第一章
緒論

近年來，由於資訊的日新月異，以及網際網路的蓬勃發展，「網路」這一名詞似乎已成了日常生活中不可或缺的一份子。也由於網路快速的成長，網路上的節點也以極高的速度成長中，但因為每一個上網的節點都需要一個全世界唯一的位址，因此網際網路位址早已面臨不敷使用的狀況。

現行的網際網路位址架構為 IP(Internet Protocol)或 IPv4(Internet Protocol version 4)[1]，長度為 32 個位元，也就是說在目前的網路架構中，所有能用的位址總數共有 2^{32} = 4294967296，且在扣除一些保留及特殊的位址之後，真正能使用的位址就更少了。隨著電腦及網路的日益普及，二十一世紀將是「E－世代」，每個人的家中都將有固接式網路，只要一開電腦便可立即上網，享受高品質網路所帶來的便利。屆時，在家辦公、開會，自行架設網站，或是坐在電腦前觀看著一場高品質、立即轉播的球賽，都將不再是神話。但前提是每一台電腦都有自己的 IP 位址，就如同每一個人的家中都要有住址才能讓每一封信件能夠安全無誤地送到目的地一樣。因此面對呈爆炸性成長的網路節點數目，IP 位址的不足的確是刻不容緩亟待解決的一大挑戰。

由於目前的網際網路架構已經相當龐大，如果驟然以另一種協定或方法來取代或改進目前所用的 IPv4，則必須同時更換許多網路設備(路由器、交換器，…，等)，不但將造成大部份人的不便，以商業的角度來看也完全不合乎成本。因此面對 IPv4 位址不足的現況，不僅要發展一套新的通訊協定以解決此問題，更重要的是能不能以過渡的方式，慢慢的取代現有的 IPv4。所以新的通訊協定必須具備跟原有的 IPv4 協定共存的特性，並能解決網路位址不足的問題，以此目標發展出來的新一代網際網路協定就是 IPv6(Internet Protocol Version 6)[2]。

雖然 IPv6 能夠解決網路位址不足的問題，但因為它有過渡期的痛

苦，因此其他試圖解決網路位址不足的方法也應運而生，NAT (Network Address Translation)[3]就是其中的代表。NAT 與 IPv6 兩者最大的不同處是：NAT 是在原有的架構下，利用一些技術，使得某些保留的 IPv4 位址得以重覆使用，但還是有其難以達到的功能，如電腦的 IP 位址將不固定，雖然一般用戶端(client)同樣能夠運作無誤，但對於需要永久固定位址的伺服器(server)便會造成困擾。而 IPv6 則是對於現有的 IPv4 架構做了些許改進及修正，因此能在不影響現有 IPv4 網路之下提供所有網際網路帶來的便利。

除了 IP 位址的不足造成下一代的網路勢必要走向 IPv6 之外，現在的網際網路還面臨了另一個狀況。由於網際網路的規模越來越大，因此之前完全無政府無管理的狀態帶給網路用戶的許多問題的影響範圍也日趨增大，慢慢的脫離了可以忍受的範圍。這些問題如不請自來的垃圾信件、各式各樣的網路攻擊以及程度參差不齊的使用者等。如果不解決這些問題的話，不但網路的頻寬再怎麼成長仍然無法跟上這些不必要的浪費，而且隨之而來的管理與設備成本也令人無法忍受。QoS(Quality of Service)的觀念隨著頻寬的不足與連續大量資訊傳輸的應用，如線上播放影片等的興起而逐漸盛行。這些都讓網際網路需要更多的管理已達成更好的使用品質。不管是企業、家庭甚至 ISP (Internet Service Provider)都需要被管理過的網路，以達成更好的使用品質、獲得穩定的頻寬、以及保證網路的安全，因此這些網路用戶需要將具管理功能的網路設備置於網路出口。因此，傳統針對 OSI 七層中的第二、三層進行的網路交換與路由決定已經不能符合現代網路的需要，而需要針對第四層或更高層的封包標頭(Header)進行分析並加以處理。但由於第四層以上的封包標頭較第二、三層複雜，且能在上面發展的應用更多，因此此一課題也吸引了許多廠商或研究單位投入，希望能帶來更好的網路品質。

本論文將在第二章中針對 IPv6 的封包格式探討其特性，並應用於 IPv6 封包分類器之中。之後各章將陸續介紹 IPv6 封包分類器的方法、架構、實作與測試結果，最後為本論文之結論與參考資料。

第二章
相關研究

本論文所探討的IPv6封包分類器有許多先前的研究成果。在IPv6方面，IETF[4]所制訂的數個RFC，如RFC2460[2]介紹了IPv6的整體規範，RFC2373[5]、RFC2374[6]與 RFC2375[7]分別定義位址架構及兩種主要的位址使用方法等，這些RFC清楚定義了IPv6的封包格式及如何使用。上層軟體的標準也有制訂，如 RFC2080[8]中所制訂的RIPng即為在IPv6上的路由交換協定。RFC1933[9]以及RFC2529[10]則定義了IPv6與IPv4間要如何共存，以及如何自原有之IPv4網路轉換至IPv6網路。對於封包分類而言，第三層的封包查詢方法為最早的相關問題，[11][12][13][14][15][16]等論文都提供了第三層的封包查詢方法，而封包分類的方法則見於[17][18][19][20][21][22]等論文中。這些論文提供了一般的封包查詢或分類的最壞狀態保證。不論針對封包查詢或封包分類，除了一般使用隨機存取記憶體的解決方法以外，其他的技術也一一的被提出，例如Trie[23]或CAM[24]，而[25]為加速CAM的封包分類速度的方法。此外，整體的封包分類系統設計可參考[26]，利用NetBSD[27]實作出一完整之下一世代路由器的模型。[28]則設計一IPv6交換式路由器，根據封包的第三層資訊進行交換，並實作於PC平台之上。

由以上的研究結果可歸納出，第三層的封包查詢與第四層以上的封包分類這兩種問題雖然有其類似的地方，並且可以使用相近的演算法。但由於封包分類所需的資訊有數個欄位，與封包查詢的單一欄位複雜度相差甚大。另外，封包查詢的評斷方式為速度，但封包分類則有許多的評量方法，速度並非唯一的取決因素。在問題的本質方面，封包查詢為標準的最長字首查詢(Longest Prefix Match)，而封包分類是以完全比對(Exact Match)為主，搭配最長字首查詢。因此雖然第三層的封包查詢技術已經漸漸成熟，針對第四層的封包分類卻還需要進

一步的研究。目前針對第四層以上的一般性封包分類方法，有以下幾個特點：

- 對於兩個欄位以下的封包分類可獲得相當良好之成果，或者，
- 只在數目不多(小於五十)的封包分類器下可獲得良好的成果，或，
- 無法準確預期封包分類的所需時間，或，
- 太慢或使用過大的儲存空間。

且這些演算法幾乎都用理論值加以模擬，而非實際以軟硬體實作來觀察執行的速度，故無法實際得知實際執行所耗費的時間。本論文將針對以上的整理結果，實作出硬體的封包分類模組及整體系統。

第三章
IPv6 封包的格式與特性

3.1 IPv6 封包標頭格式

IPv6 與 IPv4 有許多不同之處，圖 3.1 與圖 3.2 為 IPv6 與 IPv4 的封包標頭示意圖。

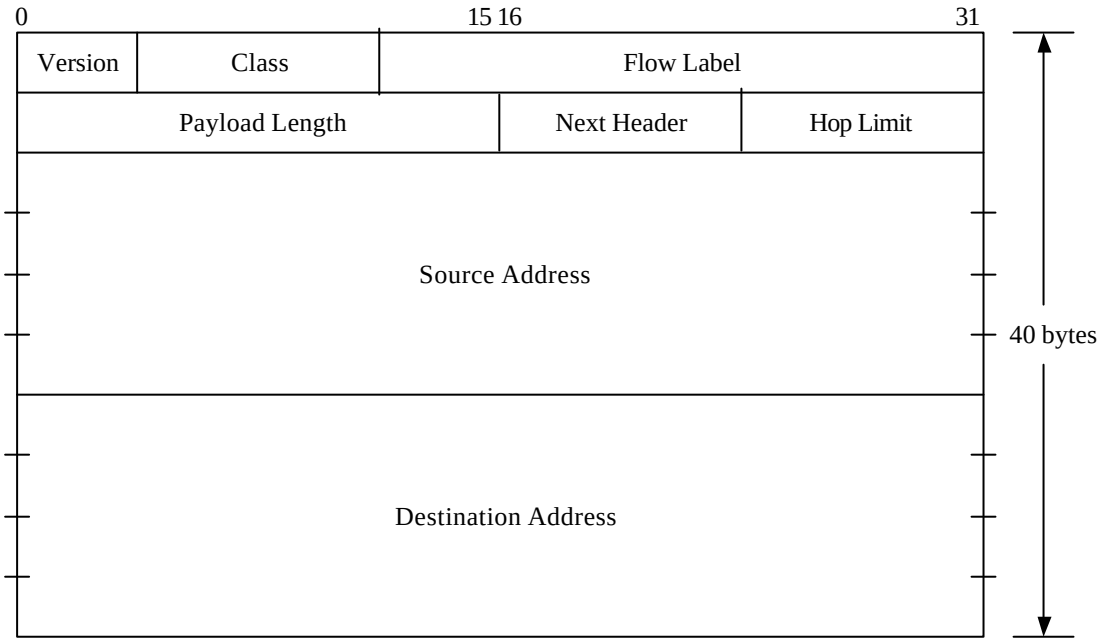

圖 3.1 IPv6 封包標頭

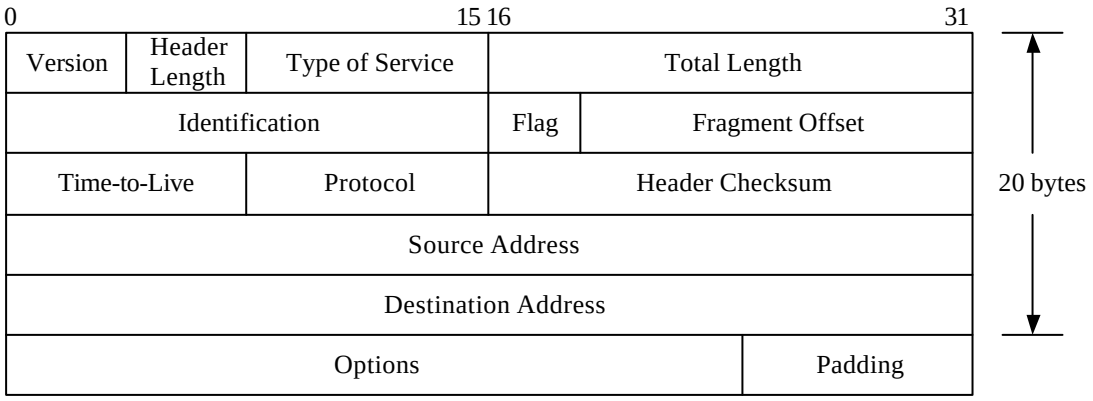

圖 3.2 IPv4 封包標頭

藉由圖 3.1 與圖 3.2 的標頭格式，可以清楚的看出，IPv6 封包與 IPv4 封包標頭中的欄位有很大差異，同時也可發現 IPv6 封包標頭比 IPv4 簡易。IPv6 封包標頭只有六個固定欄位與兩個位址，而 IPv4 封包標頭有十個固定欄位、兩個位址和 Options。其中，只有版本(Version)欄位維持相同的意義與位置。為了使 IPv4 與 IPv6 封包能夠同時在網路上運作，可藉由版本欄位來判別封包格式，但事實上它的作用並不大，因為在第二層 MAC 訊框的型別(Type)欄位中就可辨別出封包格式，如：Ethernet 的 MAC 訊框的型別值若是 0x8000 為 IPv4，0x86DD 則為 IPv6。IPv4 封包標頭中共有六個欄位被捨棄，分別是：Header Length、Type of Service、Identification、Flags、Fragment Offset 和 Header Checksum。三個欄位只是更換名稱，但性質與功能一樣，分別是：Total Length、Protocol Type 和 Time-to-Live。

另外，IPv6 新增了兩項欄位－Class、Flow Label。這兩項欄位的目的是為了提升 QoS，以處理「即時資料」(Real-time Traffic)。Class 欄位共 16 個位元，用來區分不同等級的封包，並按照欄位內數字的大小，區分封包的重要性。而 Flow Label 欄位的目的是使同一資料流(Flow)的封包能得到相同的對待。截至目前為止，這兩個欄位的內容尚未正式定義。

在位址結構上，IPv6 捨棄 IPv4 的 Class A、B、C、D 與子網路遮罩結構。但基本上，仍有許多不同位址類型且各有其意義，如：Unicast、Multicast 等等，能夠應用在各種特殊的領域上。

3.2 IPv6 位址特性

在以上與 IPv4 的欄位差別外，IPv6 的 IP 位址長度為 128 位元，如此的長度是為了應付日後網路龐大的成長空間，以避免不敷使用的情形。在目前有較完整定義的 Aggregatable Global Unicast Address[6](簡稱 Unicast 或 Unicast Address)共 128 位元當中，前半段的 64 位元是紀錄 Prefix，並且用 Prefix 的內容及長短來分辨不同的位置。後半

段的 64 位元則是存放 EUI-64 位址。Aggregatable Global Unicast Address 為 IPv6 全域位址，如果想要跟遠方電腦溝通，就至少必須具備一個全域位址。

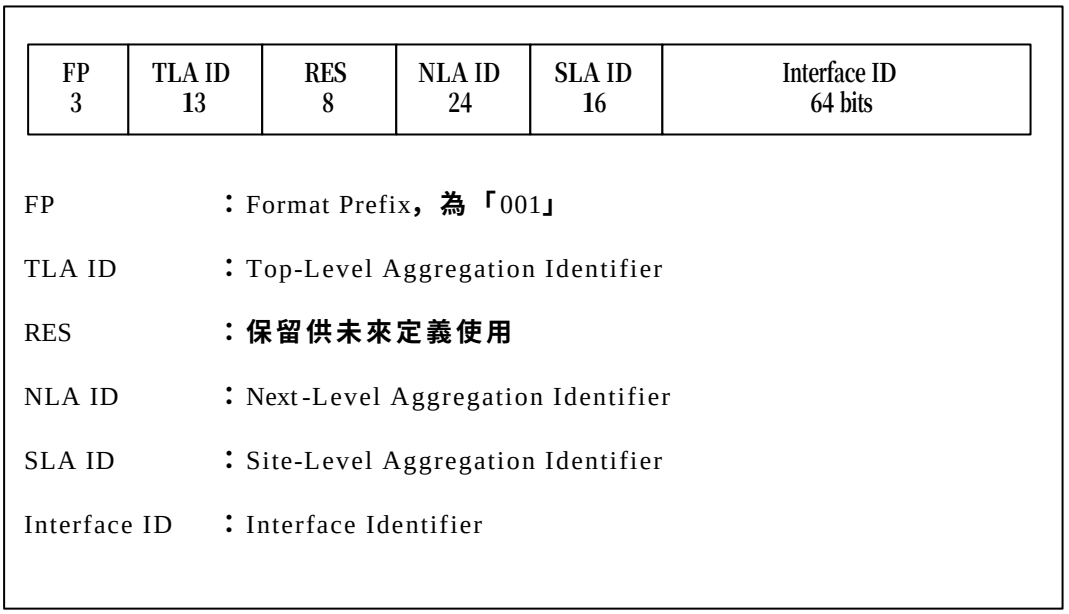

圖 3.3 Aggregatable Global Unicast Address

IPv6 利用不同的前置位元來分辨不同的網路區域，進而用於封包轉送上。在 Unicast Address 中，除了前半段的 prefix 以外，後半段為根據 IEEE EUI-64 標準所定出來的 Interface ID，不論使用任何網路介面都會有一個唯一的 EUI-64 Interface ID。由於 IPv6 的址長達 128 位元，所以在應用上很容易因為長度太長而造成記憶體存放不易。因此可以根據用於何種實體網路介面而將此 Interface ID 轉換回原來的表示方法，以節省記憶體空間。例如本論文實作於 Ethernet，所以可以將此 64 位元長的 ID 轉換為 Ethernet 的 48 位元 MAC 位址。此部分會於第三章中詳述。

3.3 IPv6 特性與封包分類

在介紹如何將 IPv6 封包分類之前，有必要先對封包分類作一明確而完整的定義。國際標準組織(International Standard Organization,

ISO)將資料通訊網路分為七層[29]，分別為：

1. 實體層(Physical layer)：以網路設備的硬體實現實際的信號。
2. 資料連結層(Data-link layer)：處理固定長度的封包。
3. 網路層(Network layer)：連接通訊網路與封包的傳送。
4. 傳遞層(Transport layer)：網路低階存取與使用者訊息傳送。
5. 會議層(Session layer)：實現會議、或行程間的通信協定。
6. 表現層(Presentation layer)：解決網路節點間的格式轉換。
7. 應用層(Application layer)：負責直接與使用者接觸的工作。

但因為 ISO 發展出來的此一規則較晚出現，因此分層較簡單的 TCP/IP 協定堆疊(Protocol stack)[30]，如圖 3.4 所示，仍為目前網路的主流，比起 ISO 的規則而言它則是更加的有效率。

