-bit0 為位址輸出線，是要 access memory 位址的部份 bits。

ADDBUS[7..4](位址線)　(TYPE : INOUT)

A7-A4 為三態位址雙向輸出信號線(因為只有在 DMA 動作週期時才有做用)，只有在 DMA 週期時才被用來 access memory(是位址線的 bit7 至 bit4。

NEOP(DMA 處理終了信號線)　(TYPE : IN)

為一低電位動作的信號線。外來的低準位信號將使得 DMA 結束正在進行的 DMA 動作。

註:dma 的結束有二種情況，一為當計數終了，即所要傳的 bytes 完全傳輸完畢。二為當 neop 的信號是為低準位時，提早結束 DMA 傳輸。

NEOPOUT　(TYPE : OUT)

為一低電位動作的信號線，當計數終了時，DMA 將經此接腳輸出一個低電位的脈衝信號，以示 DMA 傳輸完畢。

DBEN(資料致能信號線)　(TYPE : OUT)

DBEN 為一高電位動作的信號線。它主要是用在，當 CPU 讀取 DMA Controller 內部暫存器時，通知 CPU 抓取正確的資料之用。

AEN(位址致能信號線)　(TYPE : OUT)

AEN 為一高電位動作的信號線，它除了可用以致能位址信號線(ADDRESS[15..0])外(表示現在於位址匯流排上的資料是正確的)，亦可在 DMA 動作週期時令 CPU 所控制的系統匯流排失效。避免衝突。

ADSTB(位址觸發信號線)　(TYPE : OUT)

ADSTB 為一高電位動作的信號線，主要是用來將位址的八個 most-significant bits，ADDRESS[15..8]拴鎖於外部位址閂鎖器上。注意，位址 ADDRESS[15..8]乃是 DMA 在動作週期時，經由 DATABUS 給到外部位址拴鎖器的。

5.3 DMA 控制器裡的暫存器

TABLE 1 及 TABLE 2 分別是要讀寫 DMA Controller 內部暫存器所對應到的位址。(只用到 4 個 bits)

TABLE 1

REGISTER ADDRESS MAP					
ADDRESS[3..0]				CHANNEL	WRITE
A3	A2	A1	A0		
0	0	0	0	0	BASE AND CURRENT ADDRESS REGISTER
0	0	0	1		BASE AND CURRENT COUNT REGISTER
0	0	1	0	1	BASE AND CURRENT ADDRESS REGISTER
0	0	1	1		BASE AND CURRENT COUNT REGISTER
0	1	0	0	2	BASE AND CURRENT ADDRESS REGISTER
0	1	0	1		BASE AND CURRENT COUNT REGISTER
0	1	1	0	3	BASE AND CURRENT ADDRESS REGISTER
0	1	1	1		BASE AND CURRENT COUNT REGISTER
1	0	0	0	X	COMMAND REGISTER
1	0	0	1	X	
1	0	1	0	X	
1	0	1	1	X	MODE REGISTER
1	1	0	0	X	CLEAR BYTE POINTER COMMAND
1	1	0	1	X	
1	1	1	0	X	
1	1	1	1	X	MASK REGISTER

NOTE: If the byte pointer is set to 0,the byte pointer flag selects the least signific
If the byte pointer is set to 1,the byte pointer flag selects the most signifi

The byte pointer flag is a single-bit internal register that selects either the significant or most significant byte of the 16-bis register in the DMA Cor allowing the microprocessor to write and read via the 8-bit data bus.

The X indicates "don't care".

表 5-1 讀寫 DMA Controller 內部暫存器所對應到的位址(1)

TABLE 2

REGISTER ADDRESS MAP					
ADDRESS[3..0]				CHANNEL	READ
A3	A2	A1	A0		
0	0	0	0	0	BASE AND CURRENT ADDRESS REGISTER
0	0	0	1		BASE AND CURRENT COUNT REGISTER
0	0	1	0	1	BASE AND CURRENT ADDRESS REGISTER
0	0	1	1		BASE AND CURRENT COUNT REGISTER
0	1	0	0	2	BASE AND CURRENT ADDRESS REGISTER
0	1	0	1		BASE AND CURRENT COUNT REGISTER
0	1	1	0	3	BASE AND CURRENT ADDRESS REGISTER
0	1	1	1		BASE AND CURRENT COUNT REGISTER
1	0	0	0	X	STATUS REGISTER
1	0	0	1	X	REQUEST REGISTER
1	0	1	0	X	COMMAND REGISTER
1	0	1	1	X	MODE REGISTER
1	1	0	0	X	
1	1	0	1	X	
1	1	1	0	X	
1	1	1	1	X	MASK REGISTER

表 5-2 讀寫 DMA Controller 內部暫存器所對應到的位址(2)

COMMAND REGISTER(命令暫存器)

