以 FPGA 完成影像壓縮晶片之設計

林灶生

國立勤益技術學院 電子工程系 副教授

摘 要

本研究旨在以場規劃閘陣列(Field Programmable Gate Array，FPGA)設計一影像壓縮晶片系統。在VLSI晶片設計技術領域中，FPGA屬於可程式邏輯元件(Programmable Logic Device，PLD)的一種，其最適合量少之特殊產品的設計，可快速轉換成半訂製IC製程(Semi-Custom)以降低成本。尤其在開發半訂製IC時，以其當作硬體功能測試之雛型，則可縮短設計時間及降低產品成本。由於FPGA以SRAM爲其內部之功能設定元件，具有無限多次使用之優點，在個人電腦之工作環境下即可完成設計。本研究即著眼於FPGA此項優點，配合開發系統所提供之硬體描述語言(Very high speed integrated circuit Description Language，VHDL)界面，設計一以方塊截短編碼(Block Truncation Coding)爲主之影像壓縮晶片系統。

一、簡 介

資料壓縮是一個用來減少表示訊息所需要之訊號空間量的方法。而訊號空間指的可能是實際上的空間(例如記憶體的大小)；或一段時間(例如傳送訊息所需的時間)；或者是傳送該訊息所需之頻寬。雖然記憶體以及傳輸線的技術在近年來都有很大的進步，但儲存在電腦內以及傳輸於電腦間的大量資料卻以更快的速度成長，以致於資料壓縮技術在資料儲存與傳輸上的需求與日俱增，而無減少的跡象。舉例來說，一片光碟可以儲存 600 百萬個以上的英文字母，足夠儲存一部相當完整的百科全書。但是即使是像光碟這麼大的記憶容量，在許多新興的應用上仍嫌不足。一個很典型的應用便是學術性論文的儲存。目前，國內許多國立大學的圖書館都允許使用者透過網路搜尋論文，而這些論文便是存放在光碟上。但是單單 IEEE 及 IEE 近十年的期刊與會議論文便佔了三百多片光碟！而且每月增加中。資料壓縮不但可以減少所需要的光碟數目，也可以降低一篇論文在網路傳輸所需要的時間。

目前的影像壓縮演算法，就靜態影像而言，大部份乃是以軟體加以實現。然陸續有研究者紛紛以硬體實現其快速壓縮之目的以應用於及時系統之上。如 JPEG 晶片即是一靜態影像壓縮晶片。本研究亦希望以現有之方塊截短編碼及動量絕對值方塊截短編碼(Absolute Moment BTC)演算法 [1] 設計出實用之 VLSI 晶片。

二、積體電路設計

積體電路設計包括不同之技術，每一種技術均有其優劣，本節就各種技術分述如下：

2.1 全訂製 IC 設計(Full-Custom)

全訂製 IC 設計從基本的邏輯閘至 Flip-Flops 與 Buffer，…等元件均須由設計者自己建立，且設計者須自行佈線與邏輯設計，當完成雛型時，設計者須自行測試，產品之特性在於元件具有高效能，大量生產則價格低，其特點爲：

- 設計週期長（最快 3～6 個月）
- 開發費用高
- 生產週期約一個月以上

2.2 標準元件設計(Standard Cell)

設計製造工廠提供標準零件如邏輯閘、正反器、計數器、解碼器，及其他巨集元件供設計者使用，設計者須進行邏輯設計與雛型測試。其特點在於：

- 開發與製造費用較全訂製 IC 低。
- 效能亦較全訂製 IC 低
- 開發週期較全訂製 IC 短（約 3～4 週）

2.3 閘陣列(Gate Array)

設計製造工廠僅擁有基本邏輯閘(不含巨集元件)，佈線過程較標準元件 IC 複雜，故效能低，佈線面積太大，造成價格昂貴，惟僅須留 2～3 層金屬層佈線，故製程較短，約一週即可生產完成。

2.4 可程式邏輯元件(Programmable Logic Device，PLD)

可程式邏輯元件包括了簡易型 PLD(Simple PLD)、複雜 PLD(Complex PLD)及場規劃邏輯陣列(Field Programmable Gate Array, FPGA) [2]。其中的 SPLD 屬於可程式邏輯陣列(Programmable Array of Logic, PAL)，CPLA 採用 EPROM 及 EEPROM 作爲其內部之程式規劃儲存元件；而 FPGA 內部則是以 SRAM 爲架構。基本上，PLD 製造工廠已生產可程式規劃之 IC，每一不同型號之 IC，內部架構已固定，其參數與設計軟體完全相符，其特點在於：

- 開發環境便宜
- 在一般個人電腦環境下即完成設計
- 生產樣品在實驗室即可完成，適合中小企業或個人研究室
- 產品單價高
- 效能較其他設計方法產品差。

PLD 最適合量少之特殊產品的設計，可快速轉換成半訂製 IC 製程(Semi-Custom)以降低成本，尤其在開發半訂製 IC 時，以其當作硬體功能測試之雛型，則可縮短設計時間及降低產品成本。目前已有相關公司以 FPGA 開發出硬體功能測試器(Hardware Function Tester)。

