第五章 、 硬體實現

圖38是本文在硬體實踐上所使用的方塊圖,因為是以一維的過完備小波轉換(overcomplete wavelete transform,OCWT)為基礎,所以其硬體流程也會以一維處理為基礎,在ARM的架構之下,應用ARM的AHB來對外部的DRAM做存取的動作,而在我們的系統之中,由於整個系統會是由內部的DSP來控制deblocking硬體,所以在我們的系統中,啟動的程序會是由USER下命令給ARM,再由ARM下命令給DSP,最後DSP會啟動整個deblocking硬體.原因是當我們的系統在另一套硬體應用時,如MPEG4 encoder時,DSP可以用來做Motion estimation 的rate control,還有在做AV整合時,DSP也可以用來做Audio的編解碼.所以我們的硬體是由DSP來做啟動的動作,而非直接由ARM來做控制.另外還有跟AHB介接的一些元件沒有秀在圖38,但這不影響我們的基本功能,在我們的設計中,會先處理每一列(row),再處理每一行(column).而我們的目標有2個:

(1)在D1(720x482)的大小之下達到每秒30fps的速度.

(2)時脈100MHz.

所以按照第3章第4節的改進流程,可以加以管線化(pipeline)成主要3級,第一級的功能為過完備小波轉換的分解,第二級的功能是針對第一級小波細節係數做donoho的閥值運算,第三級的主要功能是過完備小波轉換的合成.以下將會做更詳細的解釋.