COMMAND REGISTER FORMAT	
BIT	DESCRIPTION
0	RESERVED
1	RESERVED
2	0=CONTROLLER ENABLE 1=CONTROLLER DISABLE
3	0=NORMAL TIMING 1=COMPRESSED TIMING
4	0=FIXED PRIORITY 1=ROTATING PRIORITY
5	0=LATE WRITE 1=EXTENDED WRITE X IF COMPRESSED TIMING
6	0=DREQ ACTIVE HIGH 1=DREQ ACTIVE LOW
7	0=DACK ACTIVE LOW 1=DACK ACTIVE HIGH

表 5-3 命令暫存器狀態表

為 8bits 的暫存器，主要是用來控制 DMA 的運作。CPU 可利用此一命令暫存器規劃 DMA 作業模式。當硬體重置(接受到 RESET 訊號)或執行主控制器清除(MASTER CLEAR)指令時，命令暫存器的內含值將被清除為 0。

B0:「X」，沒有作用。

B1:「X」，沒有作用。

B2:「0」-DMA 控制器致能

「1」-DMA 控制器禁能

當 DMA Controller 被禁能時，DMA Controller 僅在閒置狀態，而無法進入動作週期。當在做 DMA Controller 的初始值設定時，最好先將 DMA Controller 禁能，以免發生不必要的錯誤。

B3:「0」-正常時序

「1」-壓縮時序

因為壓縮時序比正常時序少一個脈波週期,所以傳輸資料所需的時間較正常時序短。請見下圖

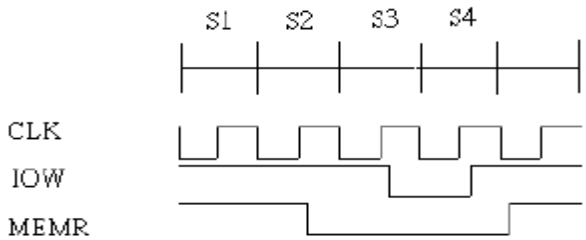

圖 5-2 正常時序圖

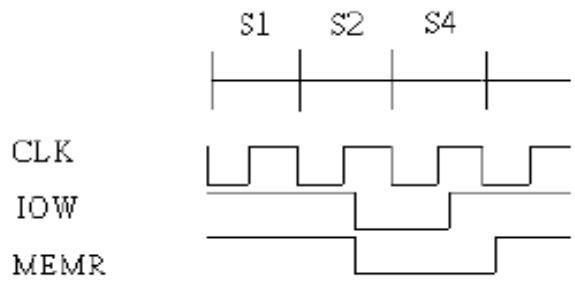

圖 5-3 壓縮時序圖

B5:「0」-寫入信號延後

「1」-寫入信號擴展

如果 B3 為 1，此位元無效。它們之間的差別在於 WRITE 訊號降至低電位的 TIMING 不同。請見下圖。

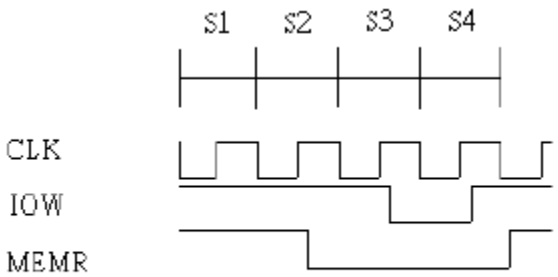

圖 5-4 LATE WRITE

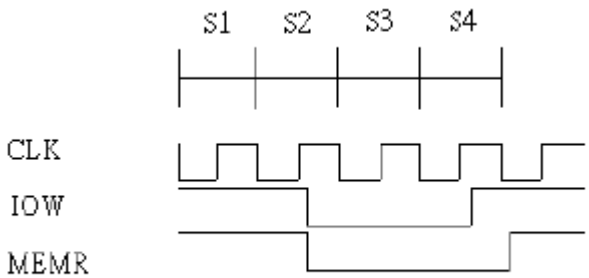

圖 5-5 EXTENDED WRITE

B6:「0」-DREQ 信號高電位動作

「1」-DREQ 信號低電位動作

DMA 請求可以設定為高準位動作或是低電位動作。

B7「0」-DACK 信號低電位動作

「1」-DACK 信號高電位動作

DMA 認知訊號亦可以設定為高電位動作或低電位動作。

MODE REGISTER(模式暫存器)

MODE REGISTER FORMAT	
BIT	DESCRIPTION
1..0	RESERVED
3..2	00=VERIFY TRANSFER 01=WRITE TRANSFER 10=READ TRANSFER 11=ILLEGAL
4	0=AUTO-INITIALIZATION DISABLE 1=AUTO-INITIALIZATION ENABLE
5	0=ADDRESS INCREMENT 1=ADDRESS DECREMENT
7..6	00=DEMAND MODE SELECT 01=SINGLE MODE SELECT 10=BLOCK MODE SELECT 11=UNUSED