三、方塊截短編碼法

方塊截短編碼 (Block truncation coding，BTC) [1-3]主要應用於影像壓縮設計。近年來雖然有人試著將 BTC 用在一維訊號上，卻不如用在二維訊號來得自然。在一個 BTC 系統中，影像首先被分割成許多不重疊的 $n\times n$ （一般是 4×4 ）的方塊，每一個方塊上面的像素（$n\times n$ 個) 原先可能有各種值，但是在經過 BTC 編碼後，只能有兩個值。對於任何一個方塊，決定於方塊上像素值的統計情況，我們以一個位元圖 (Bit Map，也是 $n\times n$) 及兩個重建階(Reconstruction Levels)，a 和 b，來表示該方塊。位元圖及重建階的計算程序是先計算原方塊裡所有像素的量化臨界值 (通常取爲像素值之平均)，原來之像素值若大於此臨界值，則位元圖上該位置的位元設爲 1；否則設爲 0。所有位元圖上 1 的位置，重建之像素值都爲 b；所有位元圖上 0 的位置，重建之像素值都是 a。而 a，b 的選擇一般則以保留原方塊的統計特性使得重建方塊也具此特性爲標準。

最早的 BTC 壓縮法，其量化器之設計便以保留方塊之平均值及方差爲準。影像首先被切割成不重疊的 $n\times n$ 的方塊。令 m = $n\times n$ 且 x_1，x_2 ……，x_m 爲原影像中一個方塊的像素值。希望量化器能保留住的是第一階與第二階之動量：

$$\overline{X} = \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\overline{X^2} = \frac{1}{m}\sum_{i=1}^{m} x_i^2$$

方差 (Variance)值則爲：

$$\sigma^2 = \overline{X^2} - \overline{X}^2$$

因此只要 $\overline{x}$ 與 $\overline{x^2}$ 能保持不變，方差也就能保持不變。現在需要找出臨界值 x_{th} 及兩個重建階 a 與 b 使其滿足：

$$\hat{X}i = \begin{cases} a & , \ if\ x_i < x_{th} \\ b & , \ if\ x_i \geq x_{th} \end{cases}$$

其中 i = 1,2,3.....,m

由於希望能保留住前兩階的動量，需要兩個自由度，而這兩個自由度可以由重建階 a 與 b 來提供，多出來的一個自由度使留給臨界值而直接將其設爲 $\overline{x}$ 以簡化分析。令 q 表示方塊中 x_i 值大於等於臨界值的數目，要保留住 $\overline{x}$ 與 $\overline{x^2}$ ，必須滿足

$$m\overline{X} = (m-q)\,a + qb$$

$$m\overline{X^2} = (m-q)a^2 + qb^2$$

解 a 與 b 得到

$$a = \overline{X} - \sigma\sqrt{\frac{q}{m-q}} \qquad b = \overline{X} + \sigma\sqrt{\frac{m-q}{q}}$$

送出 $n\times n$ 的位元圖以標明那些像素值該重建爲 a，那些則該重建爲 b，同時也送出足以計算出 a 與 b 值的資訊給接收端。可以直接送出 a、b，也可以送出 $\overline{x}$ 和 σ 而留給接收端去計算出 a、b 值 (q 值則從位元圖中計算可得)。另一種可以避免掉平方及開根號的運算，但是卻可以得到類

似的效果的方法，稱之爲動量絕對值 BTC (Absolute Moment BTC，簡稱 AMBTC)。AMBTC 的量化器被設計成保留住動量的絕對值不變。其平均值與第一絕對主要動量之定義分別如下：

$$X = \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\alpha = \frac{1}{m}\sum_{i=1}^{m} |x_i - \bar{x}|$$

設定量化器的臨界值爲 $\bar{x}$，能保持 $\bar{x}$ 與 α 不變的重建階爲

$$a = \bar{X} - \frac{m\alpha}{2(m-q)} \qquad b = \bar{X} + \frac{m\alpha}{2q}$$

其中 q 爲方塊中像素值 $x_i > \bar{X}$ 的數目。和前面一樣，可以直接將 a 和 b 編碼送出，也可以聯合量化 $\bar{X}$ 和 α 以降低位元率。

四、BTC 晶片設計

在本研究中使用 XILINX XC4000EPC84 晶片[4, 5]來設計 BTC 壓縮晶片，此元件約有 3200 Gate Count 且工作於 3.0V～3.6V 之間，屬於低電源消耗之元件。XC4000 系列 FPGA 結合結構化能力，晶片上可選擇邊緣觸發及雙埠模式之 RAM 記憶體、快速能力、豐富的佈線資源及最新且複雜之軟體，可完成完全自動化之高密度與高效能之複雜設計。本系列元件效能可達 66MHz，本身擁有變化多端之陣列結構及回讀能力與舊有 XC4000 相容，屬於第三代之 FPGA。其擁有之特點包括：可選擇之 RAM 記憶體，晶片上同步寫入選項或雙埠 RAM 選項之超快速 RAM;完全符合 PCI 界面;相當多的正反器;極具變通之函數產生器;提供高速進位邏輯;存在於每一邊界之多邊緣解碼器;聯結線之階層屬性結構;內建三態匯流排能力;8 個共用低曲率(Skew Rate)之時脈信號。

