uTCP：微型高速 TCP/IP 協定堆疊

蘇益慶　　李志峰　　陳健維
義守大學資工系　　義守大學資工系
davidsu@isu.edu.tw　　a.chjason@gmail.com

摘要

影像感測器網路（video sensor network）可廣泛應用於保全、反恐、科學研究等，其擷取影像若要與智慧型影像處理演算法（如顏面識別）結合，則以高解析度影像為佳。針對未來高解析度、高畫質影像感測器所需之高速資料傳輸功能，本研究提出一個微型 TCP/IP 協定堆疊模組（μTCP）之軟硬體協同設計架構。其軟硬體分工方式是以一微型內嵌 8 位元控制器軟體來處理複雜的 TCP 協定流程，包括 TCP 有限狀態機的控制、重傳以及亂續封包之判定等；其餘硬體則負責高速處理重複性高但較為單純之資料搬運與檢查碼計算等功能。本研究目標是希望透過軟硬體協同設計方式結合軟體的可變性及硬體的高速性，來完成適用於高流量影像感測器之低成本、高效能的 TCP/IP 傳輸堆疊模組。

關鍵詞：TCP/IP offload engine（TOE）、TCP/IP、Video Sensor Network、軟硬體協同設計

Abstract

Video sensor networks can be widely deployed in many fields as security, anti-terrorism, and scientific researches. In case the captured images would be input to some intelligent video processors (e.g. face recognition systems), as so the higher image resolution is better for improving the ratio of correctness. To equip the emergent high-resolution, high-quality video sensors with a light-weight but high-speed data transmission module, in this project a compact TCP/IP protocol stack (uTCP) based on hardware/software codesign has been proposed. About the hardware/software partition in uTCP, a tiny embedded 8-bit controller handles the top-level complex control flow in TCP/IP by software, and the remained hardware circuits take charge of those time-consuming but regular jobs like data moving and check-sum calculating. When the clock rate is 100MHz, uTCP's output throughput can reach 1.3Gbps.

keywords：TCP/IP offload engine（TOE）、TCP/IP、Video Sensor Network、software/hardware codesign

1.前言

現今社會中，人們越來越注重自身、居家以及工作場所的安全，社會上，許多刑事案件往往需要藉由影像監控系統來破案，而高畫質影像監控網路，可廣泛應用於研究、監控、保全等用途。完整的高畫質影像監控網路解決方案，需具備影像編碼技術、高速網路傳輸以及資料儲存，然而，目前已有許多高畫質影像編碼技術，及影像硬體編碼晶片，能提供即時壓縮及擷取高畫質影像。相較之下，雖然目前存在許多高速網路解決方案，如具備 TOE[3]（TCP/IP Offload engine）技術支援之網路卡。針對高畫質影像網路監控應用而言，一顆適用於嵌入式系統之高速 TCP/IP 網路晶片，通常其價格過高，且需要作業系統支援，無法有效降低成本。因此，本研究希望設計出一顆低成本、低電耗、高效能且無須作業系統支援之網

路晶片硬體架構，並能獨立執行於高畫質影像監控網路，以得到更為有效之高速網路解決方案。

針對高畫質影像感測網路而言，高速傳送龐大的影像資料，必定需要高效能傳輸網路，然而現今傳輸介面的設計方式多為以下兩種方法。第一，由 CPU 配合由軟體設計之協定來完成 TCP 傳送，雖然此種解決方案有許多優點，卻因網路介面發出多次 I/O 中斷服務，使得 CPU 需花費過多資源用於控管網路傳輸流程，造成傳輸效率不甚理想等問題，以致於網路速度無法提昇。第二種方式則是採用軟硬體混合設計，不但擁有軟體在 TCP/IP 協定上之可變性，並結合硬體能同步搬移大量資料的特性，來達成低成本，高效率之目的。

本研究提出適合應用於高畫質影像感測網路之微型網路晶片之硬體架構，稱為 uTCP[1][2]，具備高速傳送封包之能力，採軟硬體混合設計方式，設計一顆微型且高效能之 TCP/IP 網路晶片，使高畫質影像感測器能將大量影像資料在經過編碼之後，透過網路晶片將資料封裝成 TCP 封包格式後，高速地將封包送至網際網路。如圖 1。

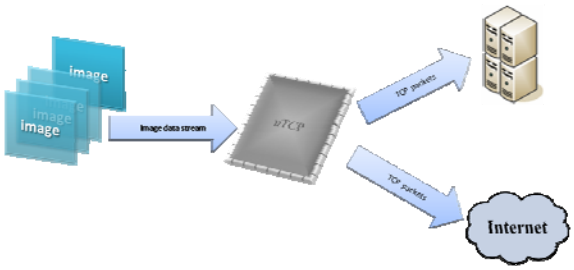

圖 1. uTCP 之應用

硬件設計方面，為了讓微型網路晶片能廣泛應用於多種晶片設計之中，因此硬體架構設計為適合應用於 FPGA 及 ASIC 上之矽智財（Silicon Intellectual Property 簡稱 SIP）。

2.相關研究

2.1 TCP/IP offload engine（TOE）技術發展

網路傳送過程中，封包流量以及 TCP/IP 協定控制訊號通常兼由 CPU 控制處理，在大量傳送資料的情況之下，網路傳送速度將受限於 CPU 時脈，思考著將網路協定的處理獨立於 CPU 之外，讓應用程式與資料的輸出/輸入分別執行於不同的硬體，而這個為了處理網路協定所量身打造的硬體架構，稱為 TCP/IP 網路協定卸載引擎(TCP/IP Offload Engine ; TOE)[4]。如圖 2。