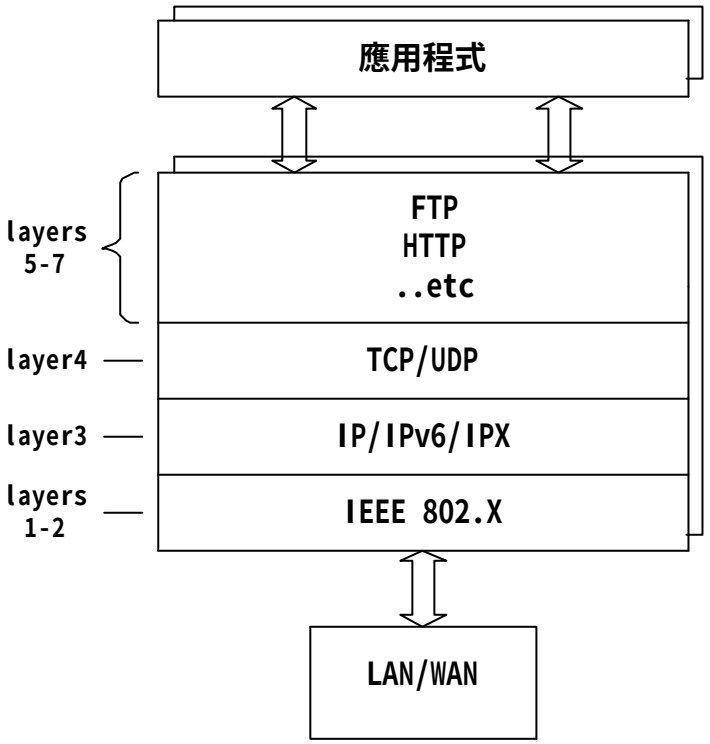

圖 3.4 TCP/IP 協定堆疊

目前網際網路中的交換設備多為 layer2 的交換器(Switch)或是 layer3 的路由器(Router)，這兩種設備主要是根據 MAC 或 IP 位址來進行封包的處理與交換，並且均以目的地位址作為處理與交換的依據，也就是說只負責封包的送達而並不管由何地何機器送出。有鑑於此，本論文將 TCP/IP 標頭中的五個欄位取出作為將封包分類的依

據，這些欄位為：

- Source IPv6
- Destination IPv6
- Source port(SP)
- Destination port(DP)
- Protocol(Pro)

前兩個欄位代表封包的來源以及送往何處，第三以及第四個欄位的內容則可以根據 well-known port[31]來區分封包所要求的服務為何，例如 80 通常為 WWW(World Wide Web)或 25 代表 SMTP(Simple Mail Transfer Protocol)等。第五個欄位可分辨封包內所使用的協定，例如 TCP。許多即時的應用，例如線上播放電影會持續一段時間的產生標頭相同的封包，這些封包可歸類為同一資料流，如果將相同的資料流利用相同的 ID 加以辨別，例如 IPv6 位址中的 Flow Label 欄位，則可省下許多判別欄位內容的時間。規定了五個欄位的內容之後，即可訂定封包分類之規則，如表 3.1 所示：

Source IP	Destination IP	SP	DP	Pro	說明
*	*	*	*	*	所有封包
*	www.spam.com	*	*	*	所有往該網站的封包
*	*	*	*	TCP	所有 TCP 協定的封包
*.spam.com	Mycom.com	*	25	*	從特定網域到我方網域的所有郵件遞送
Server.com	Host.my.com	80	*	*	所有自 Server 回傳的 WWW 訊息
*	*	*	*	UDP	所有 UDP 協定的封包

表 3.1 封包分類規則範例

從表 3.1 的例子可以看出，規則中可能會有許多的萬用字元，因此不可能用 0 或 1 來表示每一條規則，為了解決這個問題，必須使用 0 與 1 之外的第三態『Don’t care』或『X』來表示萬用字元，同樣的，

儲存這些規則的記憶體也必須要有支援第三態的能力。例如『0XXXX』表示『0 到 31 的所有值』，或將整個欄位的內容填入『X』以表示不管此欄位的內容為何。

定義封包分類的規則之後，就可以利用這些規則將網路上的封包加以分類，並對封包進行動作與管理，本論文將此動作稱為 Action。Action 的存在讓封包的分類能真正的帶來效用，例如將某網站送來的封包全部丟棄、將所有連線電玩的連線優先權加以調低，或為線上播放影片保留頻寬等等都是可能的 Action 內容。

第四章
IPv6 封包分類器之方法與架構

4.1 系統架構

以下為本論文所研發的整體系統架構圖。

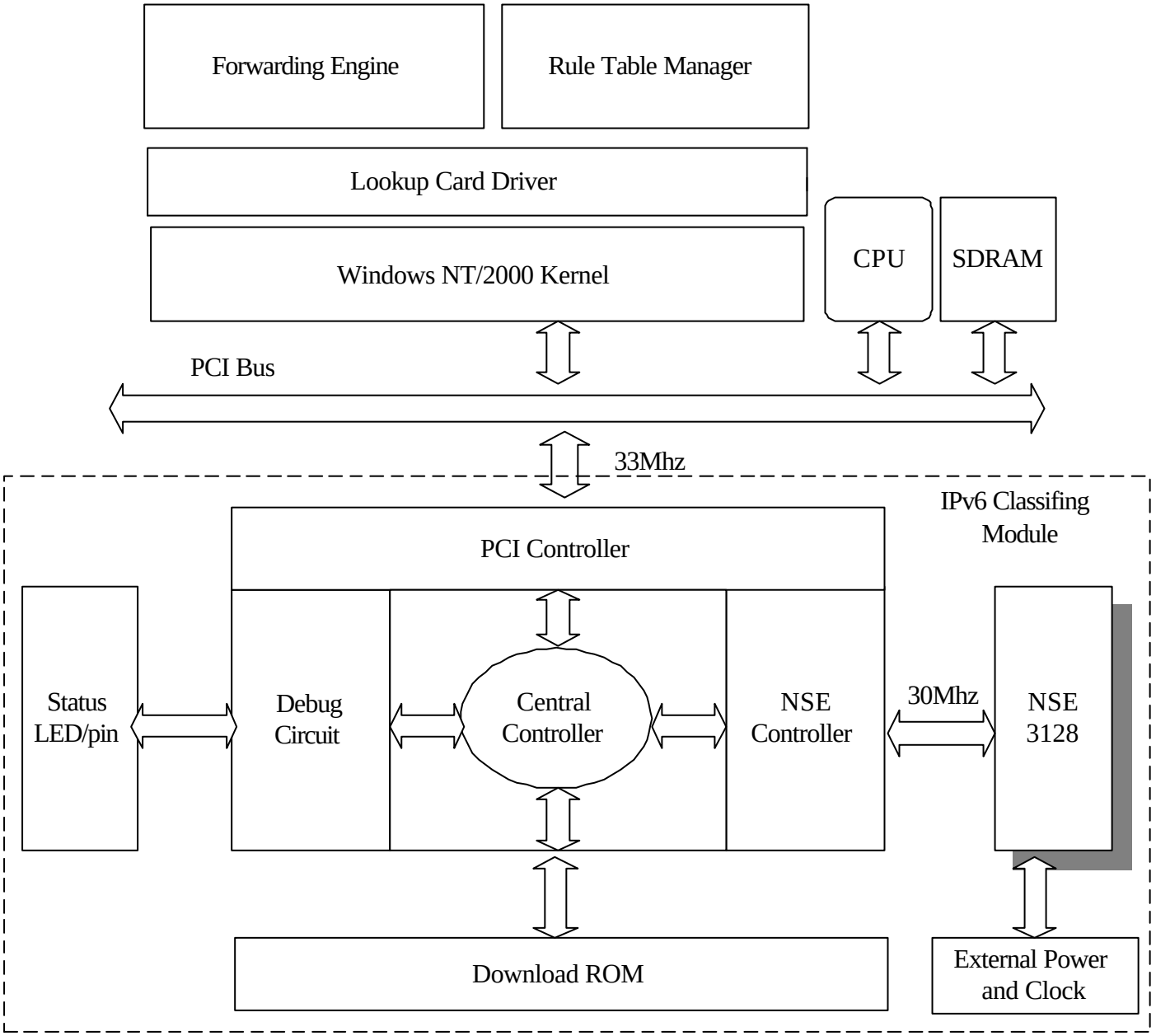

圖 4.1 整體系統架構圖

如圖 4.1 所示，本系統主要由下列幾個部分組成：

- 整體系統發展之平台
- 高速封包分類模組的中央控制器
- 高速封包分類模組的查詢用記憶體
- 高速封包分類模組的驅動程式(driver)
- 封包分類器之管理程式
- 封包分類器之其他應用程式

以下將分項簡述本系統之架構，並於第四章加以詳述。

4.1.1 整體系統發展平台

有鑑於 PC(Personal Computer)的普及、優良的價格效能比且兼具取得與移植方便的特性，本論文將整體系統發展於 PC 平台之上，也就是採用 Intel x86 指令集的中央處理器。在中央處理器之外，使用目前相當普及的記憶體 SDRAM(Synchronize Dynamic Random Access Memory)作為主記憶體。匯流排則採用目前個人電腦平台的主流 PCI 匯流排(Peripheral Component Interconnect bus)，作為 CPU 與下層的硬體，也就是高速封包分類模組溝通的橋樑。平台之上的作業系統則採用 Microsoft Windows NT/2000[32]為發展平台，以提供軟硬體間溝通的能力，並作為應用程式與驅動程式執行的環境。

4.1.2 高速封包分類模組

本論文自行研發高速封包分類模組，利用 PCI 匯流排作為介面與 CPU 溝通，並利用高速的 Content-Addressable Memory(CAM)作為查詢用的記憶體，以提升整體系統之效能。該模組上除了 CAM 以外，並使用 FPGA(Field-Programmable Gate Array)來作為中央控制器(Central controller)，FPGA 為可程式化之硬體的特性兼顧了硬體實作所能帶來的速度與效能，以及開發期間不斷修改，容易模擬驗證的需要。此模組尚有部分其他電路，例如除錯用的電路，電源與保護電路等，能更加保證整個模組的正常運作以及之後不斷研究發展的需求，

希望能兼顧效能與開發的需要。另外由於 CAM 以及 FPGA 兩部分的設計為本分類器的硬體核心，故將此二部分的設計內容於 3.2 以及 3.3 兩章節中詳述。

4.1.3 封包分類器軟體

在本論文中所使用到的軟體包括高速封包分類模組所需要的驅動程式，以及上層的管理程式，封包轉送引擎等。以上軟體均在 Windows NT/2000 平台上開發，而模組所需的驅動程式更需兼顧 PCI 匯流排的特性以讓系統完整運作。

4.2 封包流程與演算法

封包在經由網路介面卡(Network interface card, NIC)進入本論文所研發之系統以後，會經過以下的流程來進行處理，如圖 4.2 所示，並詳述如下：

1. 檢查封包的 destination MAC 位址是否是給予系統，若是則繼續處理，若否，則丟棄封包並結束。
2. 將封包的 layer3(destination IPv6 位址)及 layer4 (source/destination IPv6 位址及 source/destination port)資訊取出供下一步處理。

之後 layer3 的資訊會進行查表(lookup)，以取得繞路(routing)資訊，若能得到繞路資訊則將此資訊取出，否則就將繞路資訊設定為預設繞路(default route)。另外 layer4 的資訊也同時進行分類，並讀出該資料流的 Action，以決定要讓封包通過或者丟棄。在 layer3 的查表以及 layer4 的封包分類均獲得結果之後，則將兩者之結果依照表 4.2 的關係加以運算，結果如下：

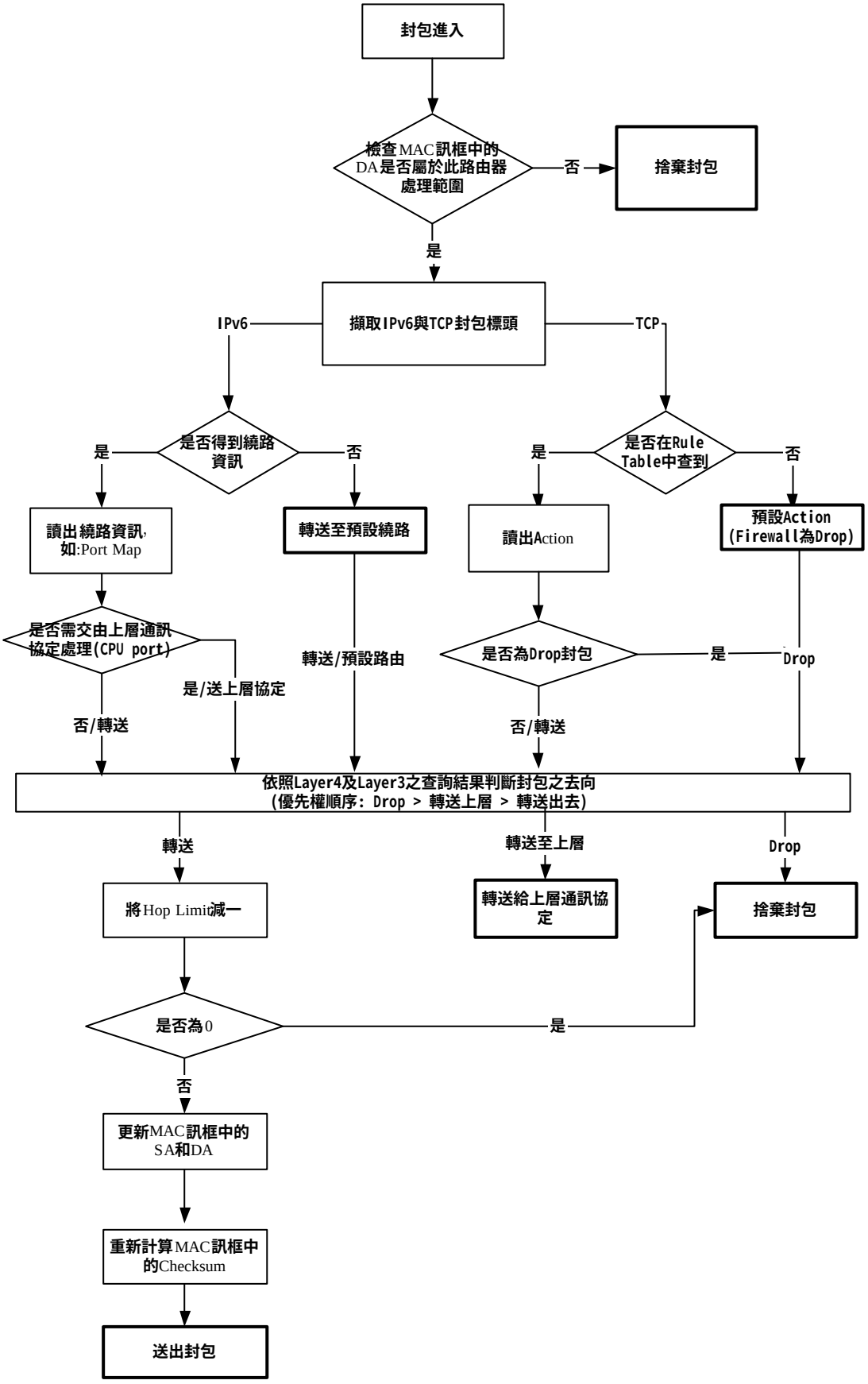
封包進入
檢查MAC訊框中的DA是否屬於此路由器處理範圍
否
捨棄封包
是
擷取IPv6與TCP封包標頭
IPv6
TCP
是否得到繞路資訊
是
否
是否在Rule Table中查到
是
否
讀出繞路資訊, 如:Port Map
轉送至預設繞路
讀出Action
預設Action (Firewall為Drop)
是否需交由上層通訊協定處理(CPU port)
轉送/預設路由
是否為Drop封包
是
Drop
是/送上層協定
否/轉送
否/轉送
依照Layer4及Layer3之查詢結果判斷封包之去向
(優先權順序: Drop > 轉送上層 > 轉送出去)
轉送
轉送至上層
Drop
將Hop Limit減一
轉送給上層通訊協定
捨棄封包
是否為0
是
否
更新MAC訊框中的SA和DA
重新計算MAC訊框中的Checksum
送出封包

圖 4.2 封包流程

Layer4分類結果(Action) Layer3 查表結果	Pass	Drop或無法查到
轉送至上層	送往CPU	丟棄
轉送至特定介面	轉送至特定介面	丟棄
無法查到	預設繞路	丟棄

表 4.2 封包查表運算結果

1. 是否丟棄封包，若否，則
2. 是否送往本機，也就是將封包內容送往層加以處理，若否，
3. 將封包往外送出，並決定由那個網路介面送出，並根據 RFC 中對 TCP/IPv6 以及 IEEE 802.3 標準對 Ethernet 的規定進行下面的手續：
    1. 將封包中的 hop limit 減一
    2. 若 hop limit 減一後為零則丟棄封包
    3. 更新封包中 MAC 位址的 source 位址
    4. 重新計算 MAC 層中的 checksum
    5. 將封包由指定的介面送出

封包在進行 layer3 以及 layer4 查詢的過程中均有可能發生查詢表中並沒有任何一個記錄能符合該筆封包的狀況。若 layer3 發生此種情形則會使用預設繞路將封包送出。layer4 分類時無法查到封包的內容則會與查到將封包丟棄時相同的把封包丟棄，這是因為本論文目前將此封包分類器實作為防火牆(Firewall)，若將本封包分類器實作為第四層交換器、QoS 路由器等其他應用，則表 4.2 的內容就需要加以改變，例如根據服務等級將表 4.2 的內容加以修改，欄位加入頻寬的等級，或以其他方式表示服務的合約內容(SLA, Service Level Agreement)等等，以符合實際的需求。

4.3 記憶體維護與管理

在設計出完整的 IPv6 封包分類器的過程當中，無可避免的需考慮封包分類的規則需如何設計及存放，以符合實際的需要，並兼顧實作的可能性。

4.3.1 記憶體資料結構

如第二章所述，每一筆規則中的每一個位元應該會有三種可能，即 0、1、或 X，也就是 Don’t care，但由於硬體實作的限制，必須使用 Mask 的方式來實作出 Don’t care，因此每一筆資料會切割成兩部分，由於長度與篇幅的關係，以 IPv6 128 位元位址為例：