在 BTC 晶片中，接收已分割之不重疊的方塊，以順序式一次處理 4×4 像素，並送出二個重建階及位元圖。系統方塊如(圖一)所示，內部詳細之計算流程則如(圖二)所示。

五、以 VHDL 完成 BTC 的設計

VHDL 電路設計語言提供一個用來描述電路特性與行爲的工業標準語言[6]，從摘要式的電路描述到具體細節的電路描述風格，已廣被各種電腦輔助的電子電路設計工具所採用，包含有電路模擬工具，電路合成工具，以及電路佈局工具等。結構性描述的 VHDL 電路架構，主要透過元件的宣告（Component Declaration）與元件的叫用等方式，構成電路中各元件間的連線關係。由於此方式的描述風格，相當於在描述電路中其各元件間的相互連線關係，故又可稱爲連線關係描述風格(Netlist Description Style)。資料流型式的 VHDL 電路架構描述風格，主要是使用 VHDL 語言中所內建的標準布林運算函式爲主軸，將各訊號間的布林代數關係，以布林方程式來表示。而資料流的描述敘述所構成的電路關係，在 VHDL 語言中是具有共時性（Concurrent）的，故又稱其爲共時性敘述（Concurrent Statement）。在 VHDL 中，較複雜的電路部份，通常使用電路特性的行爲性描述來設計，如此透過電腦的合成與最佳化，可以加快產品的設計週期，並使設計之

電路，非常容易維護。在 VHDL 語言中的行爲性描述電路設計，通常是使用 Process 的方式達成。由於用 Process 所寫成的 VHDL 電路設計程式碼，具有優先秩序的關係，所以又稱爲循序性敘述（Sequential Statements）。

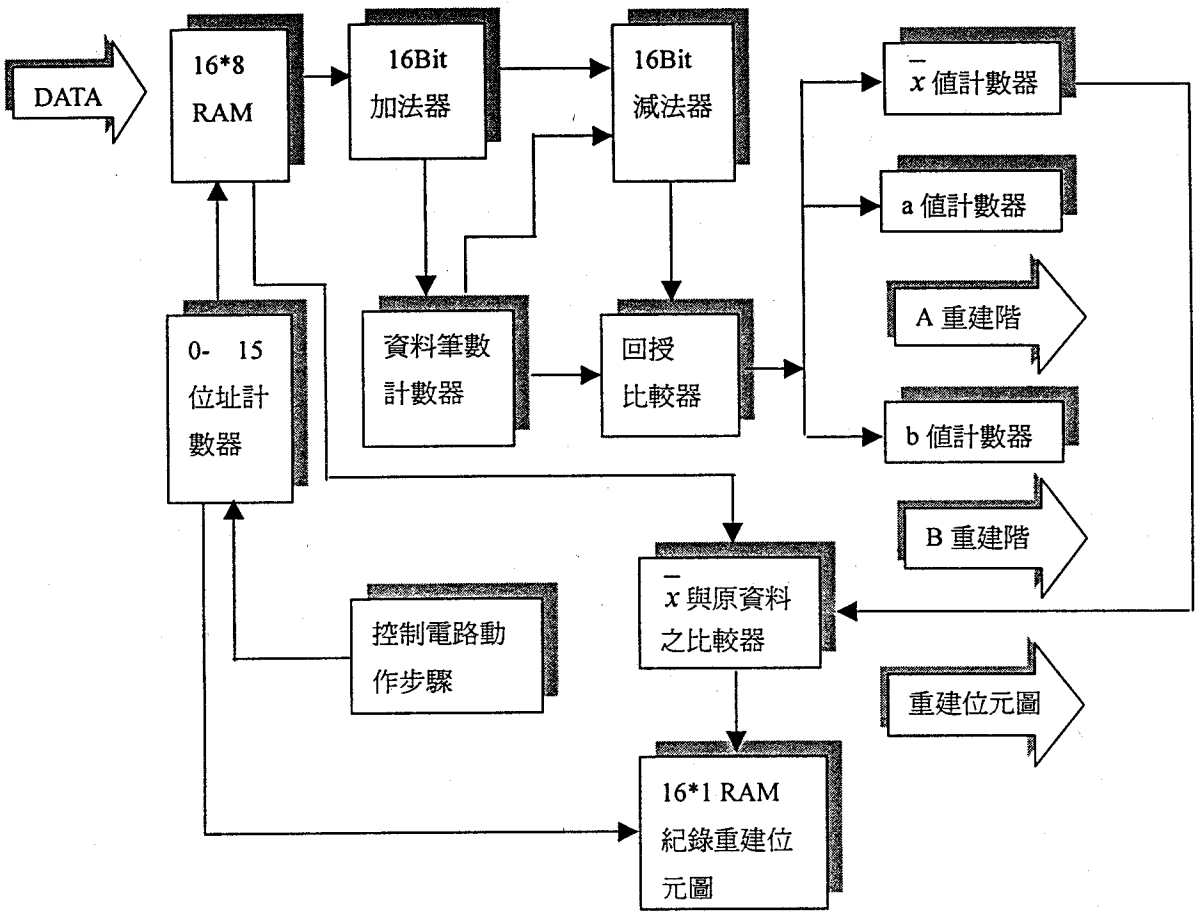
針對第三節與第四節所描述之原理與流程，我們完成了 BTC 與 AMBTC 之晶片，(圖三)僅列出 AMBTC 之 VHDL 語言描述。(圖四) 至(圖七)則爲以一組 4 x 4 資料測試該晶片所得之時序圖。時序圖中顯示出單組 16 筆資料測試所得結果均正確。