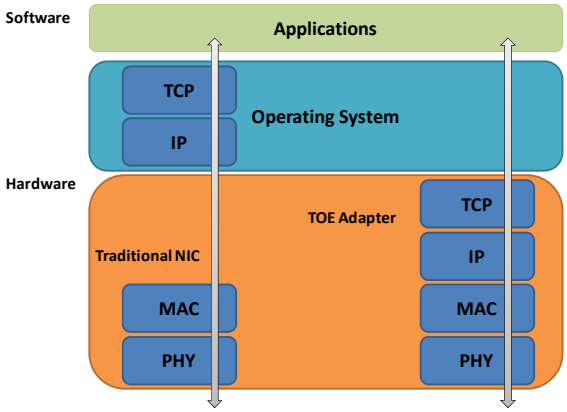

圖 2. TCP/IP offload engine

在 TOE 的研究上，大致上可分為三種設計方式。第一，直接將原有網路協定移植至一特定處理器運行的方式，為三種方式設計思維裡最快速且最具彈性的設計方式，但其缺點通常伴隨著效能不彰等問題。而相對於軟體的協定卸載，純硬體的協定加速器，具有其效能與成本優勢，卻也是最不具設計彈性。而第三種方法則融合了上述方式的優點並減少其缺點的影響，主要是將網路協定的封包處理由一個特定處理器控制協定部份並配合一個硬體加速器稱負責處理資料流部分。這種設計的優點是具有較短的研發時程與較高的彈性，而其主要的成本則來自於處理器本身的授權費。

2.2 IPAC-E100

IPAC-E100[5]為一顆商業化的晶片，由德國 Adescom 公司所開發，該公司主要研發 VOIP 等產品。此晶片主要是代替一般嵌入式系統所採用之 TCP/IP 軟體解決方案，降低 CPU

資源負擔，取消作業系統支援 TCP/IP 協定，因此設計一顆高速且微型的 TCP/IP stack 晶片，能提供有成本考量之平台作為應用，以降低成本，降低記憶體所佔空間，提昇 CPU 效率，提昇整體效能，以最少資源得到最大效益。圖 3 為 Adescom 公司所提供之比較圖。採用軟體 TCP/IP Stack 以及硬體網路晶片之網路解決方案，相較之下，針對某些特定應用，甚至可以取消 CPU、作業系統等開發時所需之軟硬體以及時間。由此可見，微型網路晶片的應用，並非局限於特定應用，甚至能取代一般應用，降低 CPU 負擔，更能將系統效能發揮的淋漓盡致，將其效益發展至最大化。

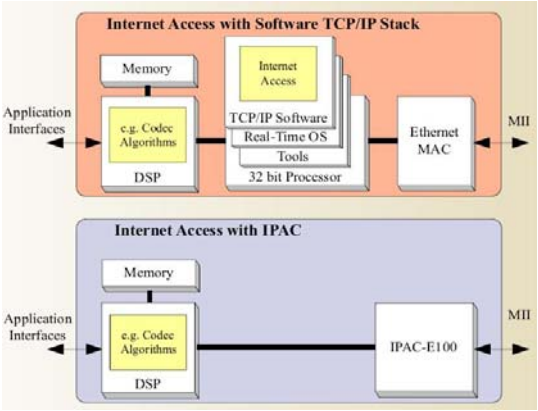

圖 3. 嵌入式 TCP/IP stack 與 IPAC-E100

為了達到高效率低成本之微型網路晶片矽智財，針對圖 4 所完成之 IPAC-E100 硬體使用率以及網路傳輸效率，將是我們邁進的目標。

	IPAC-E100	32 bit Processor System + Ethernet MAC
Sustained Bit Rate	100 Mbps	max 40 Mbps (UDP only) max 5 Mbps (TCP only)
Operating Frequency	25 MHz	150 MHz
Gate Count	95 k	~ 150 k + Cache
Embedded Memory (w/o data buffers)	6 kByte	50 kByte

圖 4. IPAC-E100 之效能與資源比較表

2.3 TCP/IP Offload Engine (TOE) for an SOC System

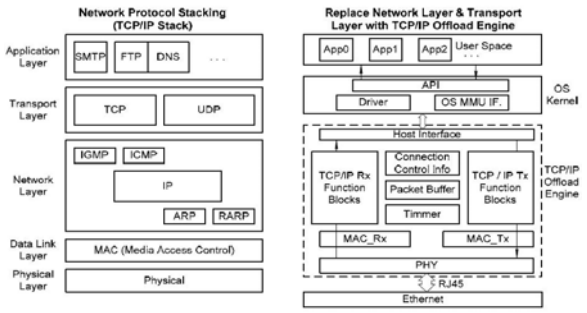

圖 5. TOE for a SOC System application

圖 5 為 TCP/IP offload engine for SOC 設計架構[6]，左邊為一般作業系統所提供協定堆疊，相較於右半邊設計一 TOE 硬體架構，其 TOE 設計架構取代 Network layer 與 Transport layer，主要應用為設計一硬體加速器用以降低 CPU 流程控制以及封包搬移等負擔。

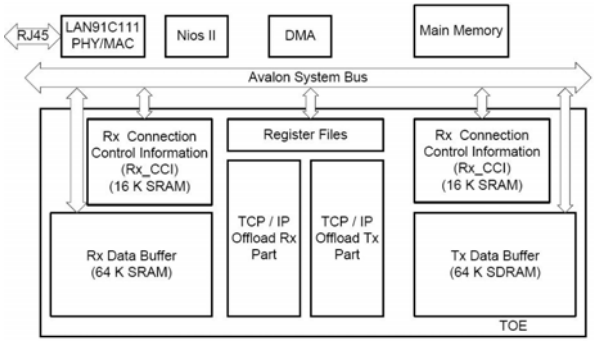

圖 6. TOE for a SOC System hardware architecture

圖 6 為 TCP/IP offload engine for SOC 之硬體架構[6]，將封包搬移設計一組硬體加速器並結合 Nios II CPU 主管協定流程控制，將其整合於 Avalon Bus 上，實做於 Altera 所提供之 FPGA 上，協定規格上實現 8 條 UDP 線程，ARP 及 ICMP，以供應用程式使用，硬體上共配置 160k bytes 記憶空間。

3.系統架構

3.1 uTCP 軟硬體協同設計

設計微型高速網路晶片通常以軟體或硬體實現之，但在嵌入式系統層級能獨立運行 TCP/IP 網路協定並能傳送高畫質影像的前提

之下，若以軟體方式實現，雖然程式撰寫較為簡單，但執行效能無法滿足高畫質影像之傳送需求，若以全硬體方式設計，卻因 TCP/IP 協定之有限狀態機過於複雜，硬體不易實現，而實做後有可能會造成硬體體積過於龐大，再者，爾後若需增加新的通訊協定，將必須重新修改硬體架構。因此，結合軟體和硬體之長處，採行軟硬體混合設計，將具有重複運算性質的工作交由硬體實現，有限狀態機判斷以及複雜性高的工作則交由軟體來完成。