例：3FFE:100::/32

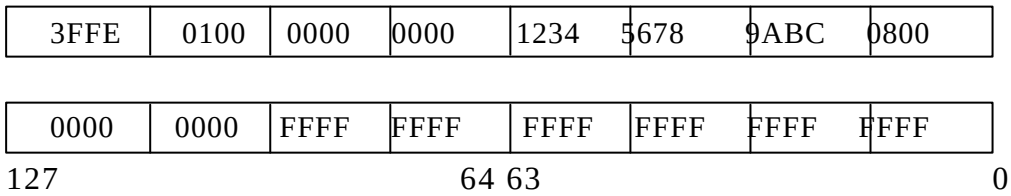

圖 4.3 三態 IPv6 128 位元位址記憶體儲存結構

上例代表 IPv6 的前 32 位元，也就是前兩欄的內容為 3FFE:0100，之後的位元均為 Don’t care 的狀態，因此上半部實際存放的資料除了前 32 位元會被拿來比較以外，後面的位元無論資料內容為何均不會被拿來比較。下半部即為 Mask，前 32 位元因為要加以比較，因此內容為 0，後面的位元因為不參與比較，所以內容為 1。如此應用於每一個欄位即可表示每一筆封包分類規則的內容，每一筆規則均用這兩種資訊來表示。

為了提升整體的效能，本論文使用 CAM 作為封包分類時存放封包標頭的記憶體。CAM 與一般的 RAM 主要的不同點在於 RAM 使用記憶體定址的方式來存取資料，而 CAM 採用內容當作 Index，也就是說輸入內容就能在一定時間內獲得其存放位址或整筆資料，非常適合用於查詢之用，但缺點是價格昂貴以及容量擴增不易。本論文使用 Netlogic 公司的 NSE3128 Ternary CAM[33]，除了擁有 CAM 查詢迅速

的特性之外，另外 Ternary 的特性表示可以實作封包分類時所需的第三態『X』(Don't Care)，並且能支援各種規則優先權排序方法。此種 CAM 寬度可調整為 288 位元，是目前可使用的產品中資料寬度最寬的一種。但根據第二章所述，IPv6 的第四層封包分類所需的每筆資料寬度至少需要：

128(SA) + 128(DA) + 16(SP) + 16(DP) + 8(Protocol) = 296 位元

目前仍不含每筆規則均需要的 Action 欄位便很明顯的超過了目前 CAM 所能提供的最大寬度，因此必須這些資料加以壓縮才能裝進 CAM 之內。本論文將 128 位元的 Source 位址，以及 Destination 位址加以壓縮，如圖 4.4：

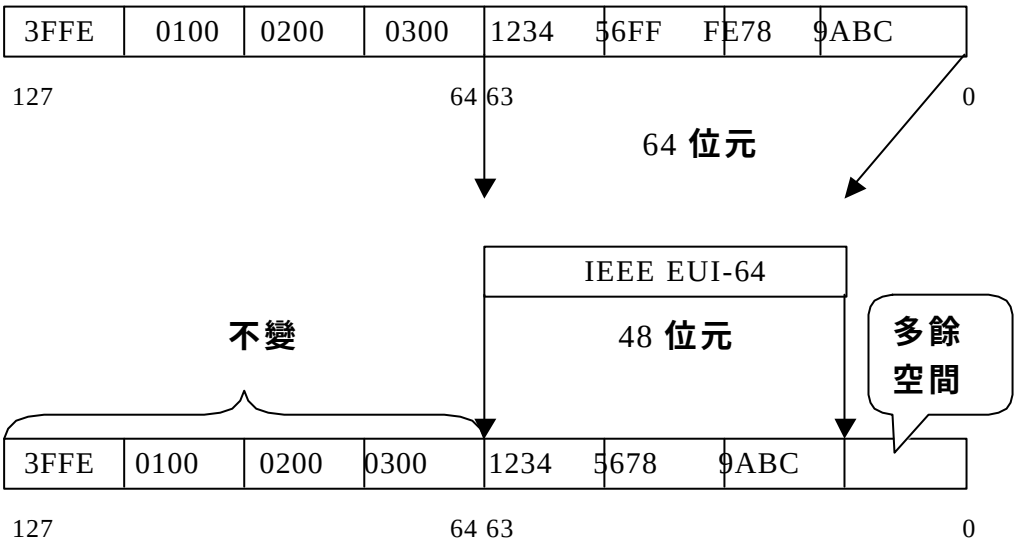

圖 4.4 壓縮 IPv6 128 位元位址

如此便將原本 128 位元寬的 IPv6 位址壓縮為 112 位元，節省了 16 位元，因此每筆資料所需的寬度也成為：

112(SA) + 112(DA) + 16(SP) + 16(DP) + 8(Protocol) = 264bits

不但可以裝入 288 位元寬的 CAM 當中，並剩下 288-262=26 位元的資料寬度，作為存放 Action 以及未來保留其他功能之用。本論文將每筆資訊的前兩個位元保留作為日後之用，並定義了 4 位元的 Action 欄位，在 CAM 當中每筆內容的資料結構定義如表 4.3。

Type (2)	SA (112)	DA (112)	SP (16)	DP (16)	Protocol (8)	Action (4)

Type: Entry type, reserved for future use
SA: Source address, original 128bits and compressed to 112bits in ethernet
DA: Destination address, original 128bits and compressed to 112bits in ethernet
SP: Source port, 16bits
DP: Destination port, 16bits
Protocal: Protocal, 8bits
Action: Action to made, now defined Pass/Drop/CPU

Now use total 2+112+112+16+16+8+4=270bits
There are 288-270=18bits reserved for future use, like QoS level or flow label

表 4.3 CAM 每筆資訊之資料結構

4.3.2 規則表管理

另外一個儲存封包分類規則時所會遭遇的問題是規則表的管理問題。規則表需要為完成所需的動作，可以直接對記憶體晶片做動作，但此方法對上層而言並不方便，也不能夠模組化以重複利用。因此可以用提供函式的方式讓外部對其存取與控制。若以函式的方式提供外部的介面，至少需提供：

- 初始化(initialize)
- 清除規則
- 填入規則
- 查詢/分類

這四個函式才能夠進行運作。這些函式都不需要定址，只需要將所需的資料視為傳入值即可。但這些函式只能提供基本的運作所需，在新加入部分規則時需要將規則全部清除後再更新，因此可將填入規則改為：

- 插入規則
- 更新規則

如此可應付規則重新輸入以及部分更新或加入的需要。同時將清

除規則視為初始化的一部份。但這樣的函式表並沒有提供部分規則刪除的可能，因此更完整的函式表應加入部分刪除的功能，但此功能需要對記憶體進行定址，否則無法辨認欲刪除之規則。

- 刪除規則

在加入刪除規則此函式之後，整個函式庫也漸趨完備，也能更快速的處理各種規則更新的要求。若要提供進階的資料結構更好的支援，則應該維護一空餘位址的列表，但由於在硬體上進行此列表維護的難度相當高，所以可以在上層的軟體上進行維護。由於本論文將此封包分類器實作為防火牆，封包分類規則更新的頻率不高，故本論文並沒有實作出刪除規則的函式，並將於 4.4.2 章節中詳述。

4.3.3 規則優先權管理

另外，規則表內儲存了封包分類的規則，因此規則間的優先權，甚至在多筆規則間有部分範圍重複時，要如何分辨優先權便是需要面對的課題。目前主要的解決方法有：

- 避免規則間範圍重複的發生。適合於 Action 簡單的場合，如防火牆若不讓封包通過即丟棄，故規則間必不可重複。
- 利用 Prefix 的長度來辨別優先權，也就是 Longest Prefix Match。
- 利用存放及搜尋的方式來辨別優先權，例如規定高優先權的規則放在高記憶體位址，並使用 Linear Search 的方式搜尋以保證高優先權的規則會被先比對到。

三種方法之間各有優缺點以及各自適合的應用範圍，本論文為了讓此分類器能廣泛的應用於各種領域，如防火牆、第四層交換器、QoS 路由器等，因此採用第三種方法。也就是說，假設將 N 筆封包分類規則加以存放，並優先存放高優先權規則的方式，則儲存結果如圖 4.5 所示。但若規則間不會重複，則可不分辨規則間的優先權，如本論文實作的防火牆即為一例。在此狀況下，分類規則存放時不必使用優先權的觀念，因此並無排序之需要。

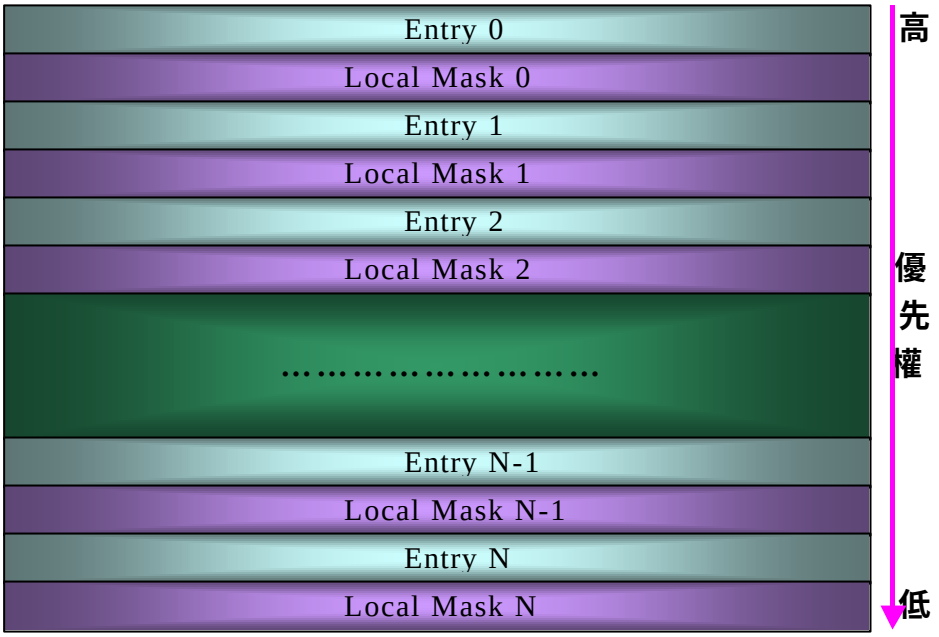

圖 4.5 IPv6 封包分類規則表優先權

4.4 中央控制器之結構

中央控制器的整體結構除了程式下載電路以及除錯用電路之外，其他的功能為：

- 透過 PCI 匯流排與 CPU/RAM 溝通
- 透過自訂的 bus 與 NSE3128 CAM 溝通
- 控制器內部的同步與流程控制

前兩個部分也是中央控制器最重要的兩個部分，如圖 4.6 所示，將於下文詳述。本論文使用 Altera[34]公司的 FPGA 以及開發環境作為開發中央控制器之用，為兼顧邏輯大小、I/O 接腳數目以及運作速度等需求，選用 EPF10K200GC503-3[35]此 FPGA。本論文利用 FPGA 所提供的特性將中央控制器切割成兩部分，可各自使用不同的頻率，一部份與 PCI 匯流排使用相同的頻率，一部份與 CAM 使用相同的頻率，兩者透過非同步的信號傳送命令與回應，並利用 FPGA 內嵌的 Embedded RAM 來傳遞所需的資料，如此可顧及設備間的不同特性並增加擴充的能力。

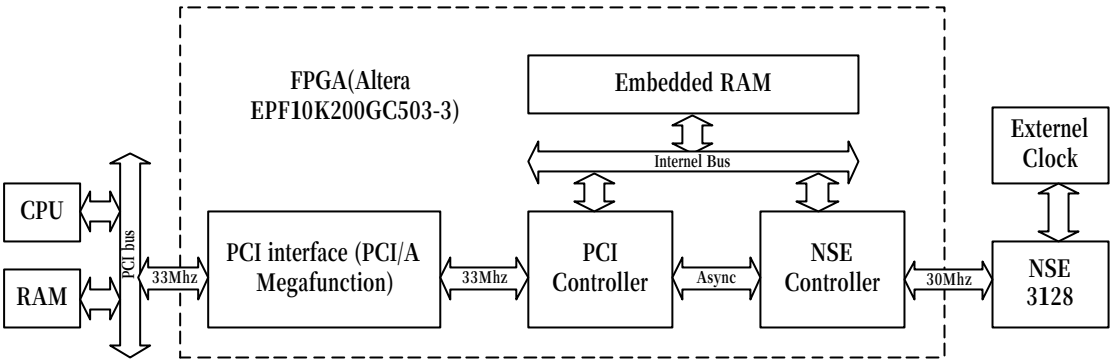

圖 4.6 FPGA **控制器整體結構**

4.4.1 PCI 匯流排控制

PCI **控制器需透過** PCI **匯流排收取上層驅動程式的命令，並將執行結果傳回驅動程式，故需實作** PCI **匯流排所需的信號及資料控制。但因** PCI **匯流排發展已經超過五年，相關的標準等早已齊備，故本論文使用** Altera **公司的** PCI/A megacore function[36]**此** Module **來實作** PCI **匯流排所需要的信號控制，並以其他程式與** PCI/A Module **配合。**

4.4.2 CAM 控制

CAM **控制器需與** CAM **進行溝通，並需提供一應用程式介面** (Application Interface, API)**供外部呼叫之用，並提供信號控制之介面。故整個** CAM control block **的** Input/Output **可分為兩個部分：**

- **利用** FPGA **的** I/O pin **控制** CAM **的信號**
- **提供外部呼叫及信號控制的** API

第一部份主要規定於 CAM **本身的** Datasheet **當中，並將於第四章中詳述。第二部分的接腳介面定義如下：**

- Instruction Bus(4bits)
- Result(4bits)
- CMD(1bit)
- OK(1bit)
- Reset(1bit)
- Data/Address Bus for Embedded RAM

Code	Instruction	Operand in embedded RAM
0	Nop	None(不需要改變 CMD)
1	Init/Reset	None
2	Insert	存入筆數/資料內容
3	Update	存入筆數/資料內容
4	Lookup	查詢內容

表 4.4 CAM control block 提供之命令

上層的電路可藉由改變 CMD 的狀態，通知 CAM control block 收取命令，並視命令的內容決定是否到 Embedded RAM 中收取所需資料。再將此命令傳遞至 CAM 執行完畢後，將結果在 Result 中回傳，並改變 OK 的狀態以通知上層命令已經執行完畢。以上定義供上層使用的指令如表 4.4 所示。由於上層程式在下令進行封包分類後真正需要的結果是 Action 此欄位，而不需要回傳整筆規則的內容，因此將 Action 欄位於 Result 接腳中回傳，而不使用 Embedded RAM 回傳以增加執行速度。

4.4.3 中央控制器的內部控制與整體流程

在將 PCI 控制器、CAM 控制器，以及其他電路組合起來之後，即為完整之中央控制器。中央控制器除了空轉(Nop)狀態以及重置(Reset)狀態並無特別的控制流程與額外的說明需要之外，其他的狀態流程列於圖 4.7 中。

由圖 4.7 當中可以得知，中央控制器主要的控制程序可以整理如下：

- PCI/A 模組收到上層驅動程式的觸發(trigger)
- PCI/A 模組收取上層驅動程式傳來之命令及資料
- PCI 控制器將上層傳來之資料送往 Embedded RAM 中
- PCI 控制器設定旗標並下命令給 CAM 控制器
- CAM 控制器收取命令

- CAM 控制器視情形收取(或不收取)Embedded RAM 中之資料
- CAM 控制器控制 CAM 的時序與命令並等待結果
- CAM 控制器收取 CAM 傳回之結果並送回 PCI 控制器
- PCI 控制器視結果將結果送往 PCI/A 模組
- PCI/A 模組提出要求並送回執行結果