六、實驗結果與討論

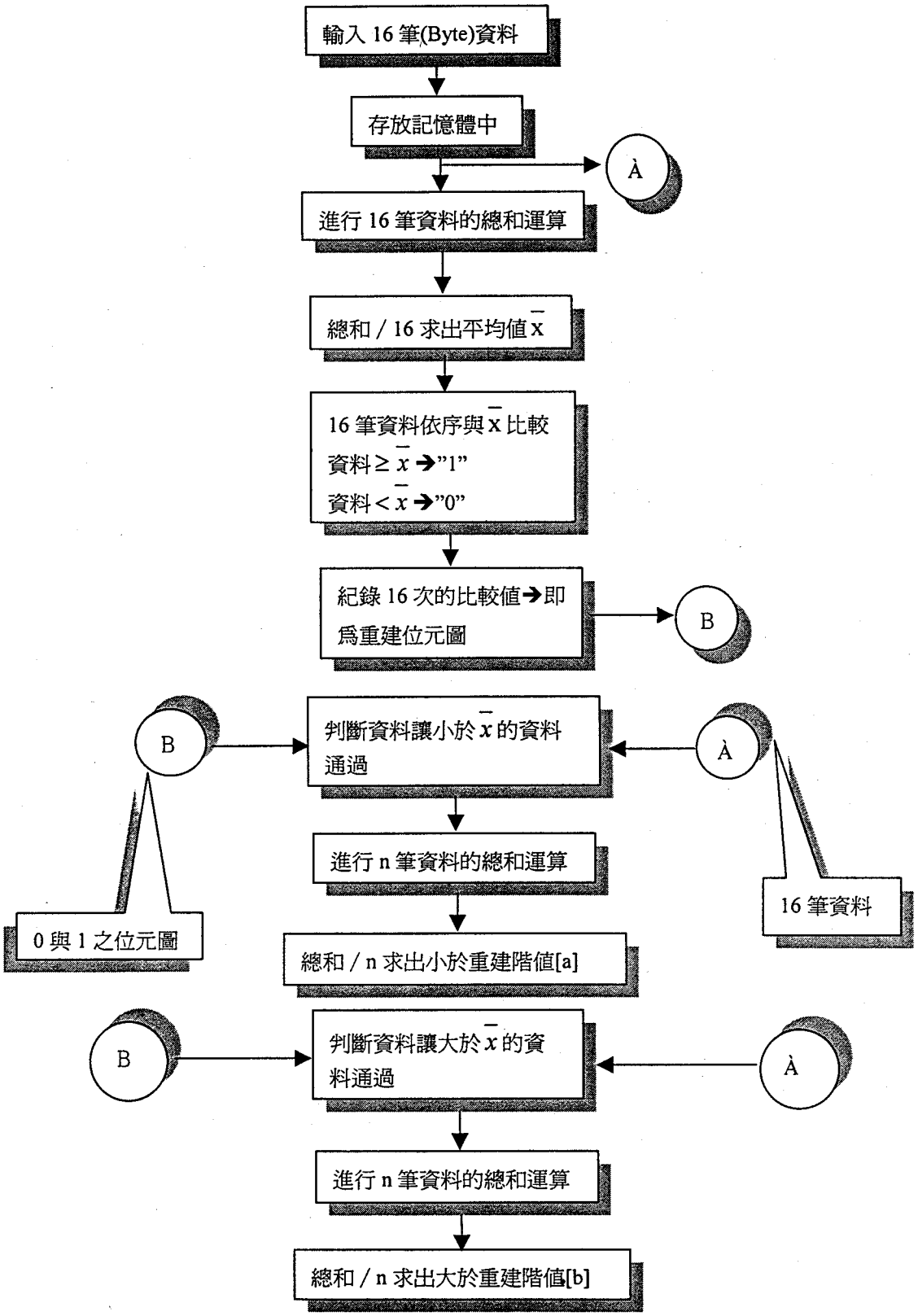
爲了完整的測試 BTC 之可用性，我們以一張如(圖八.a)所示 F-16 影像透過輸入界面饋入 IC 晶片中，實際完成壓縮之動作。(圖八.b)所示爲經由 BTC 壓縮晶片壓縮之後重建所得結果，其重建品質(PSNR)與利用軟體演算法所得重建品質完全相同，由此證實 BTC 壓縮晶片之可用性。