在選擇處理器方面，由於 16 位元或 32 位元以上的處理器成本較高，因此在設計上選擇 PicoBlaze[7]為一顆 8 位元控制器。本研究將 TCP/IP 協定分成 8 位元控制器 PicoBlaze 以及硬體加速器來完成，並將 uTCP 設計整合一顆 Opencores[8]網站上所提供之 10/100/1000 之 Media Access Control（MAC）控制器，如圖 7，實現於 FPGA 上，得以提供 uTCP 完整的驗證環境。

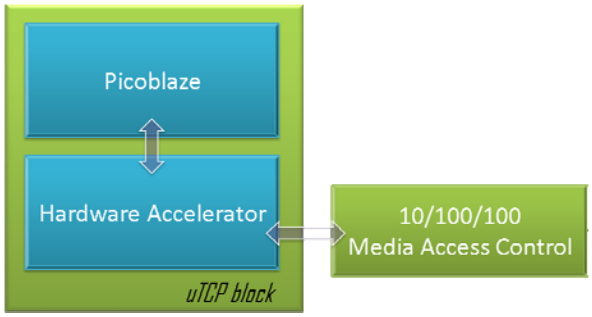

圖 7. uTCP 方塊圖

3.2 uTCP 硬體架構

uTCP 為三大硬件區塊所組成，如圖 8，分別為 PicoBlaze、In module 以及 Out module。各自處理 TCP/IP 協定流程，以及收送封包等工作。uTCP 配置了一塊資料記憶體，以及兩組暫存器組，分別為 Control registers 以及 Transmission Control Block registers（TCB）。Control registers 主要作為 uTCP 與 user 端之間的溝通橋樑，PicoBlaze、Out module 與 In module 則是透過 TCB registers 來完成，使得 PicoBlaze 能與 Out module 及 In module 兩模組間互相傳遞資料以及控制訊號。

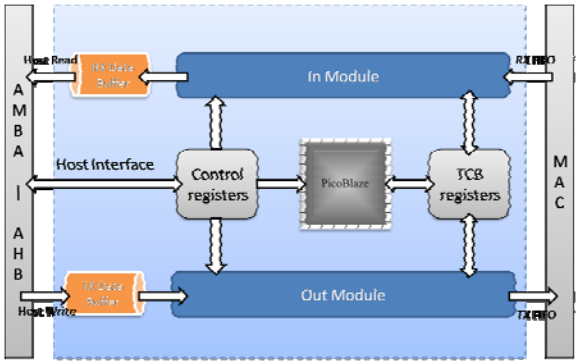

圖 8. uTCP 架構圖

uTCP 未來將設計一組 AMBA bus，提供與 CPU 或是其他 SIP 做為溝通用的介面。外部裝置可透過 AMBA bus 對 uTCP 的 Control Registers（使用者暫存器）進行參數命令等設定。另外，MAC 則是透過 FIFO 介面與 uTCP 之 In module 及 Out module 連結。

3.3 PicoBlaze

由 Xilinx 公司所提供之 PicoBlaze 8 位元控制器，PicoBlaze 是針對 FPGA(Spartan-3、Virtex and Virtex-II Pro）所架構最佳化後之 8 位元控制器，運行於 FPGA 上效能最高可達 200MHz（Virtex-II Pro），相當於 100MIPS 的執行速度，且原始碼已經對應 Gate Level 之 Library，是顆非常微小的 8 位元控制器。PicoBlaze 具有 1K word 指令寬度為 18bit 的程式空間，16 組 8bit 暫存器，及支援 31 組 10bit 硬體堆疊，供 Interrupt、Return、Returni、Call、Reset 等回返型指令使用，因此在撰寫程式碼時，需要針對程式空間大小進行評估，將程式碼控制在一定範圍之內。PicoBlaze 也提供 64Bytes 的 Scratchpad Ram 以增加額外內存空間，將利用此暫存空間，配合 TCB 表內之暫存值，進行解析，經由演算過程後，下達 TCP 狀態控制命令，完成一次狀態機的控制。透過 In Port 和 Out Port 各 256 組擴充埠，與外部記憶體或是其他裝置相互連結，例如：做為 TCB 表暫存器

的溝通介面。再者，將 PicoBlaze 與不同裝置連結，擴充其功能性，增加控制器之功用，使軟體使用者具有一定程度之擴充能力。在軟體使用者的開發過程中，將 PicoBlaze 加上 JTAG 除錯介面，軟體開發者能直接透過 JATG 介面將程式載入 PicoBlaze 驗證，加速程式驗證，進一步降低程式的開發時程。

PicoBlaze 在資源利用方面佔據非常少的資源，由於 PicoBlaze 硬體架構已針對 FPGA 作最佳化，在實際應用於 Spartan3 系列 XC3S200FT256 之 FPGA 實驗版測試後，僅佔用 151 組 slice 以及 167 組 LUT 共 7%硬體資源，效能可達到 88MHz，44MIPS 的速度。

PicoBlaze 主控 TCP/IP 協定流程管理，收送封包的過程中，透過 TCB registers 對 In module 及 Out module 發出控制訊號，收送 TCP 封包的過程中，PicoBlaze 完全不參與資料搬移之動作，僅存取封包標頭資訊，以判斷資料是否正確。因此，TCP/IP 之協定控制，全權交由 PicoBlaze，並非由硬體設計其有限狀態機，此外，PicoBlaze 實現簡化後的 TCP 協定，約使用了 500 bytes word 的程式空間。

3.4 Out module

Out module 主管封包傳遞的工作，uTCP 是以傳送為主之 TCP 層級晶片，為加快傳輸效率，將 Out Module 資料寬度提昇為 32 位元寬，以提昇其傳送速度。