表 5-4 模式暫存器狀態表

B1-B0:「XX」-沒有作用。

B3-B2:「00」-驗證傳輸

「01」-輸入傳輸

「10」-讀取傳輸

「11」-不合法

所謂的驗證傳輸,就是於DMA動作週期中不產生任何有效的讀(寫)訊號。因此不會有真正的資料傳輸發生在I/O和記憶體之間。請見下圖。

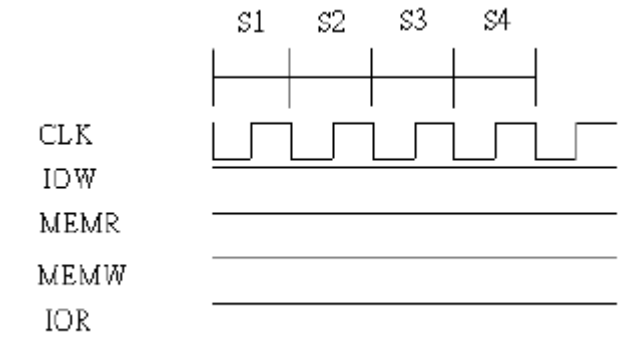

圖 5-6 驗證傳輸

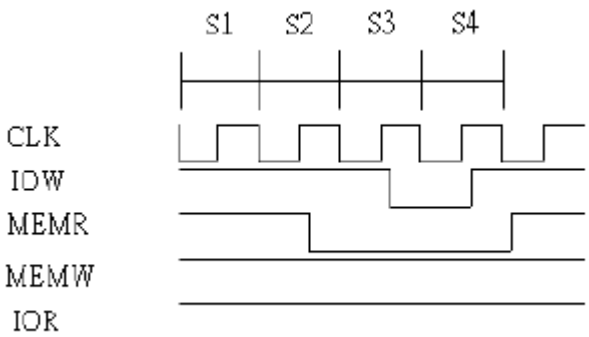

圖 5-7 讀取傳輸

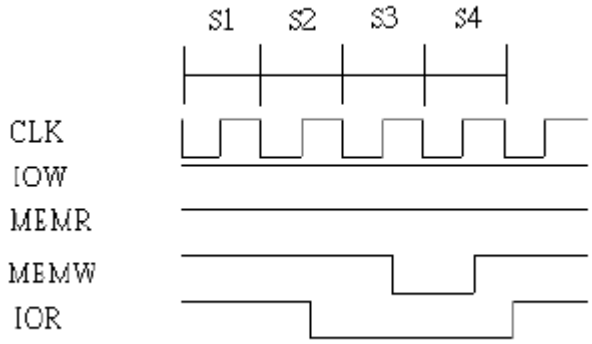

圖 5-8 寫入傳輸

B4:「0」-自動初值設定禁能

「1」-自動初值設定致能

自動初值被致能時，於資料傳輸完畢時，其所設定之原始起始位址及 WORD COUNT 計數值會再度被載入至 CURRENT ADDRESS 及 CURRENT WORD COUNT REGISTER 中，以便下一次重新的資料傳輸。

B5:「0」-位址加一

「1」-位址減一

用來將要讀(寫)的記憶體位址加或減。方便你以不同的方向做 Direct Memory Access。

B7-B6:「00」-選擇要求傳輸模式

「01」-選擇單一傳輸模式

「10」-選擇區塊傳輸模式

選擇不同的 DMA 傳輸模式，將於之後的章節作介紹。

模式暫存器，是用來設用各CHANNEL的傳輸模式。每個CHANNEL都有自己的模式暫存器。但是於DMA暫存器的ADDRESS MAP中，模式暫存只有配到一個ADDRESS。因此，在DMA Controller的設計中加入了一個暫存器，用來指出目前被指定讀(寫)的某個CHANNEL 模式暫存器。每當有讀(寫) 的動作發生時，該暫存器的值會加一，最大至三(indicates CHANNEL 3)，之後回復到0(indicates CHANNEL 0)。因此為了確保，你所要讀(寫)的模式暫存器是正確的。最好在讀(寫)之前，先把索引之暫存器清為0，之後再視要讀(寫)的模式暫存器為何CHANNEL，執行讀(寫)的動作。

請求暫存器(REQUEST REGISTER)

REQUEST REGISTER FORMAT	
BIT	DESCRIPTION
0	0=CHANNEL 0 REQUEST
1	0=CHANNEL 1 REQUEST
2	0=CHANNEL 2 REQUEST
3	0=CHANNEL 3 REQUEST
7..4	XXXX UNUSED

表 5-5 請求暫存器狀態表

除了I/O發出的DREQ之外，CPU可以利用將資料寫入REQUEST 暫

存器的方式，來要求做資料傳輸。利用 4 個 bits 對應到 CHANNEL 0 至 CHANNEL 3。

當值為　　「0」-無 DMA 請求

　　「1」-有 DMA 請求

遮罩暫存器(MASK)