當初指導同學專題製作時選用 BTC 當影像壓縮技術，是由於 BTC 之演算法較容易推演，而且曾被考慮作爲影像壓縮之標準。在這次專題製作上，當同學發生瓶頸時，總會尋求解決問題的方法，在設計過程之初，互相討論電路設計的初步構思，作爲往後設計的基礎架構，在製作過程中產生問題，同學就會提出所遇之難處進而互相研究，共同找出問題的癥結；從這樣一來一往當中使同學更增添不少解決問題之方法，同時也增加了師生間良好的互動，讓研究討論及溝通資訊交流的管道更加暢通。

由於 BTC 壓縮晶片使用了 42 隻接腳，使得此晶片之界面過於複雜，未來可將其界面接腳簡化成 18 隻，即將重建階與位元圖共用一個 Byte 分四次輸出。再者，如果允許 BTC 增加一些複雜度，可以在演算法中增加一些判斷，例如判斷出影像的某個區域屬於高頻率區，則選用較小的方塊以利於精確地重建；反之，在低頻率區則可以選用較大的方塊。藉由允許方塊大小依頻率而改變，重建影像的品質亦可以被改善，亦即在所有像素都一樣或變化很小的情況，可以用平均值代表整個方塊。在高頻率的區域則需要將大的方塊再切成較小的方塊，每個小方塊分別使用 AMBTC 的量化器編碼，送出每個小方塊的重建階與位元圖。這些技術之發展與新的影像壓縮演算法之開發均是我們未來製作晶片努力之方向。



(圖一) BTC 影像壓縮晶片系統方塊圖



(圖二)BTC 晶片內部資料計算之流程

```
ENTITY COMPRESS IS
   PORT ( DATACK , RESET : IN BIT
               DATAIN          :  IN BIT_VECTOR (7 DOWNTO 0));
               DATAOUT         : OUT BIT_VECTOR (31 DOWNTO 0));
END COMPRESS ;

ARCHITECTURE BEH OF COMPRESS IS
TYPE MEMO_VECTOR IS ARRAY (INTEGER RANGE <> , INTEGER
                              RANGE<>) OF INTEGER ;
   SIGNAL   BUFF : MEMO_VECTOR (3 DOWNTO 0 , 3 DOWNTO 0);
   SIGNAL   XCLOCK , YCLOCK : INTEGER ;

BEGIN
   PROCESS (DATACK , RESET)
   VARIABLE   A , LABEL_A     : INTEGER ;
   VARIABLE   C , I,LABEL_B   : INTEGER ;
   VARIABLE   B , MEAN , X    : INTEGER ;

   BEGIN
   IF RESET'EVENT AND RESET= '1' THEN   --當 RESET 時，全部設爲 0
      XCLOCK<=0 ;
      YCLOCK<=0 ;
      FOR A IN 0 TO 3 LOOP
         FOR B IN 0 TO 3 LOOP
            BUFF (A , B) <= 0 ;              --BUFFER 清除爲 0
         END LOOP ;
      END LOOP ;
   END IF ;
   IF DATACK'EVENT AND DATACK= '1' THEN   --當信號確認時，送入資料
      IF YCLOCK < 4 THEN                  --輸入資料
         B : =0 ;
         FOR I IN 0 TO 7 LOOP
              IF DATAIN (7 - I) = '1' THEN
                 B : =B+1 ;                    --計算資料值存放於 B
              END IF ;
              IF I < 7 THEN
               B : =B * 2 ;
             END IF ;
```