如圖 9，Out module 在整體 uTCP 的設計中，為最重要的一個硬體單元，Out module 主要工作為傳送封包、整合封包標頭、計算 TCP 資料層級之 checksum 等。程式撰寫過程中，將 Packet Generator FSM 模組分作兩個平行工作區塊，分別為 Header FSM（finite state machine）與 Data FSM，藉由判斷 TCB 控制旗號，以決定有限狀態機器之運作模式。

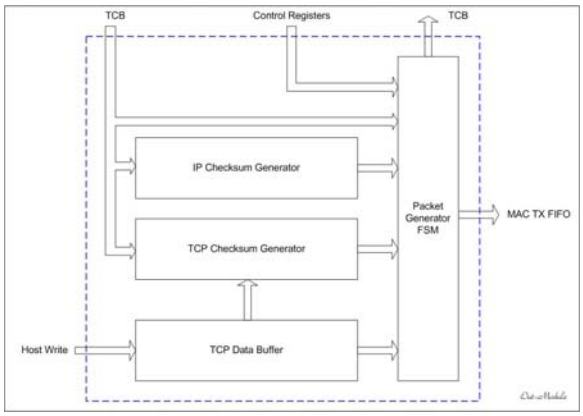

圖 9. Out module 架構圖

如圖 10，發起一次傳送前，首先，Out module 至 TCB registers 讀取資料，判別所要求傳送封包為 header 封包或是 data 封包，如判別為傳送 data 封包，則 Header FSM 由 TCB registers 讀取資料後，負責將 IP header 以及 TCP header 等資料封裝，經由 IP header checksum 計算之後，將各資料填寫進封包 header 相對應位置，並且等待 Data FSM 將資料計算完成。Data FSM 區塊則是在判斷 TCB registers 所指定之記憶體起頭位置，以及資料長度後（由 PicoBlaze 將起頭位置以及長度填入至 TCB registers），前往資料記憶體裡讀取資料（此時，不需要 PicoBlaze 資料搬移管理），並採用兩層管線的架構，同時進行資料搬移以及 TCP data checksum 之計算，經過 checksum 計算後的封包，存放於 Out module 之 Data buffer，其 buffer 具有 1460K bytes 之空間，計算方式為每筆資料以 32 位元執行加法運算，最後將 Data buffer 裡的高 16 位元以及低 16 位元資料再做一次相加後，取 1 的補數，得到正確 TCP checksum 之結果。最後，將兩區塊資料封裝後，送至 MAC FIFO，請求 MAC 將此一 TCP/IP 封包送出。

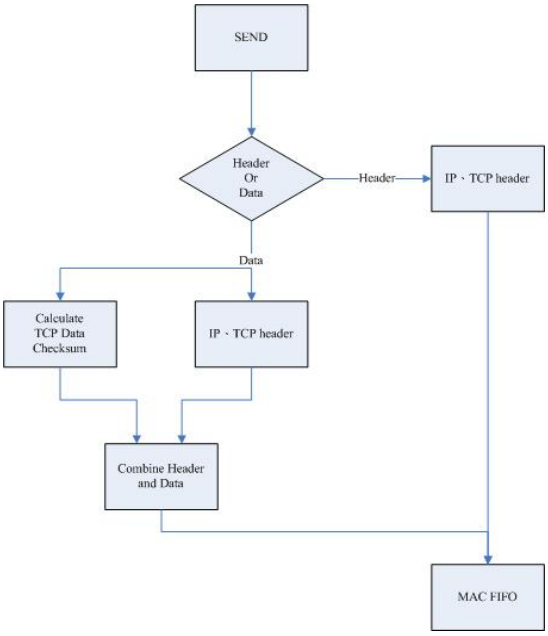

圖 10. Out module send

由於 uTCP 應用於 TCP 層級之運算，因此 MAC 以及 IP 層在 uTCP 初版的設計過程中，簡化驗證環境所造成之不確定性，Out module 所要求送出之封包，MAC type 欄位一律填入 0x0800。相同地，IP 層目前也僅支援 IPv4 以及 TCP 封包之傳遞任務，並且在 IP 以及 TCP 層皆不支援 option 之欄位應用。

3.5 In module

In module 主管封包接收，為 uTCP 硬體三大元件之一。如圖 11。In module 專注於監聽 MAC 端是否收進一包完整封包，當 MAC 完成封包接收動作時，In module 會將資料收進 In module Data buffer 裡，並進行封包解析。In module 將運作流程分為四塊子區塊進行解析，分別為 Header FSM、IP FSM、Data FSM 以及 TCB FSM。

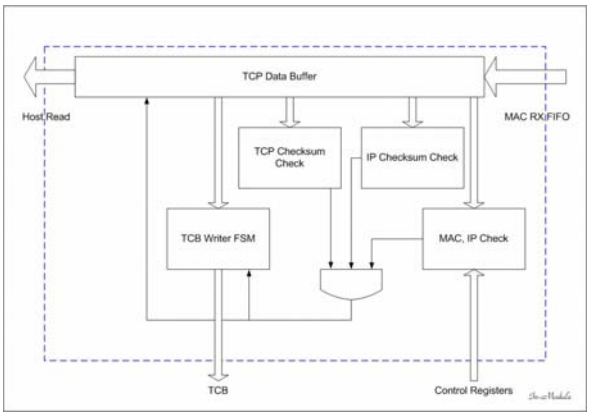

圖 11. In module 架構圖

Header FSM 將封包內 IP address 及 TCP port 取出，與 TCB registers 值進行比對後，判斷是否相同，如比對結果相同，將告知 TCB FSM 此為正確封包（送出 true 訊號）後，此狀態機進入等待狀態，假如收到錯誤封包（送出 false 訊號），告知 TCB FSM，接著，TCB FSM 將此包封包丟棄，並告知 Header FSM、IP FSM 以及 Data FSM 停止計算。IP FSM 以及 Data FSM 分別計算 IP header checksum 以及 TCP data checksum，如上述方法，結果正確之封包送出 true 訊號，錯誤封包送出 false 訊號，以告知 TCB FSM。TCB FSM 確認封包驗證之相關工作正確無誤後，將 slide window、sequence number、acknowledgement number 等標頭資料依序寫入 TCB registers，最後將 In module Data buffer 中資料區塊寫入資料記憶體中。在 In module 驗證封包 checksum 的同時，同步計算該封包長度，並將結果填入 TCB registers 裡，最後將 TCB registers 的 package arrive bit 拉起，告知 PicoBlaze，完成一包封包接收，如圖 12。