MASK REGISTER FORMAT	
BIT	DESCRIPTION
0	0=CHANNEL 0 UNMASKED 1=CHANNEL 0 MASKED
1	0=CHANNEL 1 UNMASKED 1=CHANNEL 1 MASKED
2	0=CHANNEL 2 UNMASKED 1=CHANNEL 2 MASKED
3	0=CHANNEL 3 UNMASKED 1=CHANNEL 3 MASKED
7..4	XXXX UNUSED

表 5-6 遮罩暫存器狀態表

當 MASK REGISTER 被設定時，則所對應到的 CHANNEL 即使有 DMA 請求，也將不會被 DMA Controller 所接受。

當值為「0」-DREQ 有效

　　「1」-DREQ 無效

狀態暫存器

STATUS REGISTER FOAMAT	
BIT	DESCRIPTION
0	CHANNEL 0 TERMINAL COUNT
1	CHANNEL 1 TERMINAL COUNT
2	CHANNEL 2 TERMINAL COUNT
3	CHANNEL 3 TERMINAL COUNT
4	CHANNEL 0 REQUEST
5	CHANNEL 1 REQUEST
6	CHANNEL 2 REQUEST
7	CHANNEL 3 REQUEST

表 5-7 狀態暫存器狀態表

BIT 0:CHANNEL 0 是否有 DMA 資料傳輸完畢。

BIT 1:CHANNEL 1 是否有 DMA 資料傳輸完畢。

BIT 2:CHANNEL 2 是否有 DMA 資料傳輸完畢。

BIT 3:CHANNEL 3 是否有 DMA 資料傳輸完畢。

BIT 4:CHANNEL 0 是否有 DMA 請求。

BIT 5:CHANNEL 1 是否有 DMA 請求。

BIT 6:CHANNEL 2 是否有 DMA 請求。

BIT 7:CHANNEL 3 是否有 DMA 請求。

狀態暫存器，存放有四個 CHANNEL 的狀態訊息。當 DMA 結束時，相對應的 bit 會被設定為 1，表示該通道有資料傳輸終結。同時若還有

其它的週邊設備有 DMA 請求(尚未被接受)，則相對應的 bit 也會被設定為 1，表示該通道有 DMA 的需要。當狀態暫存器被讀取後，它將被清為 0 。

CURRENT ADDRESS REGISTER 目前位址暫存器

每個 CHANNEL 都有擁有自己的 CURRENT ADDRESS REGISTER，存放的是目前要做資料傳輸的記憶體位址。(週邊設備因為接到固定的 CHANNEL，所以不需去定址它。)

CURRENT ADDRESS REGISTER 是十六 BITS 長度，受限於 DATABUS 的寬度只有八個 BITS。因此當要讀取或寫入時，必需一次讀取一個 byte，通常先對 least significant bits 做存取，再對 most significant bits 做存取。其 most or least significant byte 的選擇是由一個前/後正反器提供。(最好在讀寫前，先清除此一正反器，先對 least significant byte 做讀寫，再對 most-significant byte 做讀寫)

CURRENT WORD COUNT REGISTER 目前字組計數暫存器

每個 CHANNEL 也都具有自己的 CURRNET WORD COUNT REGISTER，它們也是十六個 BITS 長度，其讀寫的方法如同 CURRENT ADDRESS

REGISTER。

當其值由 0h 變為 FFFFh 時，代表所要傳輸的 BYTE 已經傳輸完畢(因此計數器能計數的最大傳輸數量為 512 個 byte。如 ffffh(511)-0h(0)，共有 512 個值)因此，實際傳輸的次數會比載入此暫存器的數值多一次(包含了數值為 0 時)。

BASE ADDRESS & BASE WORD COUNT REGISTER

每個 CHANNEL 都具有自己的 BASE ADDRESS、BASE WORD COUNT REGISTER。長度也皆同為 16 個 bits。

它們是存放 CURRENT ADDRESS、CURRENT WORD COUNT REGISTER 的原始值(也就是你一開始對各 CHANNEL 所設定的值)。當自動初值設定被致能時，當該 CHANNEL 的 DMA 傳輸 terminal 時，BASE ADDRESS、BASE WORD COUNT REGISTER 裡的值，就會分別給到 CURRNET ADDRESS 及 CURRENT WORD COUNT REGISTER，如此一來可省卻你做同樣初始值設定的步驟。

5.4 DMA 控制器指令

為了方便 DMA Controller 的設定，因此對於 DMA Controller 有幾個簡單的指令:

清除前/後正反器(CLEAR FIRST/LAST FLIP-FLOP)

在讀取/寫入新的位址或計數值至 DMA 控制器前執行。目的是為了確保隨後的資料能依序的寫入(讀自)位址暫存器或計數暫存器中。

主控制器清除(MASTER CLEAR)