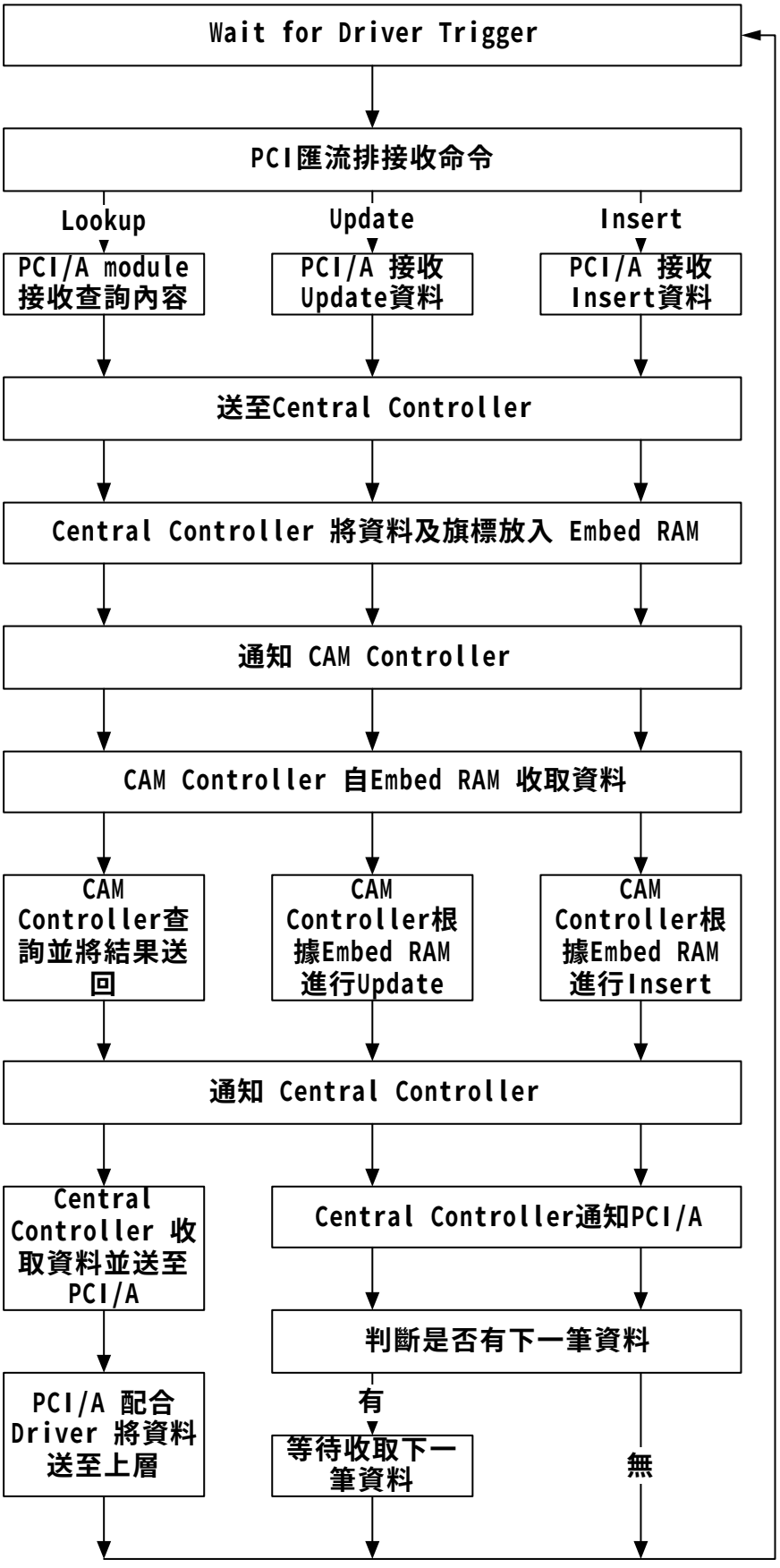

圖 4.7 中央控制器之控制流程

第五章
IPv6 封包分類器之系統實作

本論文將實作一完整的 IPv6 Packet Classifier，整體系統運作方式如圖 5.1 所示。

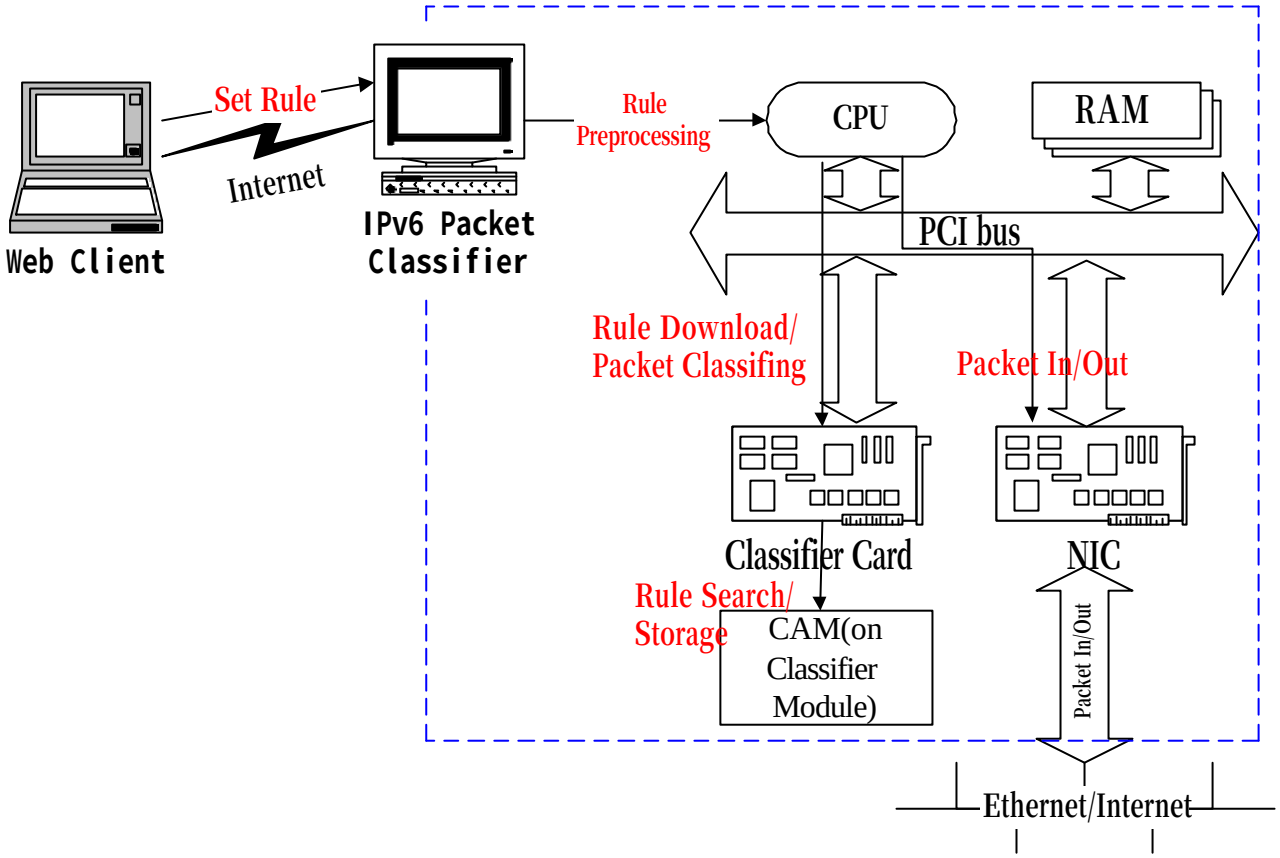

圖 5.1 IPv6 Packet Classifier 整體系統運作

如圖 5.1，使用者藉由 WWW[37]設定 Classifier 的分類規則及其 Action，藉由網際網路傳遞給此 Packet Classifier。CPU 會先對這些規則進行前處理運算，轉換其格式以適合硬體儲存與查詢。CPU 藉由 PCI 匯流排，一方面控制網路介面卡的封包輸出與輸入，另一方面控制封包分類模組的運作，包括封包分類規則的下載、及對每個封包做查詢等。封包分類模組本身在收到要求之後，控制模組上的 CAM 以進行實際的搜尋與儲存。若以軟硬體架構的方式概觀整體系統，則將如圖 5.2 所示。

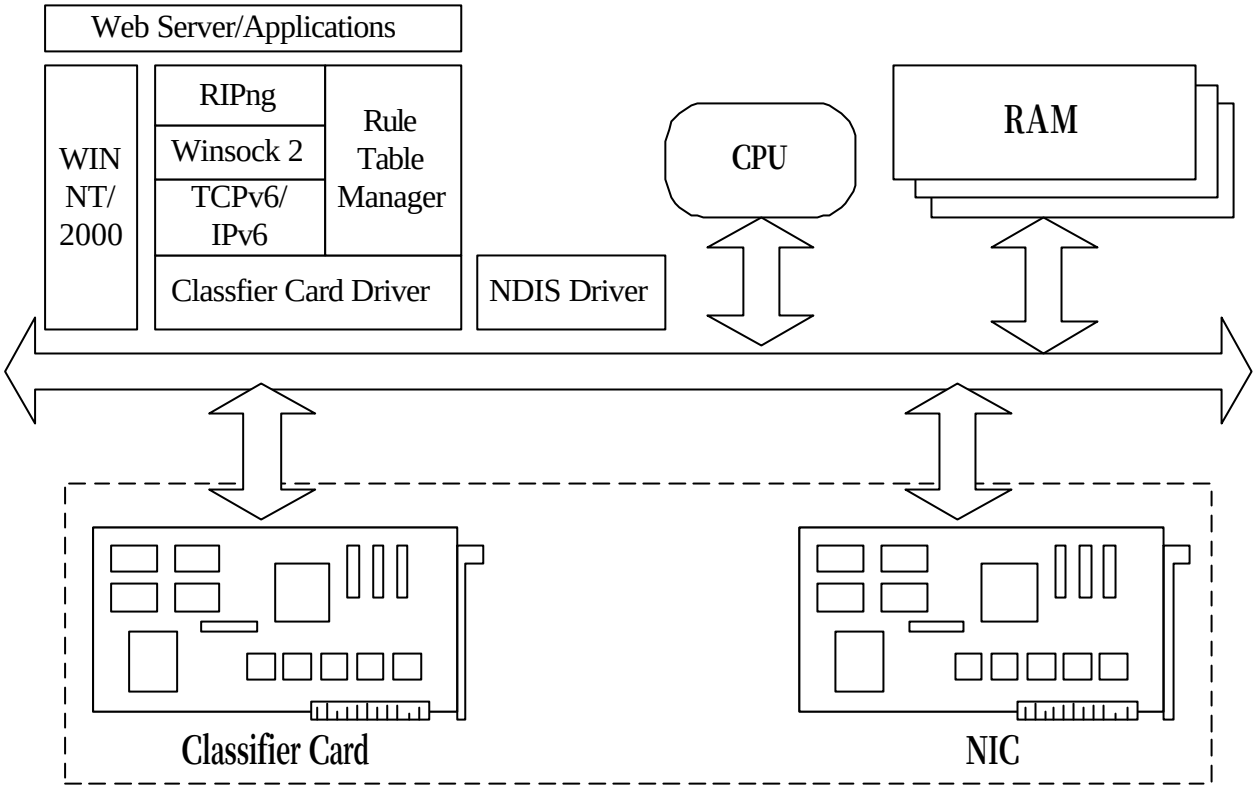

圖 5.2 IPv6 Packet Classifier **軟硬體運作**

圖 5.3 **是一般封包所會經過的處理步驟：**

1. **在電源啟動之後，上層的軟體將進行初始化，並由** Rule Table Manager **將** Rule Table **下載給下層的封包分類卡。**
2. **封包進入，由網路介面卡收取。**
3. **網路介面卡驅動程式將封包標頭取出，並交由封包分類卡進行封包分類。**
4. **封包分類卡將分類結果回報給上層軟體，上層軟體決定封包輸出之網路介面後交給網路介面卡驅動程式。**
5. **網路介面卡驅動程式將封包交給網路介面卡，並將封包送出。**

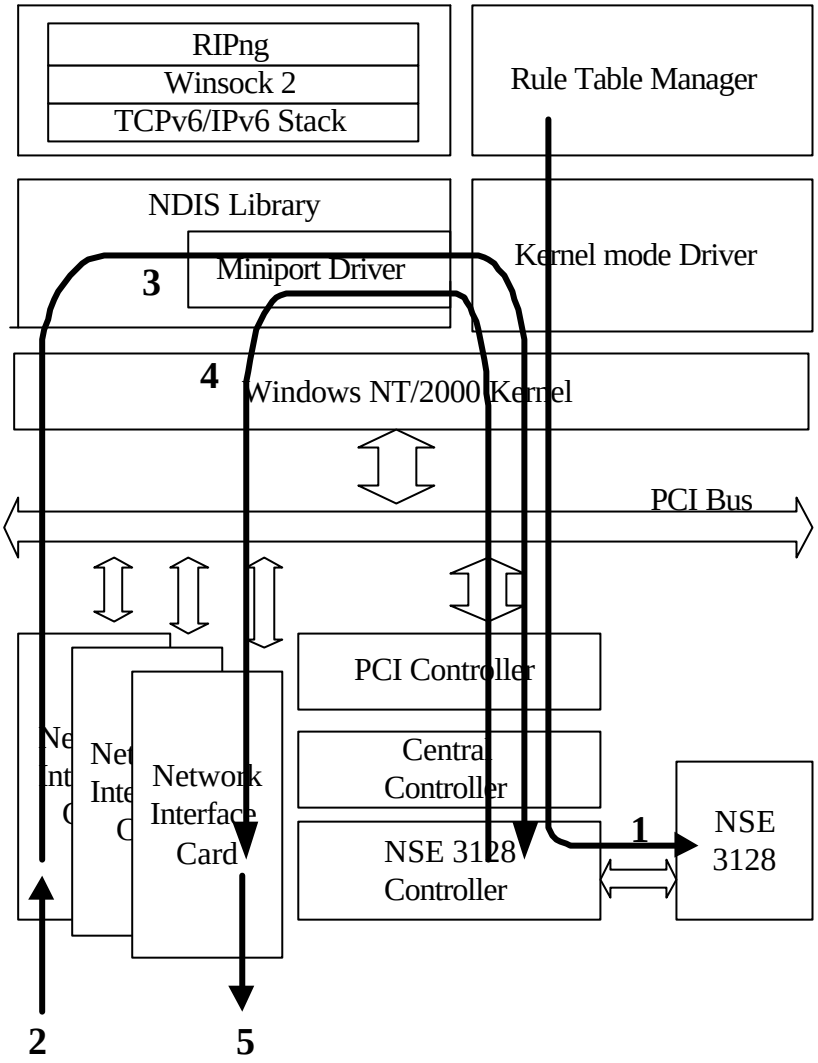

圖 5.3 一般封包處理流程

若封包交由上層軟體處理，甚或遭到丟棄，也就是封包自網路介面進入以後不再送出，則會經過下列步驟，如圖 5.4。與圖 5.3 相比較，前三個步驟均相同，但第四個步驟在封包分類卡將分類結果回報給上層軟體之後，由軟體決定封包去留及走向，最後將封包丟棄或將封包送往上層應用程式，如 Routing Protocol 等。因此圖上的第五個步驟劃上虛線，表示若封包遭到丟棄則無此步驟。

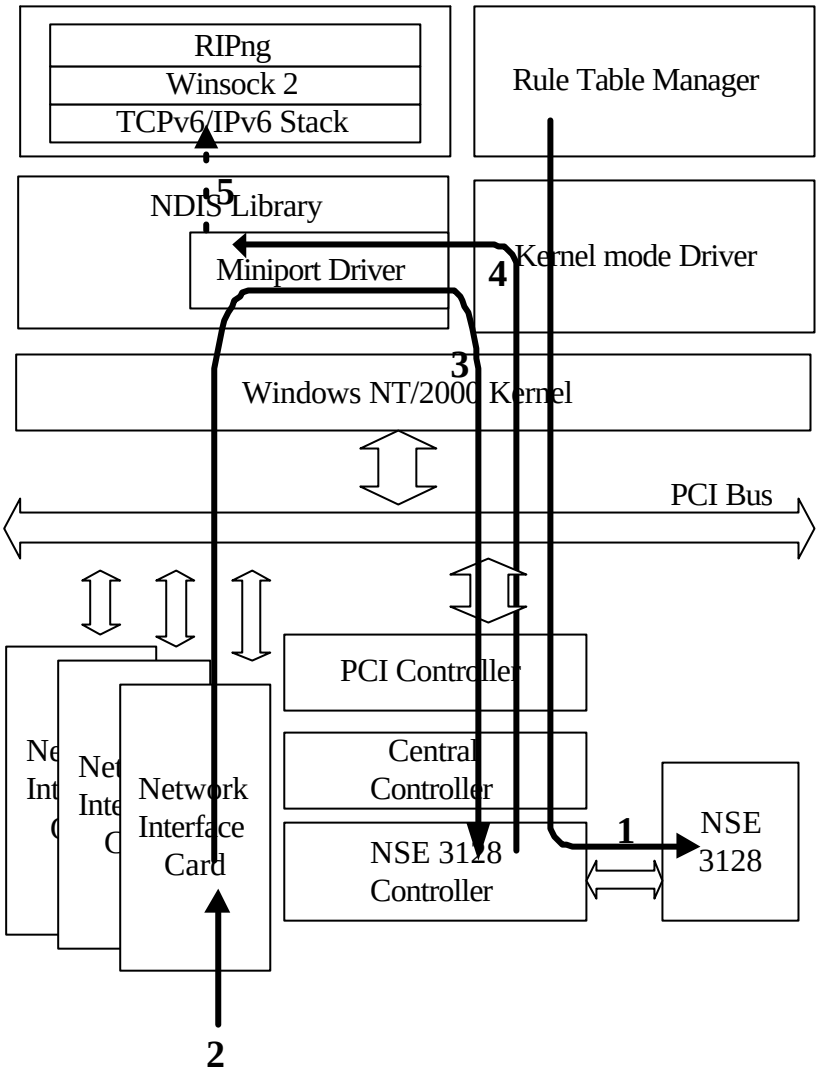

圖 5.4 特殊封包處理流程

綜合以上的介紹，可整理出整體系統可由下層硬體到上層軟體分為數個部分，分別為：

- **封包分類模組硬體電路設計**
- CAM **控制電路設計**
- FPGA **開發設計**
- PCI **匯流排介面/驅動程式開發設計**
- Software **開發設計**

本章之以下各節將分項詳述之。

5.1 封包分類模組硬體電路設計

本論文實作之封包分類模組為一延展性佳，可兼具開發驗證與實際應用功能的硬體平台，除提供 PCI 介面、FPGA 以及 CAM 整合發揮功能之平台以外，其他的功能如：

- 各元件間獨立電源以提供代換性，並能在少數元件損壞時繼續使用。
- FPGA 程式下載電路可使用內建下載電路以節省體積，方便於工業用 PC 使用，或外接以方便開發過程使用。
- 提供內建的直流電源產生器，並使用可變電阻以提供多樣化的電源供應。
- 將 FPGA 以及 CAM 的接腳外接，讓完成之查表卡可作為 Demo Board 使用，甚至作為 FPGA 之開發環境。
- 利用 IC 腳座(socket)使電路版上的晶片可重複使用。
- 除錯與外接用針腳提供排線之介面與電源供應。
- 內建相當數量之發光二極體(LED)以及跳線用接腳，並以正負兩種邏輯設計除錯用電路，提供方便之除錯環境。