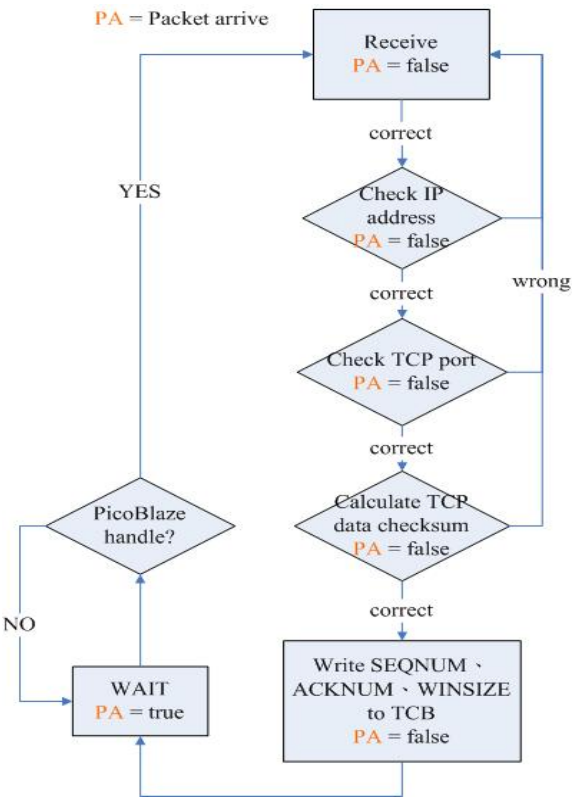

圖 12. In module receive

In module 具備 16 位元資料寬度，能透過各層級的驗證，將正確封包收下，降低 PicoBlaze 計算負擔，使用者透過 Control registers 之設定 MAC address、IP address、TCP port number 等欄位，進而過濾出 uTCP 所需要之封包。本研究所使用之 MAC 能將正確 MAC address 之 Ethernet type 封包存放至 MAC FIFO 裡，直到 FIFO 滿為止，而 In module 會依序讀取 MAC 所擷取之正確封包，因此 In module 必須具備一組緩衝來暫存驗證中之封包，當完成驗證後，才將資料寫入資料記憶體裡，交由 PicoBlaze 處理該封包，最後透過 PicoBlaze 設定 Control registers 告知 user 將封包取走，完成一次收封包之程序。

3.6 TCB Registers and Control Registers

TCB registers 作為 PicoBlaze、Out module 與 In module 之溝通暫存器組。由於此暫存器組為三模組互相溝通之介面，若將此暫存器組設計成 3 ports 暫存器，將會產生資料相依性對資料存取之間的效能產生影響。因此，製作 TCB register 的同時，將各模組間，進行資料歸納、架構優化等程序，將各模組間資料相依性降至最低，取消模組間的競爭，優化暫存器架構。

Control registers 作為 PicoBlaze 以及使用者介面的溝通暫存器。User 能透過此暫存器，設置 uTCP 以及 MAC address、IP address 以及 TCP port 等，uTCP 也藉由此暫存器組告知 user 存取之記憶體區段。

3.7 uTCP 記憶體

uTCP 配置一組內部記憶體，為資料暫存空間，並透過 AMBA bus 與使用者連結，其中 In module 與 Out module 各具備一組專屬記憶體控制器及記憶體區段，此內部記憶體做為傳送資料以及接收資料之暫存記憶體。

雖然配置內部記憶體會增加一部分的硬體資源，卻能帶給我們巨大的利益，此法不但降低與外部記憶體存取次數，減低 bus 傳遞資及控制訊號所造成的損耗。並且，針對 AMBA 設計角度而言，能取消 AMBA Master 控制器之角色（發起存取記憶體的控制訊號），僅需實現 AMBA Slave 控制器。因此，uTCP 配置內部記憶體，簡化 AMBA 記憶體控制器，為實現速度與成本協調上所提出最好的設計方法。如圖 13。

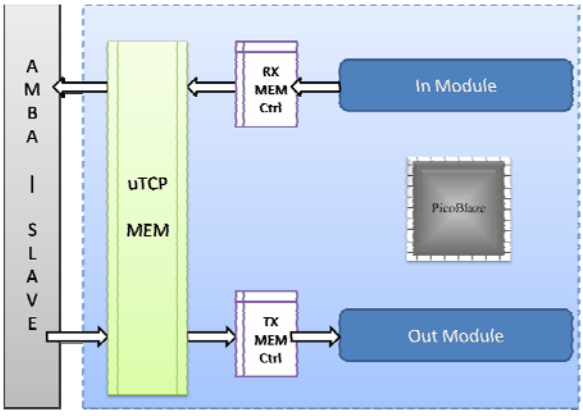

圖 13. Memory architecture

對於 TCP/IP 協定之特性，本研究提出一

個整合型硬體架構，將記憶體定址功能與 slide window、sequence number 以及 acknowledgement number 等 TCP 封包標頭資訊與記憶體定址功能予以結合，稱 SEQ 定址法。

資料記憶體的管理，原本應由 PicoBlaze 配置一組 memory mapping table，並透過類似 Link list 的方式配置封包與記憶體之間 sequence number 與 memory address 之對應關係。但 TCP/IP 協定具有錯誤控制以及壅塞控制之機制，通常發生情況為封包重傳、封包遺失以及 slide window 管理，由此可以輕易發現，如果將所有的重傳演算法以及 slide window 等計算交由 PicoBlaze 處理，那麼 PicoBlaze 將負擔龐大計算量，以及耗損約三分之一以上的程式空間，造成各項資源無法有效的利用，對傳輸速度、效率以及成本都是一大傷害。