和硬體重置(RESET)具有同等功能。

5.5 DMA 作業週期

DMA 大致上可分為二種週期，即閒置週期與動作週期。這二種週期都是由好幾個狀態所組成，各狀態均有一個完整的時序週期。

5.5.1 閒置週期:

在沒有任何有效的 DMA 請求時，其維持在不動作狀態，亦既所謂的閒置週期，此時 DMA CONTROLLER 無法做 Direct MemoryAccess 的動作，但是仍可接受來自 CPU 的命令(寫入或讀取)。SI 會一直持續到有 DMA 的請求為止，即持續到 DREQ active 或是請求暫存器有被寫入 dma request。之後進入初始動作週期 S0，並且經由接腳 HRQ 輸出一高能位的信號給 CPU 的 HOLD 接腳，請其讓出系統匯流排控制權，以便讓 DMA Controller 可以進行 Direct Memory Access 的動作。當 CPU 收到 HOLD 信號而尚未送出持有認可信號(HLDA，Hold Request Acknowledge)給 DMA 時,DMA Controller 將一直停留在 S0 狀態,直到收到高電位的 HLDA 信號為止。

5.5.2 動作週期:

當 DMA 收到持有認可信號(HLDA)之後，即開始進入 DMA 的動作，其中 S1、S2、S3、S4 為 DMA Controller 運作時的時序狀態週期。如果 DMA Controller 工作週期需要延長才能完成傳輸的工作時(即需要較長的寫入或是讀取 timing)，DMA Controller 允許在 S2 或 S3 與 S4 插入一個以上的等候狀態(延長讀取、寫入 timing)。若 DMA Controller 的 READY 輸入接腳為低電位時，其即插入等候狀態，直到 READY 輸入為高電位為止。

當 DMA 在進行 I/O 設備到記憶體(記憶體到 I/O 設備)間的資料傳輸時，亦即執行 Direct Memory Access 的動作時，IOR 與 MEMW(IOW 與 MEMR)會依讀出或是寫入的需求而對應到正確的高、低準位(如是寫入動作，則做用的信號線為 IOR 與 MEMW，反之若為讀取動作，則做用的信號線為 MEMR 與 IOW)。注意，當 DMA 在進行 I/O 設備與記憶體之間的資料傳輸時，資料並不會寫入或讀自 DMA，而是直接在二者之間游走(靠 DMA Controller 控制讀寫信號)。

5.6 DMA Controller 控制資料傳輸的模式:

它可以有三種傳輸模式，分別為單一傳輸模式、區塊傳輸模式、要求傳輸模式。

5.6.1 單一傳輸模式(Single Transfer Mode):

所謂單一傳輸模式，簡單的說就是 DMA Controller 每次僅做一筆資料(one byte)的傳輸，之後便結束 Direct Memory Access 的動作週期，回到閒置週期，然後將系統匯流排控制權還給 CPU。然後再發出 Hold Request 給 CPU，等待 CPU 再一次讓出系統匯流排的控制權。在這個模式下，每當傳輸完一筆資料時，字組計數器便會自動減一。當字組計數器由值 0h 減一變為 FFFFh，表示傳輸完畢，隨即結束 Direct Memory Access 動作週期，並使 NEOPOUT 輸出為低電位，以示 Direct Memory Access 的結束。

於此一模式下，DREQ(Direct Memory Access Request)信號至少必須持續到 DACK(Direct Memory Access Acknowledge)信號被週邊設備接收為止，意即，DREQ 為高電位的時間必須保持到 DACK 信號變成高電位為止(假設 DREQ 與 DACK 皆設定為高電位動作)。當 DACK 變成高電

位且單筆資料完成轉移之後，若 DREQ 信號依然處於動作狀態時，DMA 不會去理它，而會自動的令 HRQ 變為不動作狀態，並歸還系統控制權給微處理器，以完成單一傳輸的動作。如果 DREQ 仍為高電位的話，則另一次的 DMA 請求，必須等到 CPU 執行完一個完整的機器週期後，才會再發生。換言之，在此一模式下，如果 DREQ 一直維持在高電位下，則 CPU 在每一 DMA 的傳輸過程中，都會有一完整的機器週期去執行指令。避免 DMA Controller 一直佔用系統匯流排。

5.6.2 區塊傳輸模式(Block Transfer Mode):

於此模式下，DREQ 要持續到 DACK 認可信號動作。意即，當 DACK 變成高電位後,DREQ 信號可為高或低電位(假設 DERQ 與 DACK 皆設定為高電位動作)。當要求系統匯流排控制權的請求信號(HRQ)被認可後(收到 HLDA)，DMA Controller 隨即佔用系統匯流排，然後開始做區塊資料的傳輸，一直到終止計數(TC)或外部 NEOP 的信號產生為止。也就是說，當 DMA Controller 得到系統匯流排的控制權之後，就一直使用匯流排做資料傳輸，直到整個資料傳輸完畢才歸還系統匯流排給 CPU。於此模式下，若 DMA 自動初值設定被啟動，則在資料傳輸結束或中止時，DMA 會自動進行初值設定動作。也就是恢復開始傳輸之前的各項設定值。