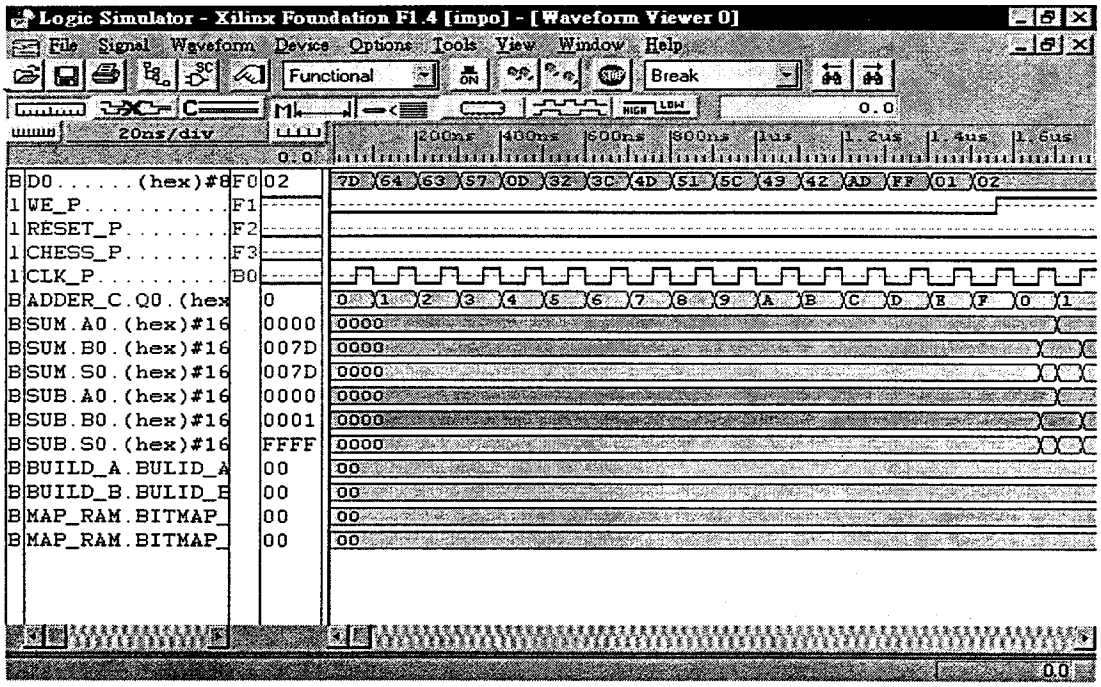
(圖三) AMBTC 之 VHDL 語言描述

```
    END LOOP；
    BUFF（YCLOCK，XCLOCK）<=B； --將資料值存入 BUFFER
    IF XCLOCK=3 THEN                --檢驗是否該換行
        XCLOCK<=0；
        YCLOCK<=YCLOCK+1；          --換列
    ELSE
        XCLOCK<=XCLOCK+1；          --下一位置
    END IF;
  ELSE
    B：=0；
    FOR A IN 0 TO 3 LOOP
        FOR C IN 0 TO 3 LOOP
          B：=B+BUFF（A，C）；      --16 筆像素總值
        END LOOP；
    END LOOP；
    MEAN：=B/16；                   --MEAN 爲總值之平均值
    LABEL_A：=0；
    LABEL_B：=0；
    X：=0；
    B：=0；
    FOR C IN 0 TO 3 LOOP
      FOR A IN 0 TO 3 LOOP
        X：=ABS（BUFF（C，A）-MEAN）；
        X：=X+X；
        IF BUFF（C，A）>MEAN THEN
            LABEL_A：=LABEL_A+BUFF（C，A）；--大於平均值之總和
            B：=B+1；               --計算有幾筆像素大於總平均值
        ELSE
          LABEL_B：=LABEL_B+BUFF（C，A）；--小於平均值之總和
        END IF；
      END LOOP；
    END LOOP；
    X：=X/16；
    IF （LABEL_A>0）AND（b>0）THEN
      LABEL_A：=LABEL_A/B；         --計算大於平均值之重建階
    ELSIF LABEL_B>0 THEN
      LABEL_B：=LABEL_B/（16-B）；--計算小於平均值之重建階
    END IF;
    FOR C IN 0 TO 3 LOOP
      FOR A IN 0 TO 3 LOOP
```

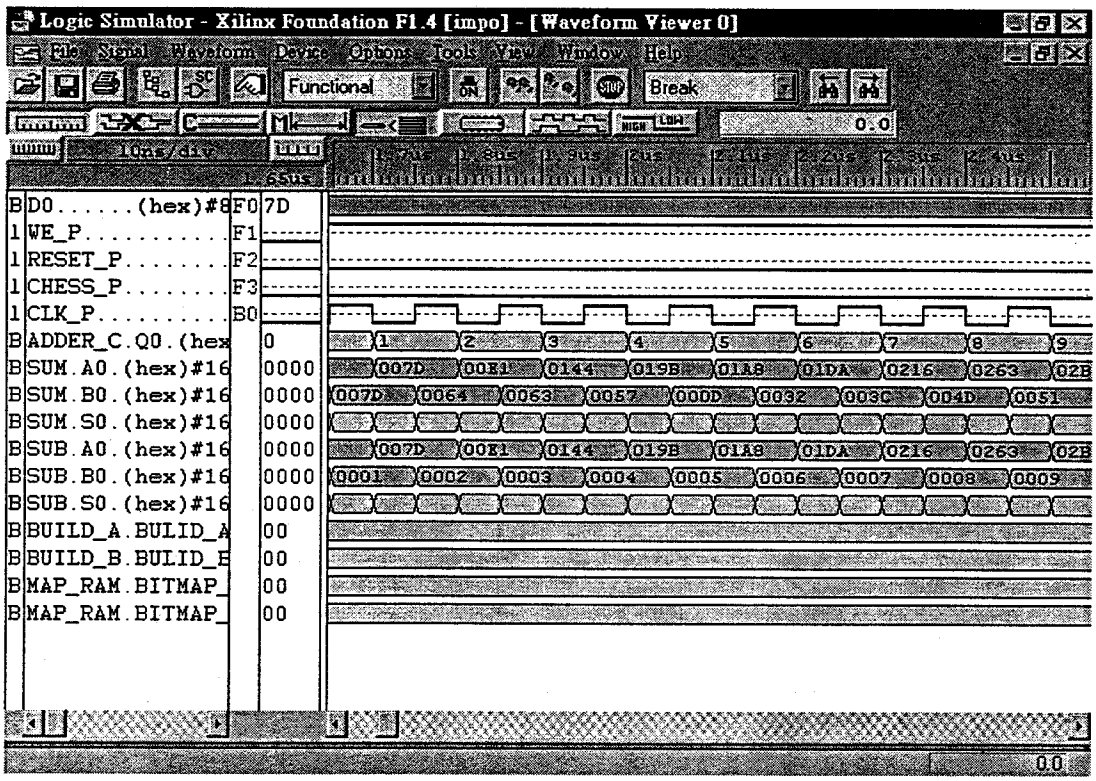
(圖三) AMBTC 之 VHDL 語言描述

```
            IF BUFF（C，A）>MEAN THEN --大於平均值者，輸出爲"1"
                DATAOUT（16+（（C*4）+A））<='1'；
            ELSE                              --小於平均值者，輸出爲"0"
                DATAOUT（16+（（C*4）+A））<='0'；
              END IF；
          END LOOP；
        END LOOP；
        B：=（MEAN*256）+X；
        FOR C IN 0 TO 15 LOOP
          IF（B MOD 2）=1 THEN          --將 10 進制轉換成 2 進制
            DATAOUT（C）<='1'；
          ELSE
            DATAOUT（C）<='0'；
          END IF；
          B：=B/2；
        END LOOP；
        XCLOCK<=0；                       --清除準備下一次資料輸入
        YCLOCK<=0；
      END IF；
    END IF；
  END PROCESS；
END BEH；
```