因此，本研究提出一個整合型記憶體控制模組，將封包內的 sequence number 取模數後直接映射至記憶體位置（模數為記憶體空間大小），由硬體判斷 TCB registers 的 sequence number，將 sequence number 轉址至實際記憶體位置。例如，記憶體大小配置為 8K bytes，此時收到一筆 sequence number 為 10000 的封包，那麼將 10000 mod 8K 等於 2000，得到封包起頭位置為 2000 並從此處開始存放封包，最後，配合 TCB registers 裡封包長度欄位來得知此封包長度。同樣的，當重傳發生時，對方所要求之 acknowledgement number 經由轉址機制，能立即得知封包起頭位置，簡化重傳發生時，所帶給 PicoBlaze 之龐大計算量，以利整體效能往上提昇，相對的也降低程式設計之複雜度及程式空間。

3.8 DMA

uTCP 提供 DMA 傳輸功能，來提昇傳輸效率。當使用者欲對 uTCP 內部記憶體發起一次傳送或接收時，就可利用 DMA 功能發起對連續性記憶體位置進行資料存取，表示使用者可利用 DMA 之功能進行一次傳送或接收超過 1460 bytes 之資料，以下簡介使用者發起傳送及接收之動作。

欲啟動 DMA 寫入模式時，user 透過 AMBA 發出控制訊號，設定 Control registers 裡的 Out buffer start address 及 Out buffer length 來完成資料區段之設置，當 user 完成暫存器的設置時，並將資料搬移至 uTCP 的 buffer 中，啟動 Host flag（HF）以通知 uTCP 傳送該區段資料，達成 user 能夠透過 uTCP 所提供的 DMA 模式，將資料大筆且連續的寫入或讀出。PicoBlaze 能透過 Control registers 的設定來告知 user interface 此時 uTCP 內部記憶體剩餘多少空間，讓 user 得知可存放資料的起始位置以及最長的資料長度。

同樣地，PicoBlaze 利用相同方式，偵測 In module 收完封包之後，PicoBlaze 也會藉由類似的設定方式，對 In buffer start address 及 In buffer length 進行資料起頭位置和資料長度之設定，來告知 user 該提取哪些區段的資料，並且透過 In and Host semaphore（IHS）的設定，告知 user 何時可讀取正確的資料區段。

3.9 uTCP 韌體架構

程式架構，分為 TCP 協定與使用者呼叫兩區塊，主要以實現簡化之 TCP 協定[2]為主，共實現 CLOSED、LISTEN、SYN-RCVD、SYN-SEND、ESTABLISHED 等五種狀態，協定撰寫部份約佔用 500 行程式空間。如圖 14

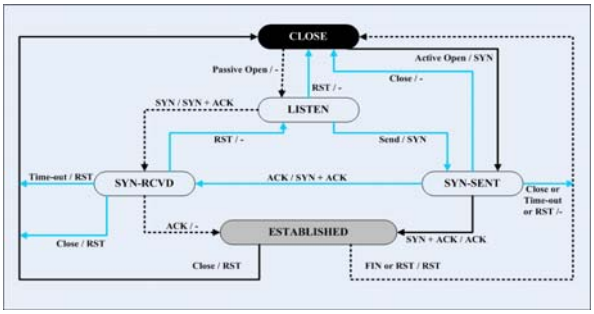

圖 14. 韌體架構之有限狀態機

4.實驗結果

4.1 實驗步驟

本研究分為兩階段驗證階段，第一階段使用硬體描述語言設計出In module及Out module並與PicoBlaze整合成一顆uTCP雛型架構，接著使用ModelSim結合Verilog PLI（Programming Language Interface）之驗證環境，進行硬體及TCP/IP協定之正確性測試。第二階段，採用FPGA驗證環境，將PicoBlaze與硬體描述語言設計出的In module以及Out module進行整合。在驗證初期，利用PicoBlaze所提供之JTAG軟體下載介面整合至PicoBlaze上，將欲下載程式利用JTAG介面，輕鬆的下載至PicoBlaze之程式記憶體，以提供韌體更新介面。

模擬環境中，uTCP設定為TCP/IP協定中之Client角色，TCP Server則由一台Linux桌上型電腦負責，此桌上型電腦具備1Gbits的網路晶片，並執行由C利用raw packet所撰寫之TCP echo server程式，進行封包傳送測試，並驗證其連線正確性。

4.2 模擬程式

模擬程式方面，由PicoBlaze以組合語言實現之，PicoBlaze以簡化後之TCP/IP協定為設計藍本，完成韌體設計後，撰寫測試程式至PicoBlaze進行封包收送，以及狀態轉換正確性等驗證。以下圖15為ESTABLISHED程式片段。

```
700 SEG_ARRIVE_E_FIN:
701     input   s6, R_flag
702     test    s6, 01
703     jump    NZ, SEG_ARRIVE_E_FIN0
704     ;write L_Seq = SND_NXT , L_Ack = RCV_NXT
705     fetch   s8, SND_NXT0
706     fetch   s9, SND_NXT1
707     fetch   sA, SND_NXT2
708     fetch   sB, SND_NXT3
709     output  s8, L_Seq0
710     output  s9, L_Seq1
711     output  sA, L_Seq2
712     output  sB, L_Seq3
713 ;calculate to send how much datas and calculate SND_NXT
714     call    SEND_SET
715     call    SEG_ARRIVE_E_RACK1
716     ; set ACK flag and call outmodule to send the packet with data
717     call    SEND_PACKET_ACK_D
718     call    RCV_PACKET_FINISH
719     return
```

圖15. ESTABLISHED

4.3 合成結果

合成電路時，採用Xilinx公司所提供的ISE合成工具。在合成時，本研究採用兩種FPGA實驗版，一版為SPARTAN-3E 1200的FPGA版本，另一版為VIRTEX-5 LX50T的FPGA。由於，兩者FPGA在先天效能具有相當大的差異，SPARTAN-3E是速度較慢但成本也較低的開發版，相反的VIRTEX-5則是速度快價位也較高的開發版。以下表1則是列出兩開發版所佔用資源以及合成速率。

表1. 合成數據比較表

FPGA Type / Resource	SPARTAN-3E 1200	VIRTEX-5 LX50T
Slice LUTs with GMAC（佔總資源之百分比）	3413（19%）	2488（8%）
Slice LUTs without GMAC（佔總資源之百分比）	1856（10%）	1394（4%）
Speed	61 MHz	100 MHz up ↑