5.6.3 要求傳輸模式(Demand Transfer Mode):

類似區塊傳輸。於此模式下，資料會連續不斷的傳輸，直到整個資料被傳輸完畢，或是接收到外部 NEOP 信號。特別的是，當 DREQ 請求信號解除，也會終止 DMA 的動作週期。當 DREQ 信號解除後，若資料傳輸尚未完畢，則當時的位址與字組計數值將各自存於 DMA Controller 裡頭的目前位址暫存器(current address register)與目前字組暫存器(current count register)中，等到 DREQ 再請求時，再從先前被暫停的地方繼續 Direct Memory Access 的傳輸。

注意，於此模式下，若要中止 Direct Memory Access 的動作週期，則 DREQ 在 S4 之前必須解除，才能準確地停止 Direct Memory Access 動作週期。。

5.7 I/O 與記憶體間資料傳輸

DMA 控制器擔任 I/O 與記憶體之間傳輸資料的工作。二者之間的資料傳輸不外乎為讀取，寫入，以及驗證三種。

當資料傳輸方向為,由記憶體至 I/O 設備時,為一讀取傳輸(memory read ， I/O write)，此時 DMA 會令 MEMR 與 IOW 動作以完成資料的傳輸。當資料傳輸方向是從 I/O 至記憶體時，為一寫入傳輸(I/O read ，memory write)，此時 DMA 會令 IOR 與 MEMW 動作，以完成寫入傳輸。至於驗證傳輸則為一虛擬傳輸，即 DMA Controller 在執行傳輸時，只產生位址及 EOP 等信號，而對記憶體與 I/O 的控制信號則不動作。因此不會有資料的傳輸產生，不用擔心不必要的資料傳輸被啟始。

DMA 還具有自動初始值設定的功能,只要令模式暫存器中的第四個 BIT 為 1 即可。若將自動初值設定 enable，則當處理終了(NEOPOUT)信號產生後,DMA 隨即將剛才做用 CHANNEL 的位址與計數暫存器的內含恢復為原設定值，然後只要再偵測到有效的 DREQ 信號後，又可開始另一次的 Direct Memory Access。如果 DMA Controller 的自動初值設定功能為 disable,則當資料傳輸結束後,所產生的 NEOP 信號將會停止 DMA 的工作。此時，若欲再起動 DMA Controller 工作，則必須重新設定一

次。

DMA 另外還具有一種可提高資料傳輸速度，進而加強系統工作效率的功能，那就是採用壓縮時序(COMPRESSED TIMING)。一般的資料傳輸通常需要三個狀態，S2、S3、S4 週期來完成(state S1 的需要與否則視位址的八個 most-significant bits 是否需要改變而定)，但若採壓縮時序下卻只需二個狀態週期(S2、S4)就可完成。其中 S2 是用來改變位址，而 S4 則用以執行讀取/寫入的工作。注意，若位址 ADDRESS[15..8] 需要更新時，則 S1 的狀態仍需被執行。

第六章 DMA 之設計與實作

我將此次的專題分成數個部份設計，最後再整合成同一個 PROJECT。

6.1 索引機制

一開始我們可以先設計二個索引機制。一為讀(寫)ADDRESS 及 WORD COUNT REGISTER 的 least or most-significant byte 的索引 flip-flop。當有讀(或寫)的動作產生時，這個索引值就由 1 變 0，0 變 1，交替地運作。(當為 0 時就指到 least-significant byte，反之則為 most-significant byte)。

其二為模式暫存器的索引製作。同樣地每當有讀(或寫)的動作發生時，其值就逐步加一，當其值為三時，再加上一就回復至零。用來索引四個 CHANNEL 模式暫存器之一。

6.2 FIXED OR ROTATING PRIORITY DMA REQUEST

這個部份是要設計出，可以接受高電位動作或是低電位動作的 DREQ。同時依據所設定為 FIXED 或 ROTATING 的優先權判定，將 DMA 的 SERVICE 給到具有不用優先權的週邊設備。之後再依據 DACK 是為高電位動作或是低電位動作來作輸出。

於 FIXED 優先權時，CHANNEL 0 具有最高的優先權，CHANNEL 3 的優先權則為最小。當 CHANNEL 1 和 CHANNEL 2 同時發出 DMA REQUEST 時，DMA SERVICE 會給具較高優先權的 CHANNEL 1。

採用 ROTATING 優先權，則可以避免具較高優先權的週邊設備一直佔用 DMA SERVICE。於此優先權判定機制下，用過 DMA SERVICE 的 CHANNEL 給定最小的優先權。例如，現在的優先權順序為 CHANNEL 0、CHANNEL 1、CHANNEL 2、CHANNEL 3(由高至低)，此時 CHANNEL 1 發出 DMA REQUSET，並得到 DMA SERVICE，則優先權的順序將會變為 CHANNEL 2、CHANNEL 3、CHANNEL 0、CHANNEL 1。(由高至低)