圖 5.5 為本模組之功能區塊圖：

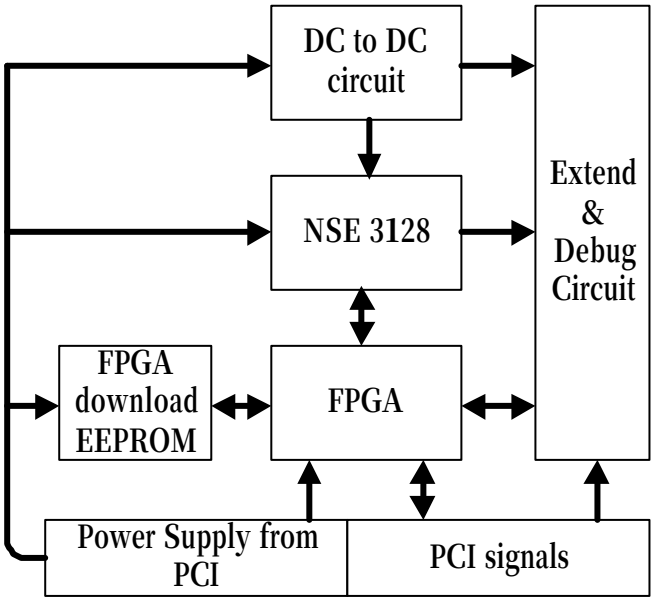

圖 5.5 封包分類模組之功能區塊圖

針對此模組的各項功能設計區塊圖之後，便以 Orcad[38]電路繪製軟體製作電路圖，經過檢查以後與廠商下訂單製作電路版，並準備電路版上所需的其他材料(如電阻等)供焊接時使用。圖 5.6 為 Orcad 執行繪製電路圖之畫面，該電路圖為直流電源轉換電壓的電路。

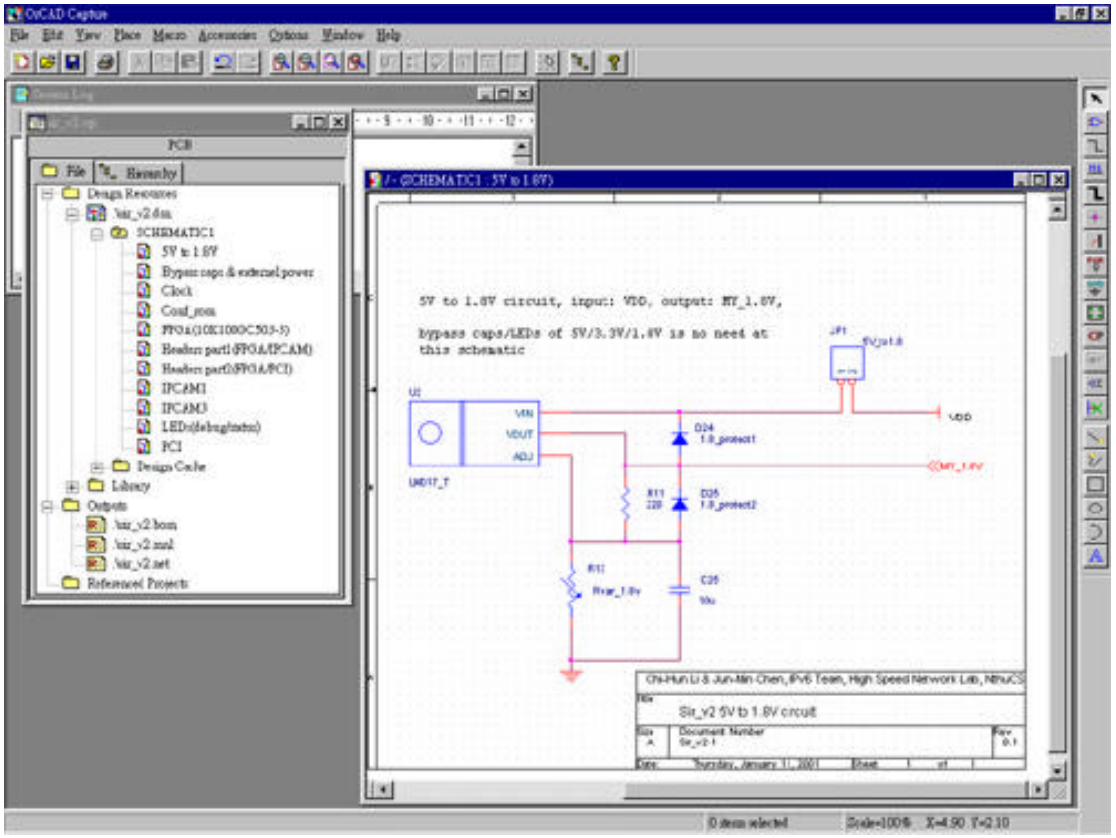

圖 5.6 Orcad 電路繪製畫面

在廠商將完成之電路版送回後，即為規劃測試與除錯之程序。圖 5.7 為本論文所完成之電路版之正反面，上方正面的電路版圖當中，右上方為 CAM 的插槽，中間已經插有 FPGA，左上方為 FPGA 之下載程式電路，右下角的電路可提供 PCI 未提供之直流電源，左方的發光二極體可顯示狀態或提供除錯之用，其他多為輸出入的外接針腳，提供其他功能的使用。下方電路版的背面圖由於尚未插件故仍無針腳之痕跡。本模組的功能已經完備，但為了擴充功能並讓電路版的面積縮小，故仍須製作下一個版本的封包分類模組。

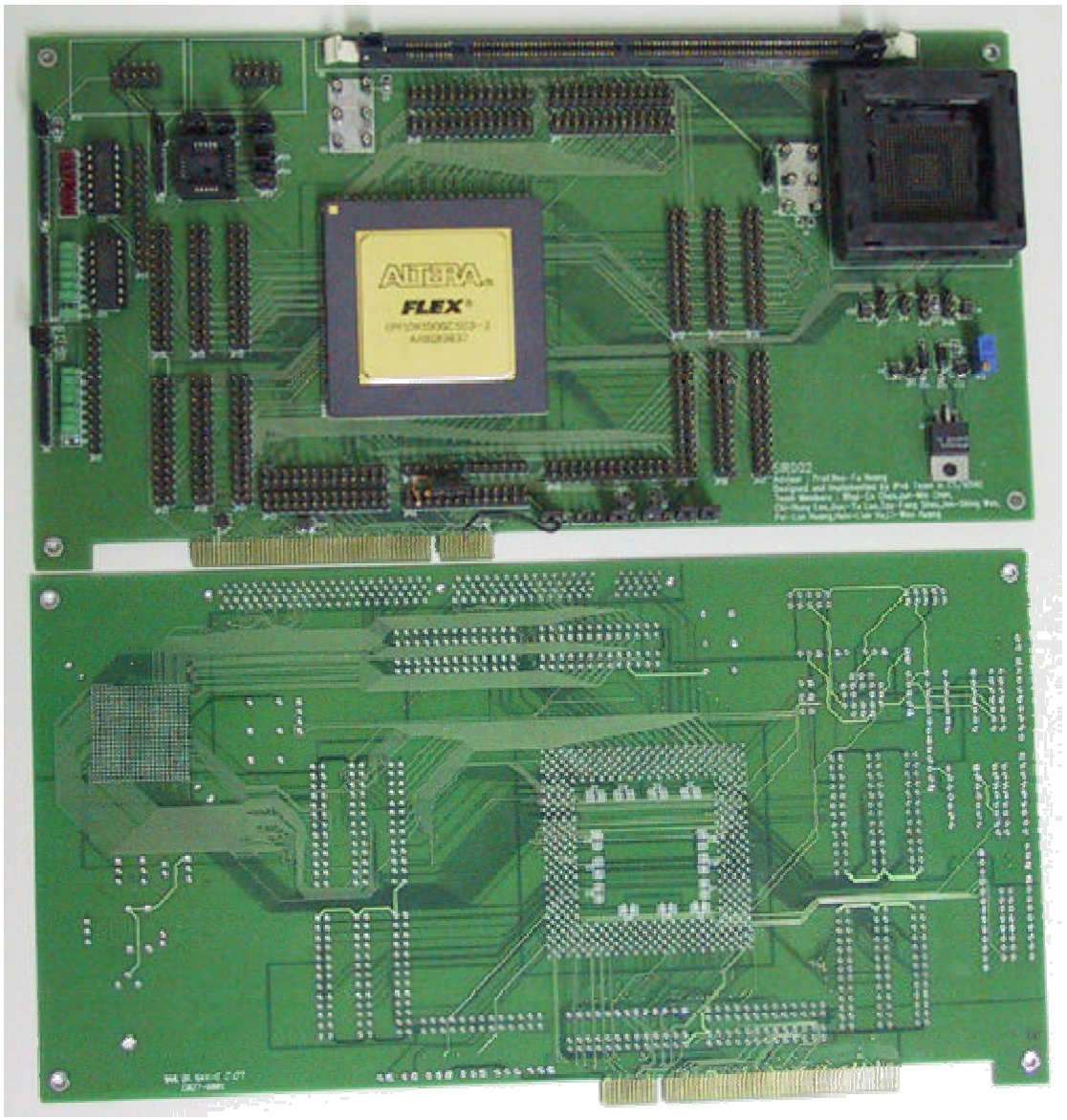

圖 5.7 完成之封包分類模組硬體正反面電路

5.2 CAM 控制電路設計

本論文所使用的 CAM，為 Netlogic 公司編號 NSE3128 之產品，其區塊架構圖如圖 5.8 所示。在設計控制電路時，主要控制對象就是傳遞指令的 IBUS 以及傳送資料的 CBUS，其他則為狀態控制接腳。藉由控制這兩個匯流排，可控制內部的暫存器內容以及真正儲存資料的記憶體。在將 CAM 加以初始化以後，即可將資料存入放置資料的記憶體中，之後將欲查詢的資料寫入暫存器中，結果置於其他暫存器

以供讀出，另外此 CAM 也提供了其他的指令用以實作其他的功能。由於本論文將此硬體平台實作為防火牆，故在此實作之中以最適合防火牆的方式儲存 CAM 的資料結構，以提升執行速度。

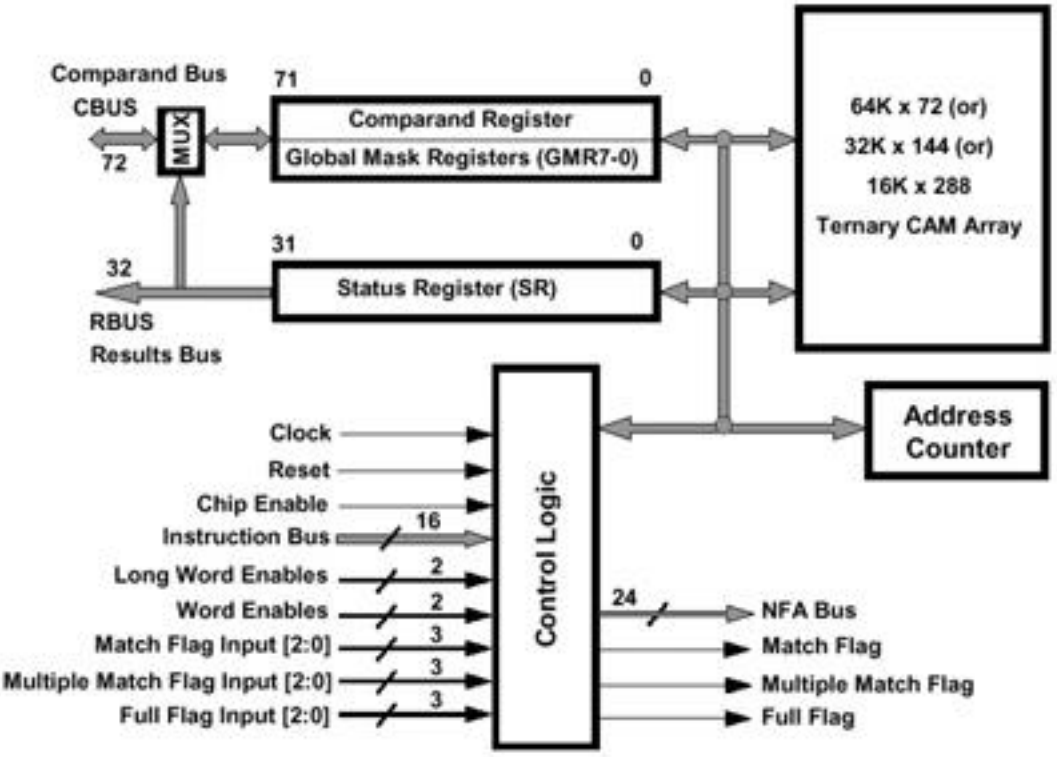

圖 5.8 NSE3128 之區塊架構圖

5.3 FPGA 開發設計

本論文開發設計 FPGA 所使用的電腦輔助電路設計工具(CAD tool)是 Altera 的 Maxplus2[39]，圖 5.9 即為整體 FPGA 程式之設計圖。Maxplus2 此環境可使用的輸入介面相當多，本實作使用 Verilog Hardware Description Language(Verilog HDL)[40]來設計每個程式區塊的內部，除了自行設計的程式區塊以外，尚使用了一些設計工具所提供的元件，最後使用圖形介面的方式將這些區塊連接起來。整體的設計流程可以簡單分類為輸入程式、編譯、功能模擬、電路合成、硬體波形模擬，最後燒錄至實際硬體以驗證正確性。程式輸入與模擬佔據了設計過程的主要部分，程式的寫作關係著實際電路的功能正確性與執行速度，模擬則是在實際燒錄測試外，先行驗證程式的正確性。

硬體波形模擬由於使用了 FPGA 硬體本身的特性，因此能準確預測其行為模式(Behavior Model)的結果，包括程式正確性以及時序分析(Timing Analysis)。也就是說能在燒錄硬體之前實際得知燒錄後的行為及執行速度的極限等。利用工具的輔助方得以順利設計並驗證 FPGA 電路設計的正確性。

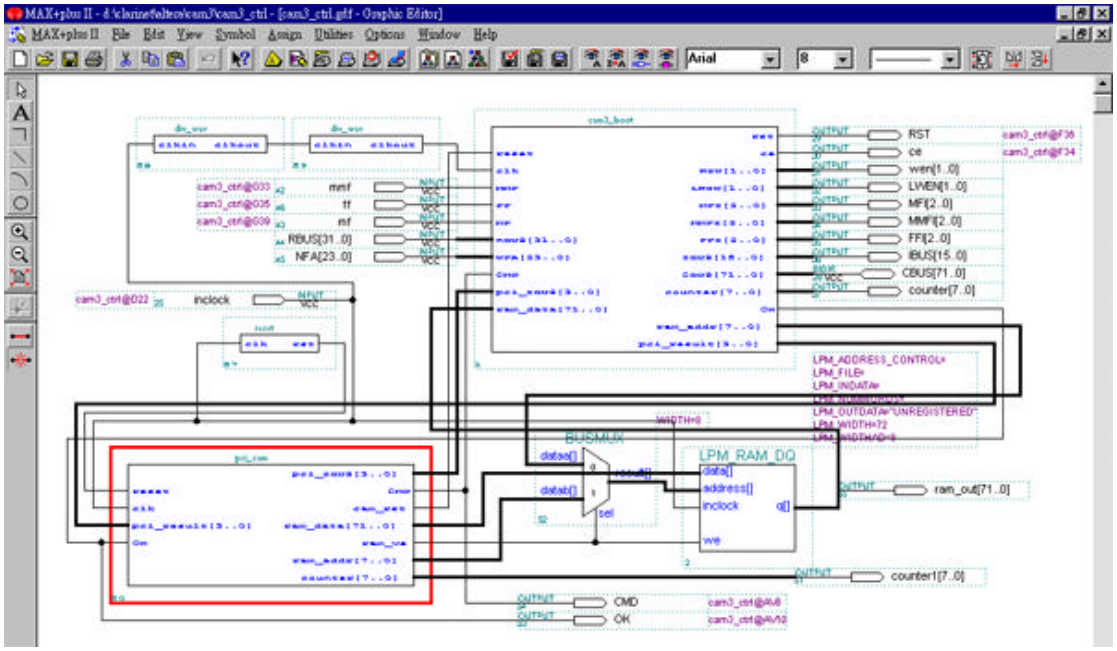

圖 5.9 Maxplus2 電路設計畫面

確定設計的電路功能正確之後，為了提升執行效率，接下來需要進行最佳化的工作。電路設計工具內的時序模擬可供找出延遲時間最長的路徑(Critical Path)與輸出入的延遲等等模擬結果，並針對這些延遲的因素修改程式的寫法、調整電路合成參數或利用 FloorPlan Editor 修改邏輯配置的位置等等漫長的努力方能結束電路的調校工作。之後便可將完整的硬體電路燒錄至可程式化唯讀記憶體(Programmable ROM, PROM)當中，進行實際硬體的測試。

5.4 PCI 匯流排介面/驅動程式開發設計

PCI 的介面可分成兩個部分：PCI/A 模組與控制器、封包分類卡之 PCI 驅動程式。PCI/A 為 Altera 公司的一個 Megacore function，負

責處理 PCI 各個接腳的訊號，其功能如下：

- PCI master feature
  - Memory read/write
  - Bus parking
  - 64 bytes (16 double words) RAM buffer
  - Zero-wait-state PCI read/write burst transactions
  - DMA engine (address counter register、byte counter register、control & status register、interrupt register)
  - Configurable interrupt source (DMA terminal count、master abort、target abort、local side interrupt)