另外，合成結果中，如圖16、圖17、圖18、圖19所示，分別提供uTCP本身合成數據外，一併附上與Opencores（OC）網站上所提供之一顆GIGA Bits Media Access Control所合成之數據。合成結果中，整體速度受限於PicoBlaze的執行時脈，為整體時脈無法向上提昇之原因。但在整體面積使用率上，僅佔用FPGA小部份的面積，展示uTCP具備體積小速度快等特性，進而達到低成本且高速之TCP晶片的目的。

Device Utilization Summary

Logic Utilization	Used	Available	Utilization	Note(s)
Total Number Slice Registers	2,200	17,344	12%	
Number used as Flip Flops	2,184			
Number used as Latches	16			
Number of 4 input LUTs	3,413	17,344	19%	
Logic Distribution				
Number of occupied Slices	2,516	8,672	29%	
Number of Slices containing only related logic	2,516	2,516	100%	
Number of Slices containing unrelated logic	0	2,516	0%	
Total Number of 4 input LUTs	3,603	17,344	20%	
Number used as logic	3,413			
Number used as a route-thru	122			
Number used for Dual Port RAMs	16			
Number used for 32x1 RAMs	52			
Number of bonded IOBs	169	304	55%	
IOB Flip Flops	3			
Number of Block RAMs	14	28	50%	
Number of GCLKs	5	24	20%	
Total equivalent gate count for design	967,668			
Additional JTAG gate count for IOBs	8,112			

圖16. uTCP with OC GMAC in

Spartan3E-1200

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	906	17,344	5%	
Number of 4 input LUTs	1,865	17,344	10%	
Logic Distribution				
Number of occupied Slices	1,232	8,672	14%	
Number of Slices containing only related logic	1,232	1,232	100%	
Number of Slices containing unrelated logic	0	1,232	0%	
Total Number of 4 input LUTs	1,970	17,344	11%	
Number used as logic	1,865			
Number used as a route-thru	37			
Number used for Dual Port RAMs	16			
Number used for 32x1 RAMs	52			
Number of bonded IOBs	150	304	51%	
IOB Flip Flops	1			
Number of Block RAMs	11	28	39%	
Number of GCLKs	1	24	4%	
Total equivalent gate count for design	749,992			
Additional JTAG gate count for IOBs	7,584			

圖17. uTCP without OC GMAC in Spartan3E-1200

Device Utilization Summary				
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Registers	2,193	28,800	7%	
Number used as Flip Flops	2,177			
Number used as Latches	16			
Number of Slice LUTs	2,408	28,800	8%	
Number used as logic	2,352	28,800	8%	
Number using O6 output only	1,834			
Number using O5 output only	53			
Number using O5 and O6	565			
Number used as Memory	56	7,680	1%	
Number used as Dual Port RAM	0			
Number using O5 and O6	0			
Number used as Single Port RAM	10			
Number using O6 output only	10			
Number used as exclusive route-thru	10			
Number of route-thrus	77	57,600	1%	
Number using O6 output only	63			
Number using O5 output only	14			
Slice Logic Distribution				
Number of occupied Slices	1,250	7,200	17%	
Number of LUT Flip Flop pairs used	3,304			
Number with an unused Flip Flop	1,191	3,304	35%	
Number with an unused LUT	896	3,304	26%	
Number of fully used LUT-FF pairs	1,297	3,304	38%	
Number of unique control sets	200			
IO Utilization				
Number of bonded IOBs	169	480	35%	
IOB Flip Flops	3			
Specific Feature Utilization				
Number of BlockRAM/FIFO	12	60	20%	
Number using BlockRAM only	12			
Total primitives used				
Number of 36k BlockRAM used	0			
Number of 18k BlockRAM used	0			
Total Memory used (KB)	396	3,160	12%	
Number of BUFG/BUFGCTRLs	5	32	15%	
Number used as BUFGs	3			
Number used as BUFGCTRLs	3			
Total equivalent gate count for design	1,407,322			
Additional JTAG gate count for IOBs	8,112			

圖18. uTCP with OC GMAC in Viretex5-LXT50

Device Utilization Summary				
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Registers	907	28,800	3%	
Number used as Flip Flops	907			
Number of Slice LUTs	1,394	28,800	4%	
Number used as logic	1,364	28,800	4%	
Number using O6 output only	1,057			
Number using O5 output only	14			
Number using O5 and O6	293			
Number used as Memory	26	7,680	1%	
Number used as Dual Port RAM	0			
Number using O5 and O6	0			
Number used as Single Port RAM	10			
Number using O6 output only	10			
Number used as exclusive route-thru	4			
Number of route-thrus	20	57,600	1%	
Number using O6 output only	10			
Number using O5 output only	2			
Slice Logic Distribution				
Number of occupied Slices	500	7,200	6%	
Number of LUT Flip Flop pairs used	1,644			
Number with an unused Flip Flop	737	1,644	44%	
Number with an unused LUT	250	1,644	15%	
Number of fully used LUT-FF pairs	657	1,644	39%	
Number of unique control sets	115			
IO Utilization				
Number of bonded IOBs	150	480	32%	
IOB Flip Flops	1			
Specific Feature Utilization				
Number of BlockRAM/FIFO	10	60	16%	
Number using BlockRAM only	10			
Total primitives used				
Number of 36k BlockRAM used	0			
Number of 18k BlockRAM used	3			
Total Memory used (KB)	342	3,160	15%	
Number of BUFG/BUFGCTRLs	1	32	3%	
Number used as BUFGs	1			
Total equivalent gate count for design	1,269,926			
Additional JTAG gate count for IOBs	7,584			

圖19. uTCP with OC GMAC in Viretex5-LXT50

4.4 實驗結果

測試環境中，以時脈100MHz為頻率單位。並撰寫一組測試程式，利用uTCP與TCP完成三項交握後，以全速向TCP echo server端發送資料，並藉由ModelSim觀察波形得出，計算出測試時間內能送出多少封包。如圖20、圖21、圖22、圖23分別為三相交握以及傳送封包部份波形。