6.3 DMA Controller 的 FINITE STATE MACHINE 設計

我們在這個時期規劃出整個 DMA Controller 運作時 STATE 的變遷情況。設計了 STATE I，STATE WI，STATE RI，STATE S0，STATE W0，STATE R0，STATE 10，STATE 1，STATE 2，STATE 3，STATE 4。

在 STATE I 時，DMA 是處於閒置狀態，等待有效的 DMA REQUEST 發生，就進入到 STATE 0。在這個時候 DMA 可以接受 CPU 的讀、寫動作。當 CPU 讀取內部暫存器時，就進入 STATE RI，將定址到的暫存器內容放至 DATABUS 上，隨即回到 STATE SI。

同理當 CPU 寫資料至 DMA Controller 內部暫存器時，則進入 STATE RI，將 DATABUS 上的資料寫入定址到的暫存器，隨即回到 STATE SI。

當接受到 MASTER CLEAR 的指令時，進入 STATE S10，將命令、狀態、請求暫存器及索引暫存器清為 0。及將 MASK REGISTER 設定 1

在 STATE S0 時，DMA Controller 將 HRQ 升至高電位，要求 CPU 釋放出系統匯流排的控制權。在 STATE S0 時，CPU 仍可以對 DMA Controller 內部暫存器做讀或寫的動作。當為讀的動作時，進入 STATE R0，將所定址之暫存器內容放至 DATABUS，供 CPU 抓取。隨即回復至 STATE S0。

當為寫入的動作時，則進入 STATE W0，在此狀態下抓取 DATABUS 上的資料(上升緣時抓取)。隨即回復至 STATE S0。

等到 CPU 回送 HLDA 時，表示 CPU 已將系統控制權交給 DMA Controller，則進入 STATE S1。

當接受到 MASTER CLEAR 指令時，進入 STATE S10。

於 STATE S1，則下一狀態進入到 STATE S2。

於 STATE S2，下一狀態可能(1)維持在 STATE S2，進入到(2)STATE S3 或(3)STATE S4。當 READY 訊號為低電位時,STATE 維持在 STATE S2(即加入 WAITING STATE，延長讀取時間)。否則當 DMA 的傳輸 TIMING 設定為壓縮模式時，則 STATE S3 就會被省略，直接跳至 STATE S4。若為一般 TIMING，則是進入 STATE S3。

於 STATE S3，下一狀態可能維持在 STATE S3 或進入到 STATE S4。當 READY 訊號為低電位時，STATE 維持在 STATE S3(即加入 WAITING STATE，延長寫入時間)。否則進入 STATE S4。

於 STATE S4，下一個狀態的變遷則較為多樣。得視，情況之不同而定：

如，傳輸模式設定為”單一傳輸”，則下一狀態則進入 STATE SI，等待下一次 CPU 交付系統控制權。

如，傳輸模式設定為”區塊傳輸”模式，則下一次的狀態變遷可能為 STATE S1、STATE S2 或 STATE SI。當外部的位址拴鎖器內容需要改變時，則進入 STATE S1。例如，當 CURRENT ADDRESS REGISTER 的內含值為 ffffh，而其位址變化是往上(也就是將目前位址加一，得到下一次資料傳輸的位址)，此時，會有進位產生，所以需要將外部位址拴鎖器的內含值改變。因此得進入 STATE S1。同樣地，當 CURRENT ADDRESS REGISTER 的內含值為 0000h，而位址變化是往下(也就是將目前位址減一，得到下一次資料傳輸的位址)。也需進入 STATE S1，改變外部位址拴鎖器的內含值。

當全部的資料傳輸完畢時，則進入 STATE SI。

6.4 輸出接腳設計

ADSTB

當在 STATE S1 時，位址的 most-significant byte 會出現在 DATABUS。接著，ADSTB 升至高電位，維持半個 clock。用以將 most-significant byte 寫入外部位址拴鎖器中。

AEN

當 STATE 是處於 S1、S2、S3、S4 時為高電位，表示位址有效。

DBEN

當 STATE 為 RI 或是 R0 時，DBEN 升至高電位，用以通知 CPU 可以抓取正確的資料。

HRQ

當 STATE 為 S0 時，便升至高電位，要求 CPU 釋放系統匯流排。於 STATE 再度回到 SI 時，降至低電位。

NEOPOUT

當 DMA 傳輸完畢時，輸出一高電位。維持一 clock 再回到低電位。

IOR

於閒置週期時，IOR 接腳是被 CPU 控制，用以讀取 DMA Controller 內部暫存器。而在動作週期時，若為寫入傳輸，則 IOR 會在 STATE S2 降為低電位。於 STATE S4 升回高電位。