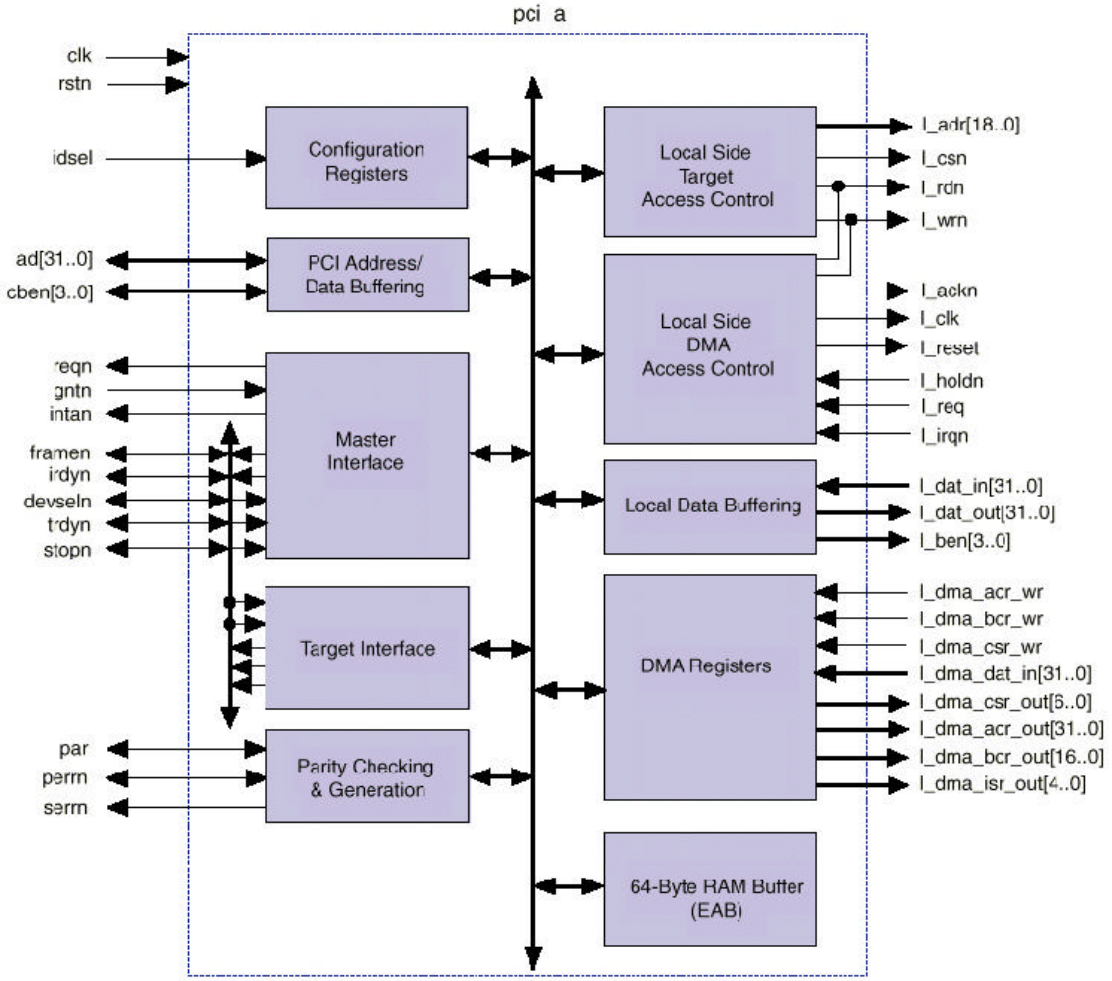

圖 5.10 PCI/A 架構

- PCI target feature
  - Type zero configuration space

- Parity error detection
- Memory read/write and configuration read/write
- Target retry and disconnect
- 1 Mbytes 2Gbytes of parameterized target memory space

■ **設定暫存器**(Configuration Registers)

- parameterized：Device ID、Vendor ID、Class code、Revision ID、Base address zero、Subsystem vendor ID
- non-parameterized：Command、Status、Header type、Latency timer、Interrupt pin、Interrupt line

PCI Bus Configuration Registers				
Address	**Byte**			
	3	**2**	**1**	**0**
00H	Device ID		Vendor ID	
04H	Status Register		Command Register	
08H	Class Code			Revision ID
0CH	BIST	Header Type	Latency Timer	Cache Line Size
10H	Base Address Register 0			
14H	Base Address Register 1			
18H	Base Address Register 2			
1CH	Base Address Register 3			
20H	Base Address Register 4			
24H	Base Address Register 5			
28H	Card Bus CIS Pointer			
2CH	Subsystem ID		Subsystem Vendor ID	
30H	Expansion ROM Base Address Register			
34H	Reserved			
38H	Reserved			
3CH	Maximum Latency	Minimum Grant	Interrupt Pin	Interrupt Line

圖 5.11 PCI **設定暫存器**

圖 5.10 **為** PCI/A **模組架構，圖的左邊為** PCI **的接腳，右邊為內部模組的接腳，提供其他模組控制** PCI/A。PCI/A **模組中有兩個比較重要的部份：設定暫存器和** DMA。**圖** 5.11 **為** PCI **規格所定義的設定暫**

存器示意圖，顏色較深的欄位為 PCI/A 有支援的部份。這些暫存器用來提供上層驅動程式、作業系統或其它軟硬體所需的資訊，如：Device ID、Vendor ID,…,等。至於這些暫存器的詳細內容及功用在此不再贅述，PCI 規格中已經有明確的定義。

在驅動程式方面，為節省 CPU 的負載，大量資料的傳輸都利用 DMA。原本 PCI 規格中，每一次的讀寫最多可包含 1 16 筆資料，每一筆資料長度為 32 個位元。但由於 PCI/A 的限制，只有利用 DMA 才能傳送大量的資料，其它方式每次只有一筆。所以，在 IPv6 交換模組中，大量的資料，如：路徑表更新等利用 DMA 來傳送，其它則單獨傳送，如：讀寫 DMA 及其它相關的暫存器。

DMA 與驅動程式傳送資料採取「記憶體對映」(Memory-mapped)的方式，所以必須有共同的記憶體配置方式，雙方才能順利的溝通。在 PCI/A 裏，所能定址的記憶體範圍為 1MB 2GB。PCI/A 將記憶體空間分為兩部份，能夠自行規劃的只有一半。前半段存放 DMA 暫存器，提供給驅動程式及內部模組控制，也就是只有後半段能供利用。並且不論管理的記憶體大小，DMA 暫存器區塊都佔去一半空間。表 5.1 為 DMA 暫存器的配置方式：

位址範圍	位元使用率 (使用/全部)	可否讀寫	預設值	名稱
00000H 00003H	8/32	讀/寫	0	DMA 控制及狀態暫存器
00004H 00007H	32/32	讀/寫	0	DMA 位址計數暫存器
00008H 0000BH	17/32	讀/寫	0	DMA 位元組計數暫存器
0000CH 0000FH	5/32	讀/寫	0	DMA 中斷狀態暫存器

表 5.1 DMA 暫存器的配置方式

以上這四個暫存器控制著 DMA 能否正常運作，雖然佔用了一半的可定址記憶體區塊，但實際上只能存取這四個暫存器。驅動程式則是透過控制這些暫存器來控制模組的運作，另一方面在 FPGA 上也是利用這些暫存器來傳遞系統運作所需要的各種資訊給上層的驅動程式以及各種應用程式。除了 DMA 的模式以外，也可以視時機使用一般的傳輸模式來傳送小量的資料，可減少驅動程式的複雜性。

5.5 上層軟體開發設計

本論文之實作除了下層的硬體平台之外，尚有軟體的配合方能使整體系統運作。以下介紹各種上層軟體：

- Web-based 的設定介面，與使用者最直接相關，使用者藉由瀏覽器(browser)以及網際網路連線以設定封包分類之規則，而伺服器端透過網頁接收使用者的設定。因此伺服器端需要提供 WWW 的伺服器，如 Apache[41]等，以及以 HTML[37]撰寫的網頁提供使用者設定的介面。
- 伺服器接收使用者的設定以後需要對這些規則作初步的整理，例如將重複的規則挑出，或者利用某些演算法將這些規則加以最佳化，並轉換為下層硬體的資料格式以利下載。
- 轉換完成的封包分類規則將透過 Kernel-mode[42]驅動程式下載給封包分類模組，其他所有與封包分類模組的溝通也是利用此驅動程式達成。
- 封包由網路介面進入後，由網路卡的驅動程式收送之。網路卡的驅動程式需以 NDIS[42]介面撰寫。
- RIPng，可與其他路由器交換並更新繞路資訊。
- 為有效率的送出封包，可藉由一獨立之 Forwarder(封包交換程式)來控制封包的輸出，並以 GUI(Graphic user interface)的方式提供使用者設定的介面。

以上的上層軟體除了網頁需以 HTML 撰寫以外，其他均以 C 語

言作為開發之語言。HTML 語言或網頁可利用許多的開發工具，如 Netscape Composer[43]等以減少開發所需的時間。驅動程式因為需要與作業系統配合，故使用 Visual C++[44]此整合開發環境(IDE, Integrate Desktop Environment)，其他應用程式層級的軟體多半使用 Borland C++ Builder[45]。藉由以上的上層軟體與硬體的配合，方能發揮硬體的最大功能，使整體系統能順利運作。

第六章
IPv6 封包分類器之系統測試

圖 6.1 為 IPv6 封包分類器之完整系統圖，整體系統的測試工作可分為以下幾部分：

- 上層軟體之整體測試，包括設定封包分類規則、封包分類規則處理、繞路資訊交換協定、封包收送與網路介面卡驅動程式之功能整體測試。
- 封包分類模組與其驅動程式之連接測試，包括功能與執行速度測試。
- 封包分類模組與 CAM 之控制測試，包括功能與執行速度。

以下數節將分項敘述測試結果，並在最後分析封包分類模組之整體執行速度，與整體系統效能之瓶頸。

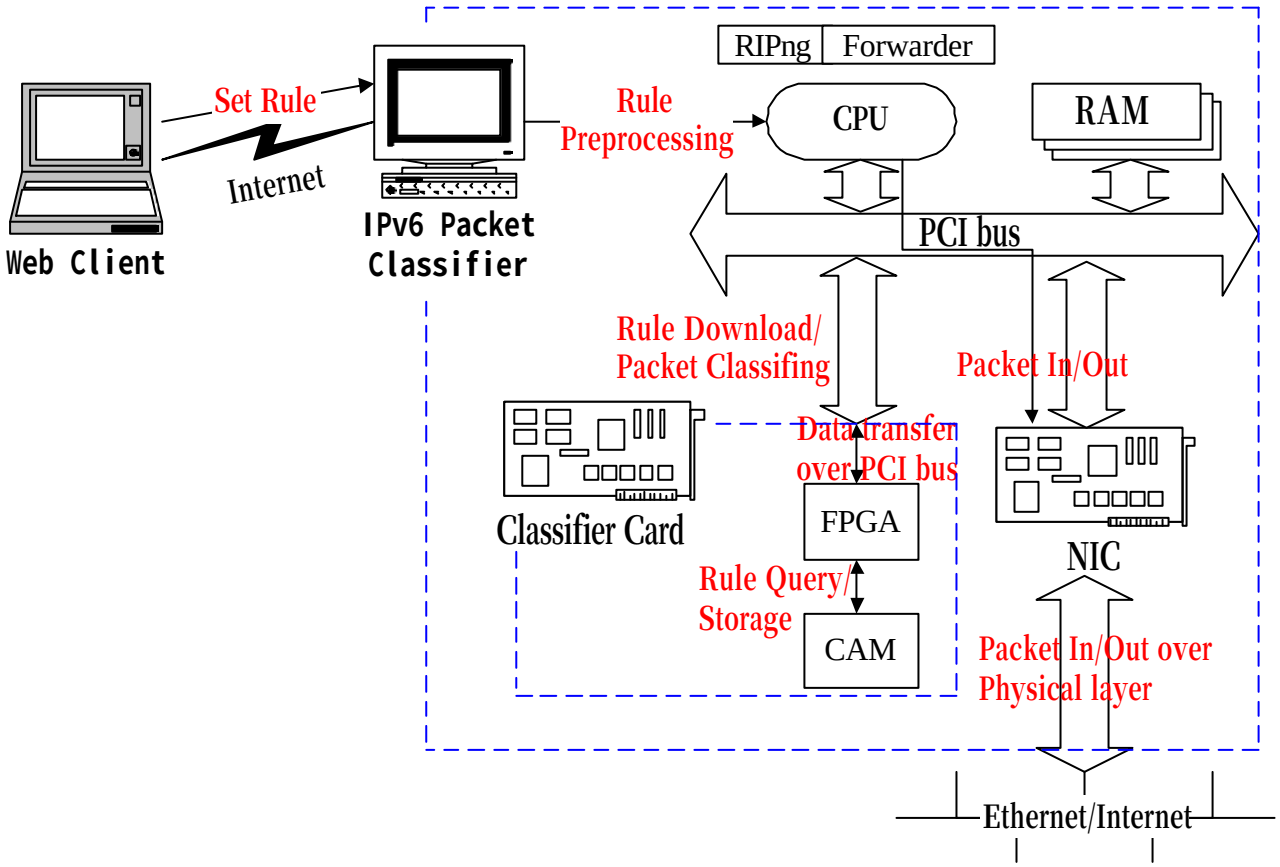

圖 6.1 IPv6 封包分類器之完整系統圖

6.1 上層軟體測試

使用者可藉由瀏覽器連線設定封包分類的規則與處理的方式，如圖 6.2。圖 6.2 左方為登入的畫面，登入後即可設定封包分類的規則，其內容有編號、名稱、來源與目的位址的內容與前置長度、來源與目的通訊阜、通訊協定等等。

圖 6.2 瀏覽器連線設定封包分類規則畫面

圖 6.3 的左方為規則設定完畢之後的畫面，也可以在實際的封包交換程式中看見設定完畢的規則，如圖 6.3 右方所示。

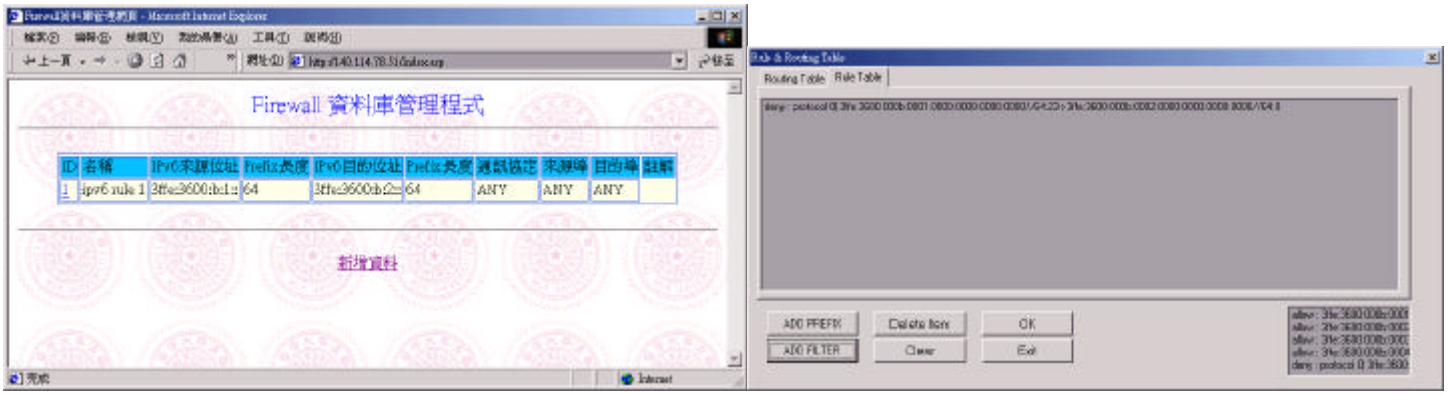

圖 6.3 已設定之封包分類規則畫面

圖 6.4 為封包交換程式的執行畫面，看起來與實際的網路設備無異，擁有每個網路連線的狀態燈號，設定的介面以及電源的開關。

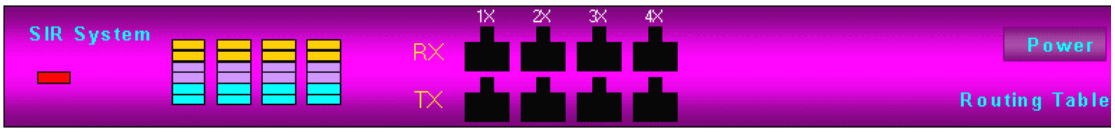

圖 6.4 封包交換程式執行畫面

圖 6.5 的上方圖形為封包交換程式設定靜態交換路徑或封包分類規則的執行畫面，下方為已經設定完畢的靜態封包交換路徑列表，右下角還有設計過程中用以除錯的訊息區，與圖 6.3 右方設定完畢的封包分類規則畫面相當類似。