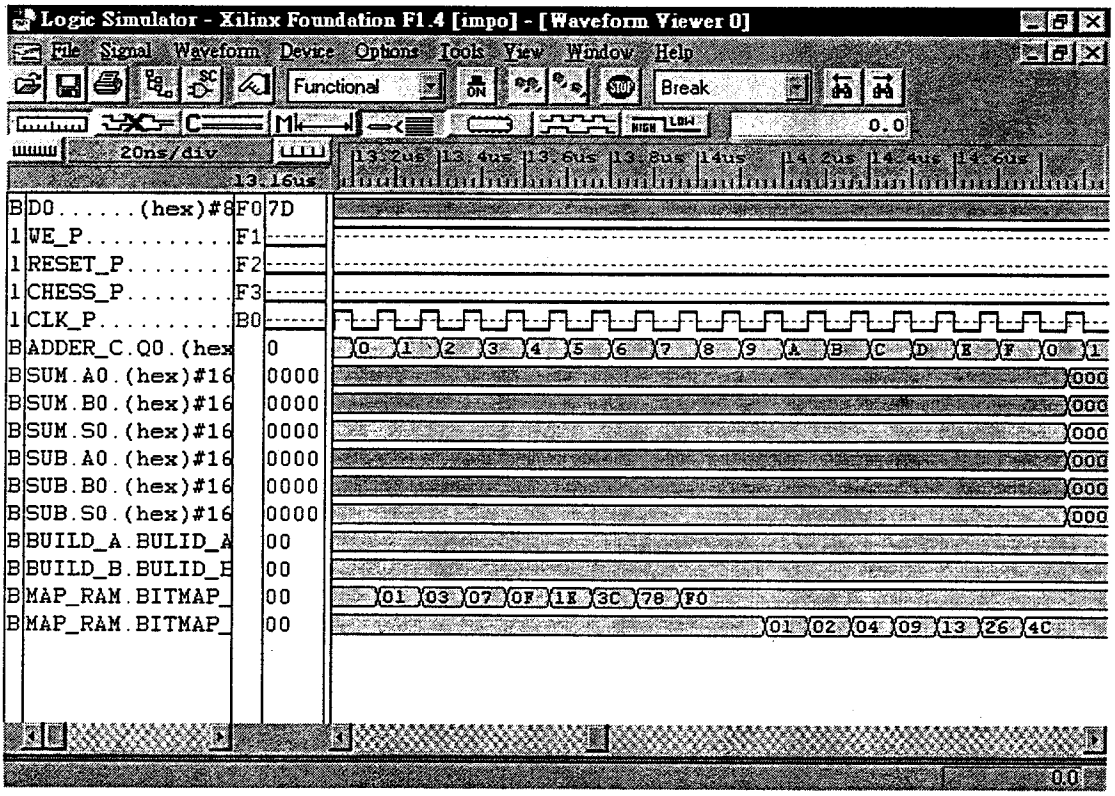
(圖三) AMBTC 之 VHDL 語言描述



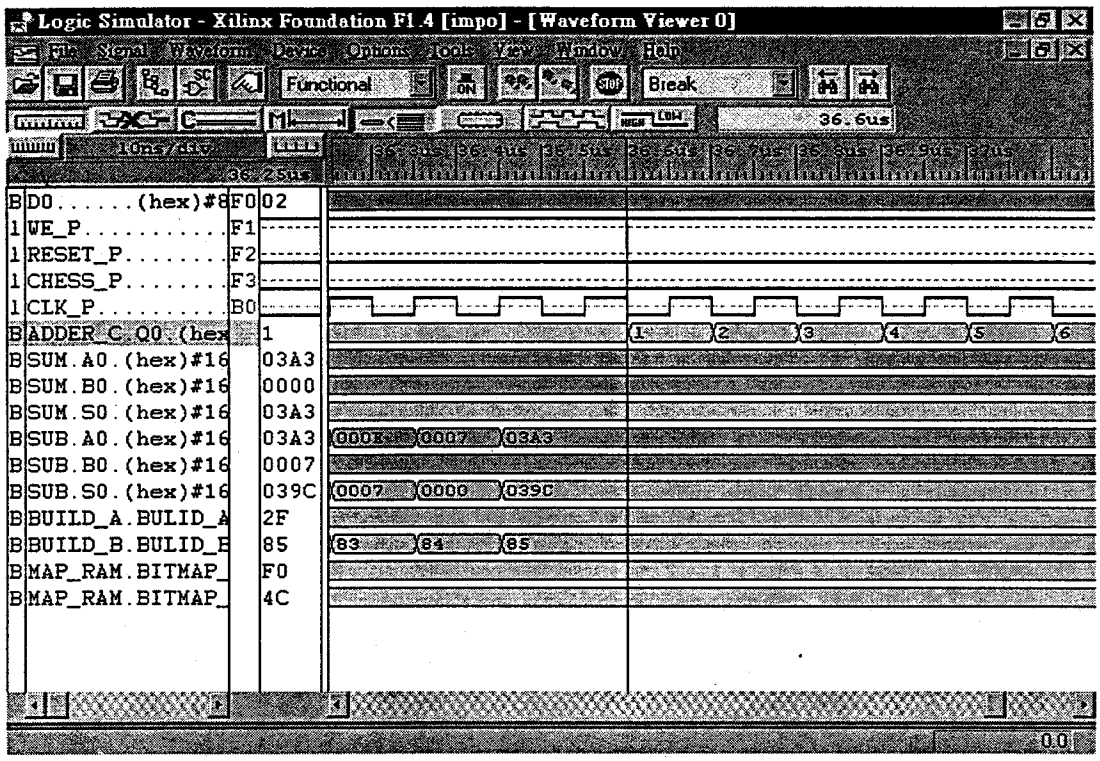
(圖四) 將 16 筆資料存入晶片之時序



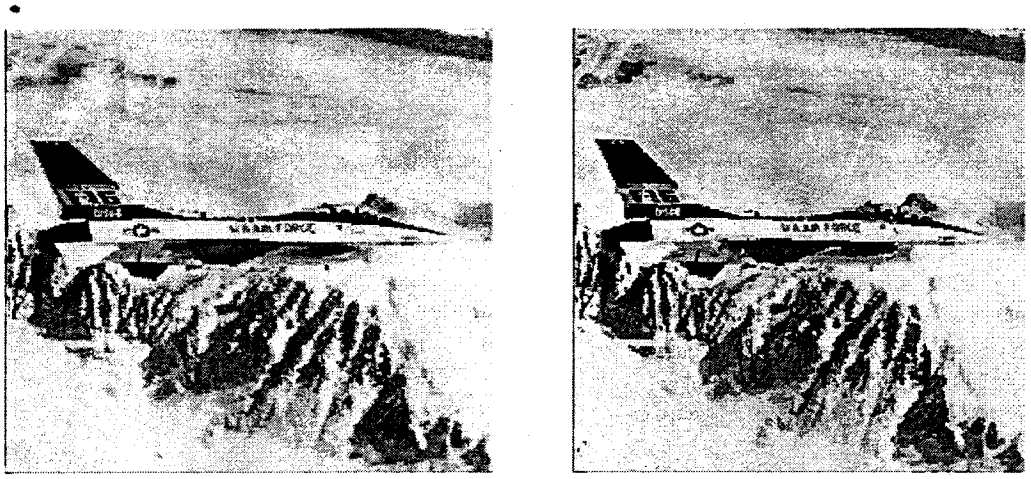
(圖五) 加法運算時序



(圖六) 位元圖資料存入 RAM 內部時序



(圖七) 運算後得到的重建階與位元圖 4 種輸出資料之時序



(a) (b)

(圖八) BTC 影像壓縮晶片測試結果: (a)原始影像"F-16"; (b)重建影像

參考文獻

[1] 戴顯權, 1996, 資料壓縮, 松崗圖書, 台北.

[2] E. J. Delp, O. R. Mitchell, 1979, "Image compression using block truncation coding," *IEEE Trans. Commun.* Vol. COM-27, pp. 1335-1342.

[3] C. W. Chao, C. H. Hsieh, and P. C. Lu, 1998, "Image compression using modified block truncation coding algorithm," *Signal Procrssing,* vol. 12, pp. 1-11.

[4] XILINX, 1998, "*The programmable logic data book*".

[5] XILINX, 1998, "*The practical XILINX designer lab book*", Prentice Hall.

[6] Z. Navabi, 1993, "VHDL analysis modeling of digital systems," McGraw-Hill.