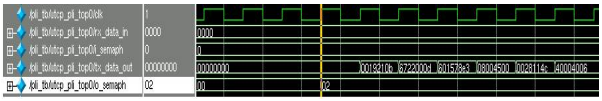

圖20. uTCP發起傳送要求（SYN-SEND STATE）

圖21. TCP echo server回傳一包SYN+ACK封包

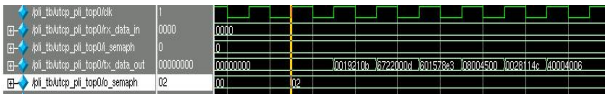

圖22. uTCP回傳ACK封包（ESTABLISHED）

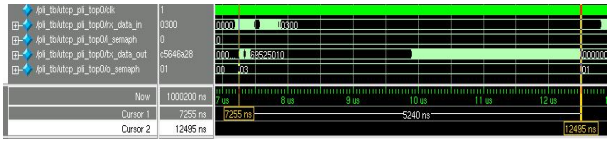

圖23. uTCP送出DATA封包

測試結果，當uTCP以全速送出資料時，uTCP的資料吐出量達到約1.3Gbps的速率，此數據與預測時的速度（1.6Gbps）[2]相差不遠。由此證明，uTCP是一顆具備低成本且高效能之TCP網路晶片。圖24為TCP echo server 端之Wireshark所擷取之封包，前三包封包為三相交握之封包，接續以uTCP（140.127.182.154）為傳送端，以及TCP echo server（140.127.182.65）互相進行連線驗證，以及封包傳送之測試，予以證明其連線正確性。

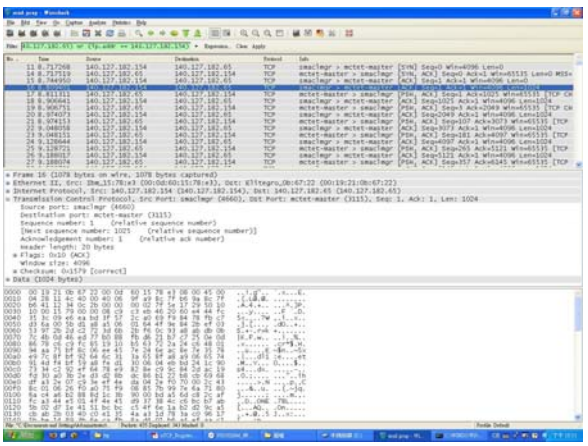

圖24. Wireshark擷取封包片段

5.結論

針對高畫質影像感測網路而言，本研究提出一顆微型高速網路晶片設計，具備傳送高畫質影像以及獨立執行 TCP/IP 網路協定之能力，並將其設計成適合應用於 FPGA 之矽智財。本研究中，提出 SEQ 定址法，在協定方面，本研究利用最少的硬體資源即可有效解決重傳、重送、封包定序、slide window 等問題；而在軟硬體設計方面，取消 memory mapping 記憶體空間之使用、降低記憶體控制器設計複雜度、減少設計 PicoBlaze 程式時之複雜度。此外，uTCP 提供使用者利用 DMA 的功能，一次傳送或接收大筆資料，降低資料搬移對 bus 所造成之傷害，以提昇 bus 傳輸效率。uTCP 之設計，在 VIRTEX-5 LX50T 開發版上，合成結果僅利用 4%之 FPGA 面積，且運行於 100MHz 的時脈即可達到 1.3Gbps 的資料吐出量，以符合高畫質影像感測網路之需求，並且在不需作業系統支援即可完成 TCP/IP 封包之傳遞，使其成為高效能、低成本且獨立應用於 FPGA 上之 TCP/IP 網路矽智財。

6.參考文獻

[1] 林玉賢 陳健維 蘇益慶, "μTCP：高速 TCP/IP 協定堆疊之軟硬體協同設計," *2007 開放原始碼技術與應用研討會*, pp.114, 台灣, 2007.11

[2] 蘇益慶 林玉賢, 陳健維, "微型高速 TCP/IP 協定堆疊之軟硬體協同設計," *TANET2008 台灣網際網路研討會*, pp.99, 台灣, 2008.10

[3] 江欣潔 戴元邦 陸志豪 王垂愈 陳世傑, "TCP/IP Offload Engine 技術與現況," *CCL TECHNICAL JOURNAL*, pp.91-97, TAIWAN, 2003

[4] Eric Yeh, Herman Chao, Venu Mannem, Joe Gervais, and Bradley Booth, "Introduction to TCP/IP Offload Engine (TOE)," *10 GIGABIT ETHERNET ALLIANCE*, Version1.0, April 2002

[5] Adescom, Inc. "IPAC-TCP/IP Protocol in Hardware," *http://www.adescom.com*

[6] Zhan Bokai, and Yu Chengye, "TCP/IP Offload Engine (TOE) for an SOC System," *Nios II Embedded Processor Design Contest—Outstanding Designs 2005*, pp.306-322, 2005

[7] Ken Chapman's, "PicoBlaze 8-bit Embedded Microcontroller User Guide," *www.xilinx.com*, ug129, Nov. 2005

[8] www.opencores.com

[9] Zhong-Zhen Wu, and Han-Chiang Chen, "Design and Implementation of TCP/IP Offload Engine System over Gigabit Ethernet," *Computer Communications and Networks, 2006. ICCCN 2006. Proceedings.15th International Conference*, 9-11 Oct. 2006, pp.245 - 250

[10] Hankook Jang, Sang-Hwa Chung, and Dae-Hyun Yoo, "Implementation of an Efficient RDMA Mechanism Tightly Coupled with a TCP/IP Offload Engine," *Industrial Embedded Systems, 2008. SIES 2008. International Symposium*, 11-13 June 2008 pp.82 – 88

[11] Wen-Fong Wang, Jun-Yau Wang, and Jin-Jie Li, "Study on Enhanced Strategies for TCP/IP Offload Engines," *Parallel and Distributed Systems, 2005. Proceedings. 11th International Conference*, Volume 1, 20-22 July 2005 pp.398 - 404 Vol. 1

[12] Dae Won Kim, Won Ok Kwon, Kyoung Park, and Seong Woon Kim, "Internet Protocol

Engine in TCP/IP Offloading Engine,"
Advanced Communication Technology, 2008. ICACT 2008. 10th International Conference,
Volume 1, 17-20 Feb. 2008 pp.270 - 275