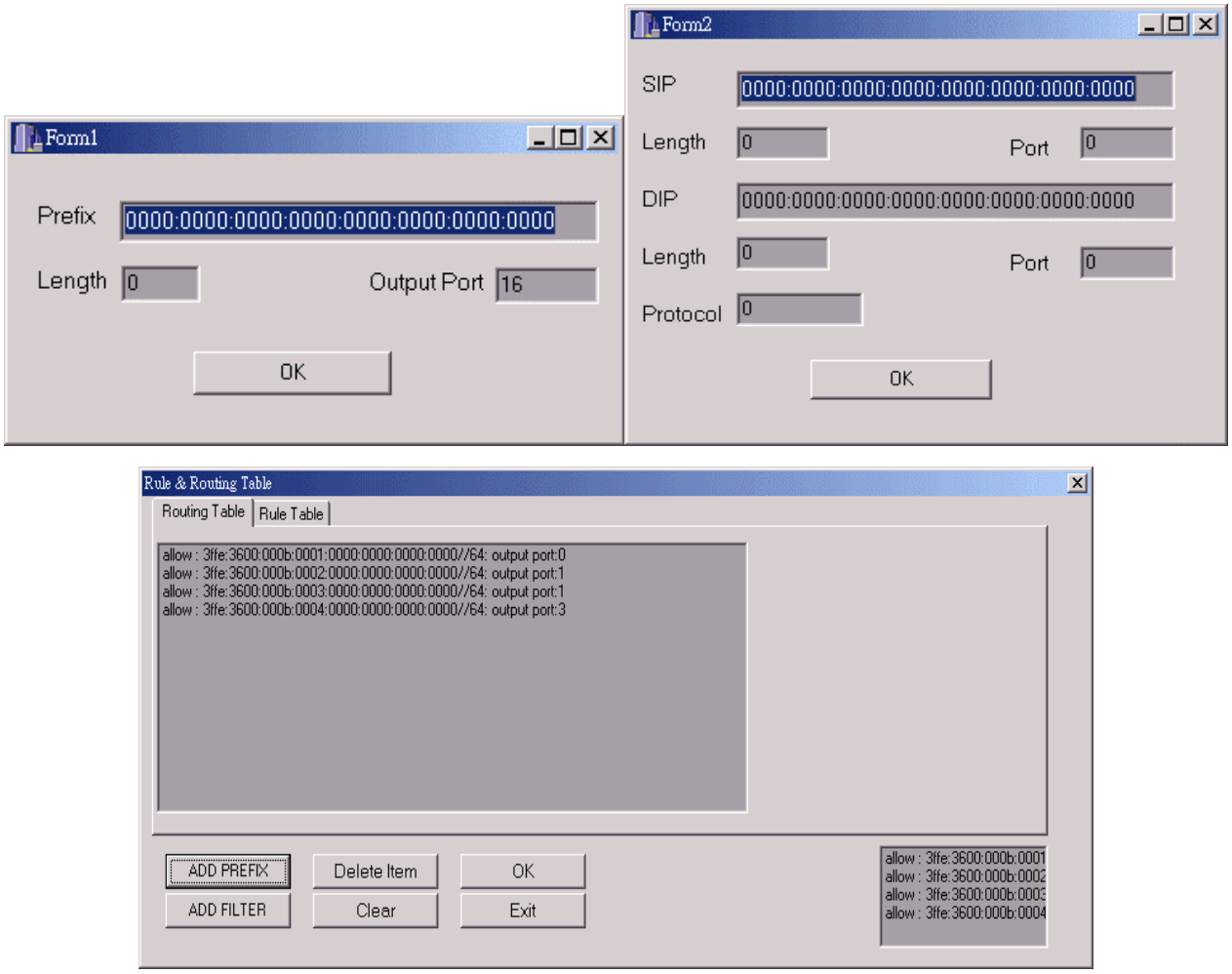

圖 6.5 封包交換程式設定畫面

利用以上的介面進行設定之後加以執行，可確定上層軟體的功能正確性，且 Web-based 與 GUI 的設計能讓使用者感覺方便而直覺。

6.2 封包分類模組與驅動程式測試

封包分類卡之驅動程式已經撰寫完成，並且已經實測完畢能夠與封包分類模組進行通訊的工作，證明其正確性。圖 6.6 為驅動程式的設計畫面，左方為 Windows NT/2000 下驅動程式編譯完成的畫面，右方為 FPGA 程式與 PCI/A 模組結合的設計畫面。

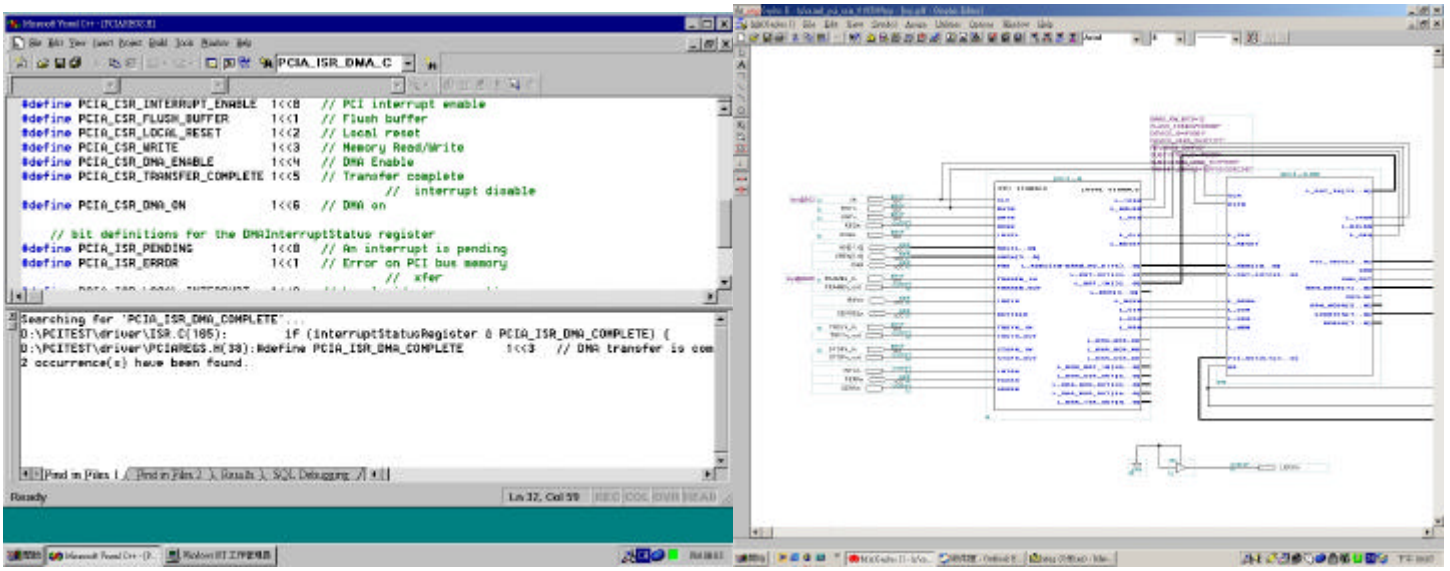

圖 6.6 驅動程式設計畫面

Windows NT/2000 與 FPGA 上的 PCI 程式執行速度與資料寬度均受到 PCI 規格的限制，分別為 33Mhz 與 32 位元。驅動程式可使用的傳輸方式有兩種，分別為：

- 使用一般的傳輸模式，傳輸一筆資料需 12 時脈週期。因此每秒最多可傳輸 2.75M 次資料，也就是 88M 位元的資料內容。
- 使用 DMA 傳輸模式，PCI/A 模組在 20 時脈週期中可傳輸 16 時脈週期的資料，因此每秒最多可傳輸 26.4M 次資料或 844.8M 位元的資料內容。

但由於封包分類時輸入的資訊與輸出的資訊各為讀出與寫入，無法將資料一起收送，故一筆封包的比對最多需要 288/32+1=10 個資料傳輸，若：

- 使用一般傳輸模式，每秒最多可傳遞 275K 個封包的資料，若封包長度均為 64 位元組，則 Wire-Speed 處理頻寬為 140.8M

位元，或在封包長為 1500 位元組時處理 3.3G 位元的頻寬。

- 使用 DMA 傳輸模式，每秒最多可傳遞 2.64M 個封包的資料。若封包長度為 64 位元組，則 Wire-Speed 處理頻寬約為 1.35G 位元，或在封包長為 1500 位元組時處理 31.68G 位元的頻寬。

但以上的速度為硬體部分的最快速度，軟體的 Windows NT/2000 驅動程式由於受到作業系統的限制、編譯程式的好壞，以及程式寫作的最佳化程度等因素影響，可能使得驅動程式無法發揮硬體最快的速度。另一方面由於 PCI 匯流排並不只有封包分類模組單獨使用，因此其他的介面卡對 PCI 匯流排的 I/O 要求也會使速度無法提升至理想值，因此在網路速度不斷提升的狀況下，勢必要放棄目前常見的 32 位元 PCI 匯流排介面改使用其他的介面。

6.3 封包分類模組與 CAM 控制測試

封包分類模組與 CAM 的搭配運作可以利用 FPGA 開發過程的硬體模擬或實體測試，但因為實際測試與硬體模擬若實體電路沒有接錯或其他錯誤，則結果就會跟硬體模擬完全相同。故在測試電路的效能時採用硬體模擬以節省時間，實際測試則用來驗證模組接線的正確性。圖 6.7 為硬體模擬之結果畫面：

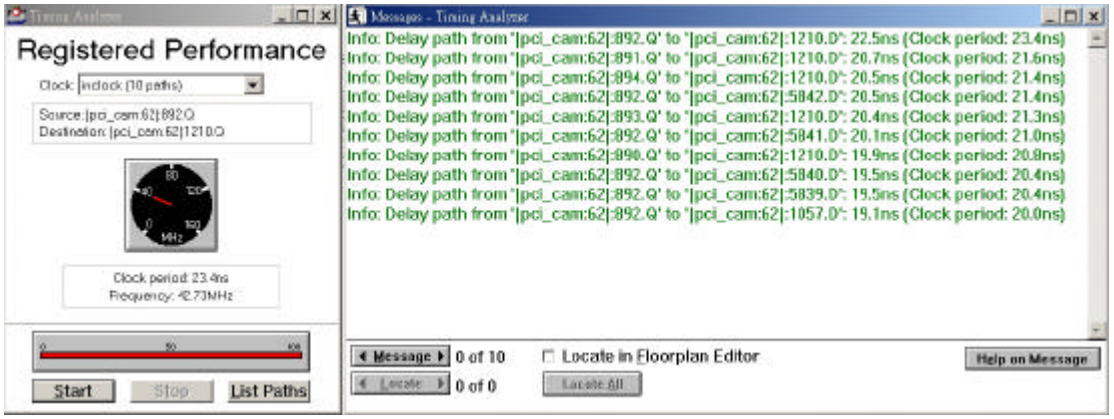

圖 6.7 CAM 與封包分類卡硬體模擬畫面

由上圖左方可以看出，CAM 的模組在測試模式下可達 42.73Mhz 的速度。上圖右方列出了時序延遲時間最長的十條路徑，藉由這些資

訊可作為進行最佳化時的依據。圖 6.8 為 CAM 與封包分類卡進行實體測試以驗證功能正確的環境。左半圖為整體的測試環境，包括待測試的模組、提供電源與輸入方波的電源供應器與波形產生器、觀測電路實際狀況的邏輯分析儀(Logic Analyzer)。右半張圖即為待測封包分類卡插上 FPGA 與 CAM，接上電源之後輸出至邏輯分析儀的四個莢(pod)共 64 個頻道。

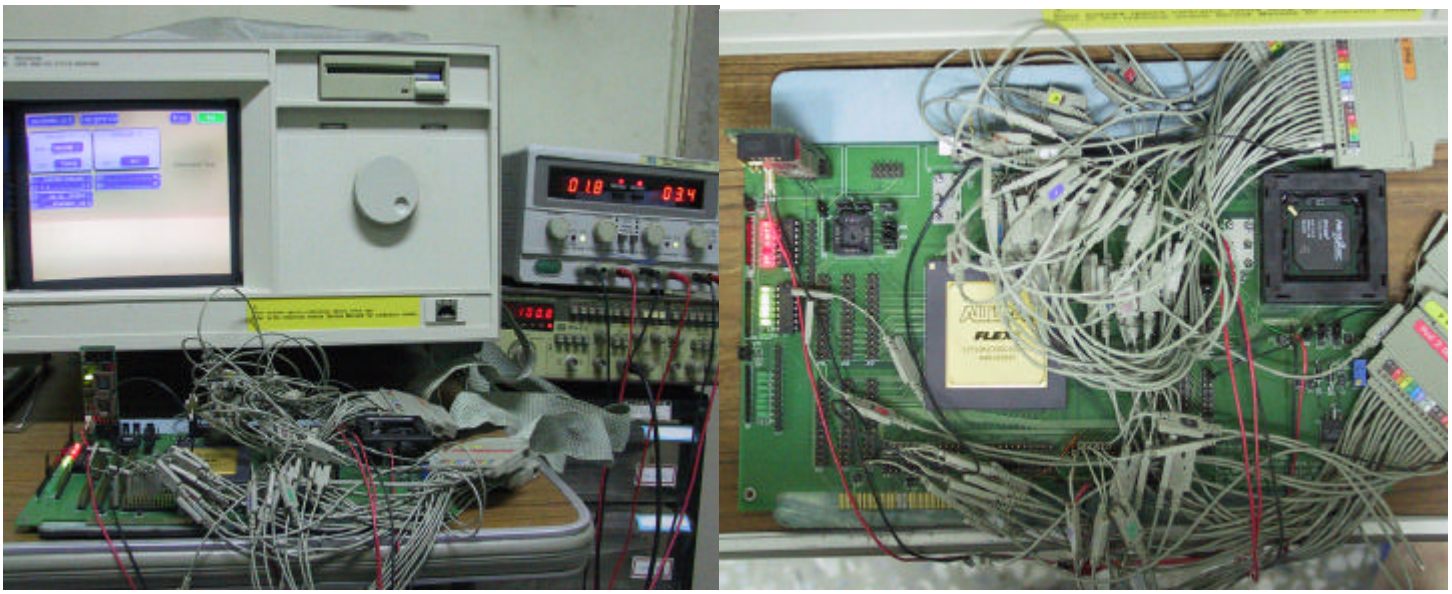

圖 6.8 CAM 與封包分類卡實體測試環境

圖 6.9 為邏輯分析儀的實際觀測情形，左半張圖是以波形的方式觀測，可以詳細的看每個接腳的狀況。右半張圖是另一種觀測模式，將某些接腳以某種方式組合成一群並觀察其值，例如將 16 個接腳以四個十六進位的數字表示等。此種觀測方法由於可以一次觀察大量的接腳，所以在測試過程中較常被使用。

由上述的測試結果可確定此硬體實作的功能正確性。而在執行速度方面，FPGA 端可執行至 42Mhz 以上的速度，而根據應用範圍與 CAM 的資料結構設計，每筆封包的分類約需要 8 至 12 個時脈的時間，若以最長的 12 個時脈計算，每秒鐘可進行 3.5M 個封包的查詢與分類，若封包長度均為 64 位元組，則 Wire-Speed 處理頻寬為 1.79G 位元，或在封包長為 1500 位元組時處理 42G 位元的頻寬。