MEMR

在動作週期時，若為讀取傳輸，則 MEMR 會在 STATE S2 降至低電位，於 STATE S4 升回高電位。

IOW

在閒置週期，它被 CPU 用來將資料寫至內部暫存器。而於動作週期時,若為讀取傳輸,則 IOW 會在 STATE S2(壓縮時序或 EXTENDED WRITE 下)或 STATE S3(正常時序下)降至低電位，於 STATE S4 回復到高電位。

MEMW

於動作週期，當為寫入傳輸，則 MEMW 會在 STATE S2(壓縮時序或 EXTENDED WRITE 下)或 STATE S3(正常時序下)降至低電位，於 STATE S4 回復至高電位。

DACK

當 DMA 動作週期開始時，依據所設定是為高準位動作或低準位動作，輸出相對應的 DMA 認知訊號給週邊設備。

6.5 成果驗證

以下我只展示出三種 DMA 的傳輸模擬時序圖。分別為 Single、Block、Demand 三種傳輸。為了方便，我把設定 DMA Controller 暫存器的順序固定，唯其寫入值不同(您可以參考 5.3 章節，了解各功能暫存器的內容值代表意義)。

第一次的 I/O write,我們寫入 CHANNEL 1 的 Base/Currnet Address Register(I/O 位址為 02h)之 least-significant byte 部份。

第二次的 I/O write,我們寫入 CHANNEL 1 的 Base/Currnet Address Register(I/O 位址為 02h)之 most-significant byte 部份。

第三次的 I/O write，我們寫入 CHANNEL 1 的 Base/Currnet Word Count Register(I/O 位址為 03h)之 least-significant byte 部份。

第四次的 I/O write，我們寫入 CHANNEL 1 的 Base/Currnet Word Count Register(I/O 位址為 03h)之 most-significant byte 部份。

第五次寫入，則設定 Command Register(I/O 位址 08h)。

第六次寫入，則設定 CHANNEL 1 的 Mode Register(I/O 位址 0bh)。

第七章　　心得與討論

從大二開始上數位電路設計的課程時，就對硬體的電路設計產生了興趣，後來到了大三陸陸續續接觸到有關單晶片設計、VLSI 等硬體相關的課程，便決定專題想做跟硬體有關的領域。於是後來就跟者陳德生老師學習 VHDL 的電路設計。

一開始花了很多時間去熟悉、了解 VHDL 的特性、功能，以及 MaxPlus II 的使用，也從圖書館、網路以及學校所開設的硬體課程中得到一些訊息及資料，花了不少時間在資料的研讀上。尤其感謝政芳學長的大力幫忙與指導。在較熟悉 VHDL 後發現在設計電路時，發現不必再像大二修習數位電路設計時那樣使用各種邏輯閘來兜出所想要的電路，現在只要依照 VHDL 程式的語法，描述出電路的特性，他便可幫我們架構出電路，節省了很多時間。而且在電路的驗證上也方便很多，不必再像以前一樣慢慢去測是哪個電路元件或是哪一條接線有問題，這些都可以由 CAD 幫我們代勞。只要輸入值，便可以透過 siumation 來 check 某一段邏輯閘的結果是否正確，真是方便極了呢!

充分了解DMA Controller的運作方式及規格之後，便要開始VHDL

的coding，一開始的時候，真有點不知如何下手的感覺，後來透過設計出它的控制單元(FINITE STATE MACHINE的設計)，將所有的控制訊號依附在它的狀態轉換，在正確的TIMING下，將訊號做正確的變化。才慢慢將其完成。在此期間，感謝陳德生老師定期抽空和我們MEETTING以及相關方面的指導。

參 考 資 料

[1][儒林]使用 VHDL 電路設計語言之數位電路設計（作者:林傳生）

[2][文魁]VHDL 與數位電路設計（作者：盧毅、賴杰）

[3][儒林]8086 微處理機之界面技術與應用（作者：劉銘中、嚴偉誠、黃素梅 編著）

[4][Addision Wesley] The Indispensable PC Hardware Book（作者:Hans-Peter Messemer）

[5][Mc Graw Hill]VHDL（作者:ZAINALABEDIN NAVABI）

[6][碁峰]計算機組織與設計 軟應體介面 中譯本(作者：Patterson Hennesey）

[7][碁峰]微電腦界面技術與實作 （作者:陳瑞熙 高志堅 鄭明哲）

[8][儒林]微電腦介面技術 （作者:于英民 莫瑋 孫全）

[9][復文]微處理機硬體接界技術及應用（作者:BREY)(鐘鴻源 洪正瑞 譯）

[10][儒林]微電腦界面實作 （作者:Edward J. Pasahow)(羅志正 譯）

[11][國家]微處理系統設計 （作者:Edwin E. Klingman)(趙元

山 譯）

[12][全華]CPLD 數位電路設計發展應用（作者：林容益）

[13][其他]微處理機聖經講義（作者:徐弘洋）

[14]http:\\www.altera.com