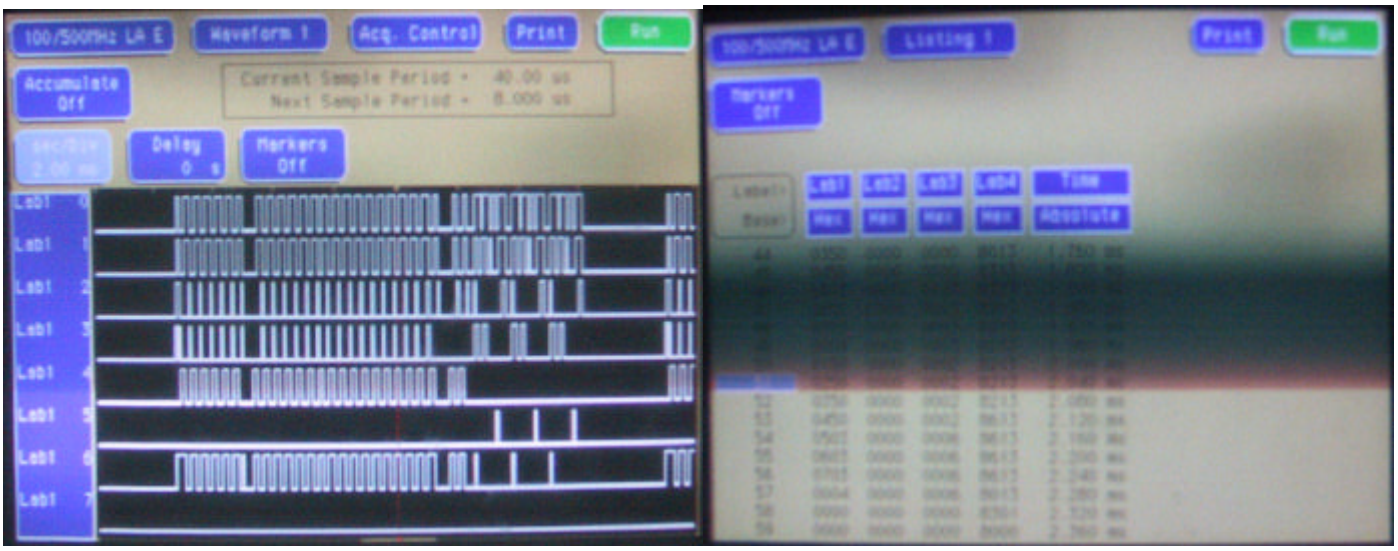

圖 6.9 以邏輯分析儀觀測 CAM 與封包分類卡實體測試畫面

6.4 封包分類模組展示系統

本論文之展示系統可分為三部分，如圖 6.10 所示。

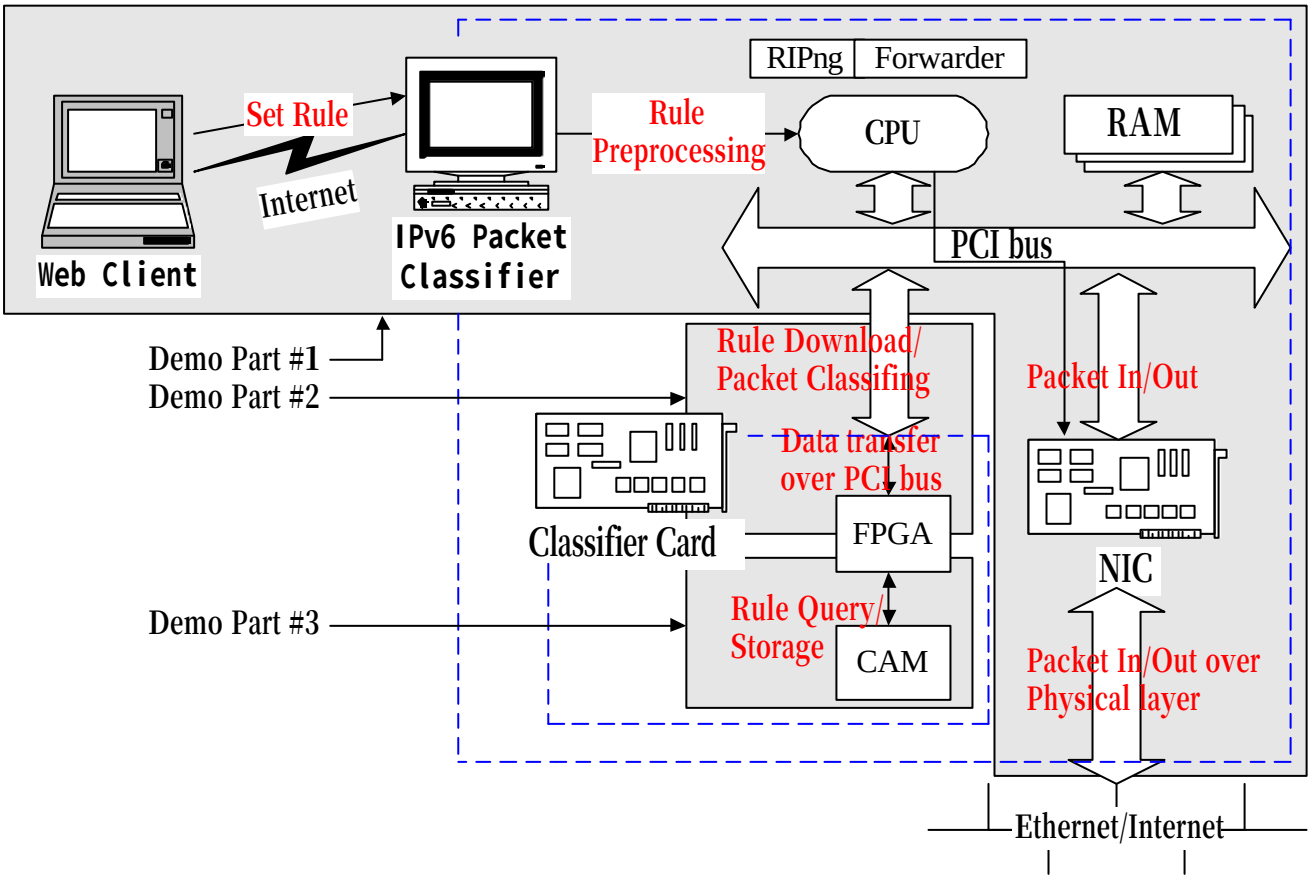

圖 6.10 本論文之展示系統

展示系統的第一部份為軟體部分，使用者可以藉由 WWW 介面設定規則，如圖 6.2 及 6.3，並藉由封包交換程式提供之介面觀看設定結果，或進行設定，如圖 6.4 及 6.5，並藉由實際的封包收送以驗證正確性。此部分的展示過程中，封包交換模組以軟體模擬的方式以確保軟體部分能驗證正確性。

展示系統的第二部分為 FPGA 與 PCI 匯流排的連結展示，利用開發軟體 Maxplus2 的硬體時序模擬進行模擬，如圖 6.11 所示。可由軟體之模擬結果準確預測硬體電路製作完成之後的執行結果，以彌補目前正在等待硬體電路製作程序之不足。

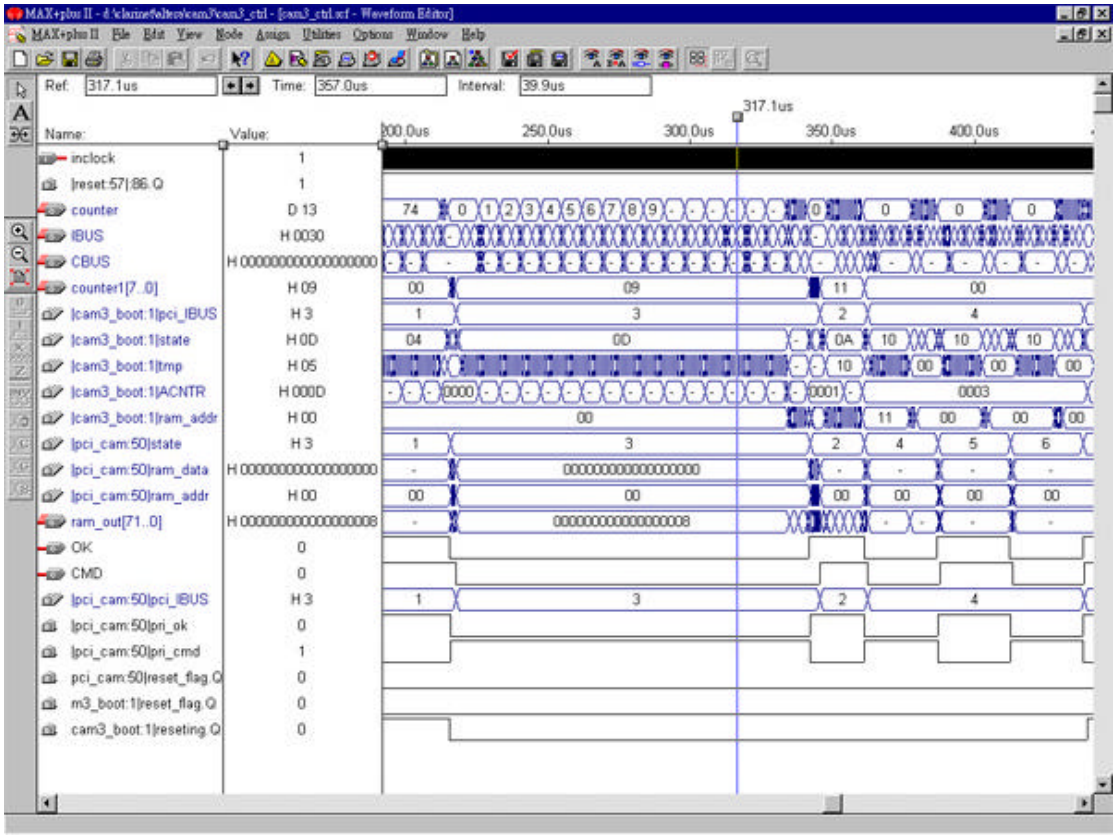

圖 6.11 硬體時序模擬波形圖

展示系統的第三部份為 FPGA 與 CAM 的整合測試，利用 JTAG(Joint Test Action Group)介面將欲對 CAM 進行測試之動作下載給 FPGA，FPGA 接收後根據這些動作給 CAM 各種命令進行測試，並利用邏輯分析儀觀察測試之結果，整體環境如圖 6.8 與 6.9 所示。可完成數百條規則的下載與數十次封包分類的模擬與測試。

6.5 封包分類模組與整體系統效能分析

由 6.2 與 6.3 兩節的測試可以得知，若不考慮上層作業系統/驅動程式的延遲，本論文實作之封包分類模組可達的速度為 PCI 匯流排與 CAM 兩部分較小的值，也就是：

- PCI 匯流排部分使用 DMA 傳輸模式的理論最大值。若封包長度為 64 位元組，則 Wire-Speed 處理頻寬約為 1.35G 位元，或在封包長為 1500 位元組時處理 31.68G 位元的頻寬。

顯而易見的，整體系統的瓶頸是在 PCI 匯流排而非 CAM 部分。因此雖然本論文實作之效能有以下的實力：

- 每秒鐘可進行 3.5M 個封包的查詢與分類，若封包長度均為 64 位元組，則 Wire-Speed 處理頻寬為 1.79G 位元，或在封包長為 1500 位元組位元時處理 42G 的頻寬。

因此如果要提升整體效能，則首先應針對 PCI 匯流排進行改善工作。若仍沿用現行的 PCI 介面則可自行撰寫 PCI 模組以加速 DMA 的傳輸，使用大量的暫存記憶體以使傳輸速度更接近理論值。或者更換更高速的介面，如下一代的 PCI 介面[46]等等以提高傳輸速率。但目前使用的 32 位元 PCI 匯流排因為開發簡單容易、規格完整、平台容易取得等優勢，雖然速度漸漸無法趕上網路成長的需求，但仍然相當適合於開發過程，若需要更高的速度再更換至更快的傳輸介面。

另一方面，根據莫爾定律(Moore’s Law)[47]，電晶體的密度會在一年半之內增加一倍，但成本卻減少一半，也就是說價格將完全不變。CAM 與 FPGA 也大致符合此定律，兩者的效能/價格比均會隨著時間同步成長，因此若要追求更快的速度，FPGA 的開發需致力於最佳化以外，CAM 與 FPGA 速度的配合也是相當重要的因素。以本論文實作為例，即使 CAM 原本設計的工作頻率就不高，但不久前推出的 CAM 已經逼近四年前推出的 FPGA 的速度，若 FPGA 不加以升級則很快就會被超過。因此硬體與硬體間的速度配合需要於設計時先行考慮，以將效能/價格比提升至最大。

第七章
結論

本論文針對目前網際網路發展的兩大問題：(1)網際網路位址，也就是 IP 位址嚴重不足，(2)對網際網路的管理已經刻不容緩 此兩大需要，提出一完整的 IPv6 高速封包分類器之系統架構，能適合於各種第四層以上網路所需要之應用，並以實作為防火牆的方式驗證其正確性。在上層的各項應用程式之外，本論文利用 FPGA 與 CAM 硬體實作出封包分類模組，由實際測試的結果可得知，此模組的速度每秒可處理 2.64M 個封包, 在處理 1500 位元組長的封包時, 相當於 31.68G 位元的頻寬，若處理 64 位元組的 Ethernet 最短封包時則是 1.35G 位元的頻寬，若更換匯流排介面則可讓此硬體平台發揮更快的速度。在高速封包分類模組與整體系統的測試完成之後，此系統尚待進行的工作包括可分為效能方面及功能方面。在效能方面可藉由軟體演算法的改進、封包分類規則的管理、驅動程式的最佳化等來提升效能，另外更換軟體平台至嵌入式系統或網路功能強大的 Unix-like 作業系統也是可行的方法。在功能方面，可以將本論文針對第四層的封包分類提升至更高層，以提供更強大的功能。與其他網路頻寬管理協定的結合，例如 Diffserv(Differential Service)等，並利用本論文實作之快速封包分類平台，相比於目前多以軟體處理封包的其他實作方法速度可大幅改進。網路處理器(Network Processor)為專門處理網際網路封包之處理器，主要應用於第三層，也就是 IP 位址以下的應用，將大幅加速第三層以下標頭之處理速度，與本論文之平台結合之後，將可分工處理第三層以下及第四層以上之標頭，成為具有強大網路管理功能及 Core-Router 等級封包處理速度之高階路由器，可作為下一代核心網路設備的雛形。

參考文獻

[1] W. Richard Stevens, “TCP/IP Illustrated Volume1”, Addison-Wesley, 1994.

[2] Robert M. Hinden, Stephen E. Deering, “Internet Protocol, Version 6(IPv6) Specification, RFC2460, December 1998.

[3] Kjeld B. Egevang, Paul Francis, “The IP Network Address Translator (NAT)”, RFC 1631, May 1994.

[4] The Internet Engineering Task Force, http://www.ietf.org

[5] Robert M. Hinden, Stephen E. Deering, “IP Version 6 Addressing Architecture”, RFC 2373, July 1998.

[6] Robert M. Hinden, Mike O’Dell, Stephen E. Deering, “An IPv6 Aggregatable Global Unicast Address Format”, RFC 2374, July 1998.

[7] Robert M. Hinden, Stephen E. Deering, “IPv6 Multicast Address Assignments”, RFC2375, July 1998.

[8] G. Malkin, R. Minnear, “RIPng for IPv6”, RFC2080, January 1997.

[9] R. Gilligan, E. Nordmark, “Transition Mechanisms for IPv6 Hosts and Routers”, RFC1933, April 1996.

[10] B. Carpenter, C. Jung, “Transmission of IPv6 over IPv4 Domains without Explicit Tunnels”, RFC 2529, March 1999.

[11] M. Waldvogel, G. Varghese, J. Turner, and B. Plattner, “Scalable high-speed ip routing lookups,” in *Proceedings of ACM SIGCOMM’97*, Cannes France, Oct. 1997, pp. 3–13.

[12] A. Brodnik, S. Carlsson, M. Degermark, and S. Pink, “Small forwarding tables for fast routing lookups,” in *Proceedings of ACM SIGCOMM’97*, Cannes France, Oct. 1997, pp. 3–13.

[13] P. Gupta, S. Lin, and N. McKeown, “Routing lookups in hardware at memory

access speeds," in *Proceedings of INFOCOM*, San Francisco, California, Mar. 1998, pp. 1240–7.

[14] B. Lampson, V. Srinivasan, and G. Varghese, "IP lookups using multiway and multicolumn search," in *Proceedings of INFOCOM*, Mar. 1998, San Francisco, California, pp. 1248–1256.

[15] S. Nilsson and G. Karlsson, "IP-address lookup using LC-tries," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 6, pp. 1083–92, 1999.

[16] V. Srinivasan and G. Varghese, "Fast address lookups using controlled prefix expansion," *ACM Transactions on Computer Systems*, vol. 17, no. 1, pp. 1–40, Oct. 1999.

[17] V. Lakshman and D. Stiliadis, "High-speed policy-based packet forwarding using efficient multi-dimensional range matching," in *Proceedings of ACM SIGCOMM'98*, Vancouver Canada, August 1998, pp. 191–202.

[18] V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel, "Scalable level 4 switching and fast firewall processing," in *Proceedings of ACM SIGCOMM'98*, Vancouver Canada, August 1998, pp. 203–214.

[19] P. Gupta and N. McKeown, "Classifying packets using hierarchical intelligent cuttings," *IEEE Micro*, vol. 20, no. 1, pp. 34–41, Jan-Feb 2000.

[20] P. Gupta and N. McKeown, "Packet classification on multiple fields," in *Proceedings of ACM SIGCOMM'99*, Cambridge, August 1999, pp. 147-60.

[21] M. M. Buddhikot, S. Suri, and M. Waldvogel, "Space decomposition techniques for fast layer-4 switching," *Protocols for High Speed Networks*, vol. 66, no. 6, pp. 277–83, Aug. 1999.

[22] V. Srinivasan, G. Varghese, and S. Suri, "Fast packet classification using tuple space search," in *Proceedings of ACM SIGCOMM'99*, Cambridge, August 1999, pp. 135–46.

[23] W. Doeringer, G. Karjoth, M. Nassehi. "Routing on Longest-Matching Prefixes." *IEEE/ACM Trans. Networking*, Vol. 4, No. 1. Feb. 1996.

[24]A. McAuley, P. Francis. “Fast Routing Table Lookup Using CAMs.” *Proc. IEEE INFOCOM*, 1993, San Francisco, USA Vol. 3, pp 1382-1391.

[25]D. Shah and P. Gupta, “Fast updates on ternary CAMs for packet lookups and classification,” *Proc. Hot Interconnects VIII*, Stanford University, Stanford, California, USA, August 2000.

[26]Dan Decasper, Zubin Dittia, Guru Parulkar and Bernhard Plattner, “Router Plugins: A Software Architecture for Next-Generation Routers”, *IEEE/ACM Trans. Networking*, Vol. 8, No. 1, Feb. 2000.

[27]NetBSD Project, http://www.netbsd.org/

[28]**李國輝**, IPv6 **交換式路由器之設計與實作**, **清華大學碩士論文**, **民國八十九年七月**.

[29]Dimitri Bertsekas, Robert Gallager, “Data Networks”, second edition, Prentice-Hall International, pp17-32, 1992.

[30]Abraham Silberschatz, Peter B. Galvin, “Operation System Concepts”, fourth edition, Addison-Wesley Publishing Company, ch15.6, 1995

[31]J.Reynolds, J.Postel, “Assigned Numbers”, RFC1700, October 1994.

[32]Microsoft NT/2000(using NT Technology)
http://www.microsoft.com/ntserver/
http://www.microsoft.com/windows2000/

[33]Network Search Engine NSE3128, Netlogic Microsystems, http://209.10.226.214/html/products/nse.html

[34]Altera Corporation, http://www.altera.com/

[35]Altera EPF200GC503-3, Altera Corporation, http://www.altera.com/literature/ds/dsf10k.pdf

[36]Altera pci/a Bus Master/Target Megacore Function, http://www.altera.com/literature/ds/pcia.pdf

[37]World Wide Web Consortium, http://www.w3c.org/

[38]Cadence Design Systems PCB Orcad eda(electronic design automation) software tools, http://www.orcad.com/

[39]Altera MAX+PLUS®II development software, http://www.altera.com/products/software/maxplus2/mp2-index.html

[40]IEEE Computer Society, “IEEE Standard Hardware Description Language Base on the Verilog Hardware Description Language”, IEEE Standard 1364-1995, 14 October 1996.

[41]Apache HTTP Server Project, The Apache Software Foundation, http://www.apache.org/

[42]MSDN http://msdn.microsoft.com/default.asp

[43]Netscape Composer, Netscape Communications Corporation, http://www.netscape.com/

[44]Visual Studio, Microsoft Corporation, http://msdn.microsoft.com/vstudio/

[45]Borland C++ builder, Borland Corporation, http://www.borland.com/bcppbuilder/

[46]PCI SIG, http://www.pcisig.com

[47]Moore’ s law, http://scisci.nctu.edu.tw/stage_old/stage/subject9904_a.htm