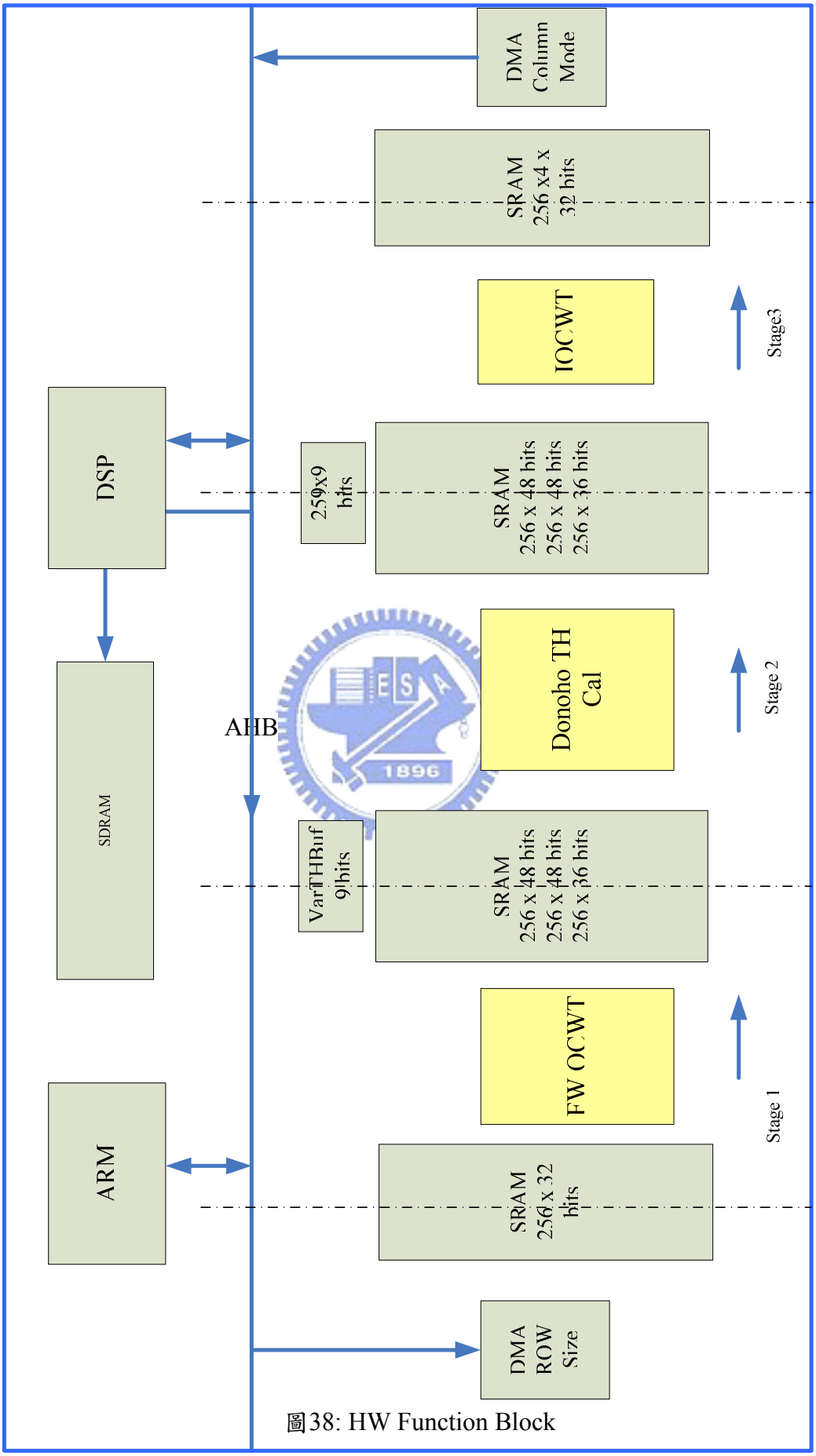

圖38: HW Function Block

5.1. 第一級的處理

在第一級中, 圖39表示出其詳細的示意圖, 其中, 最主要的是做過完備小波轉換的分解, 由於過完備小波轉換的係數[14]中已給出, 所以在表1中可以看到其濾波器係數是由小數來表示, 但是小數在硬體中會造成很大的負擔, 所以我們便想到是否可以由整數係數來做過完備小波轉換的分解, 而在過完備小波轉換的合成時再做適當的整數到小數的轉換, 這樣對硬體而言是最好實踐的方式. 所以由表1可以轉成表5的小波濾波器係數.

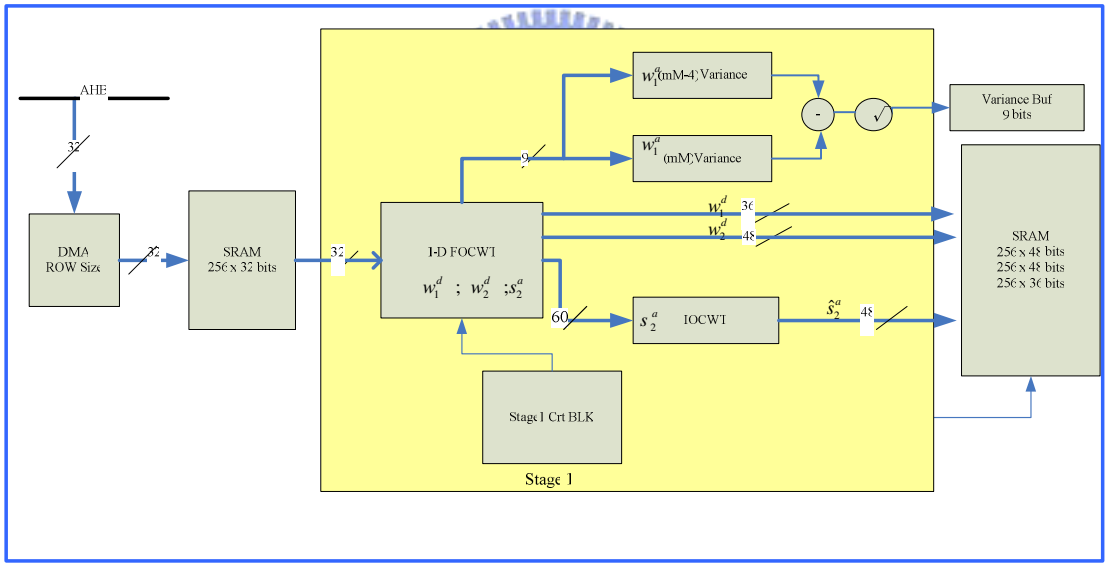

圖 39:第一級功能示意圖

n	H	G	K
-3			1/32
-2			7/32
-1	1/8		22/32
0	3/8	-1/2	-22/32
1	3/8	1/2	-7/32
2	1/8		-1/32

表格 5:過完備小波係數的改進 I

在表5中,可以看出,一般基本的濾波器會使用乘法累加器(Multiplier-Add-Accumulator,MAC),而乘法器是比較耗資源和時間的,所以我們試著用位移器(Shifter)加上加法器(Adder)來做出過完備小波轉換的分解和合成.

由表5可以看到H,G,K的係數其分母皆是2的幕次方,所以用位移器(Shifter)可以來達成,剩下的就是分子也是要用位移器(Shifter)加上加法器(Adder)來做,在H的係數中,3可以用(2^2-1)來達成,在K的係數中,7可以用(2^3-1)來達成,22可以用(2^4+2^3-2) 來達成,所以在過完備小波轉換的分解和合成中,可以完成不用到乘法器來做,這對硬體的大小有很大的改進.

接下來,我們要將過完備小波轉換的分解皆用整數來做,也就是將進來的訊號只對分子部份做運算,而分母的部分將會是在過完備小波轉換合成時做處理,這樣的好處是,從硬體上的第一級和第二級看到的小波係數皆是整數,而不是小數,如此對第二級計算donoho閥值時將會節省很多小數的運算的時間和造成誤差使精確度的降低.且其所須要的儲存器(Buffer)的大小和在要求同樣誤差時所須要做小數運算的儲存器大小一樣.所以表5的濾波器係數可以再處理如表6.

而其過完備小波轉換的分解(圖40)和合成(圖41,圖42)的硬體做法如圖所示.

n	H	G	K
-3			1
-2			7
-1	1		22
0	3	-1	-22
1	3	1	-7
2	1		-1

表格 6:過完備小波係數的改進 II

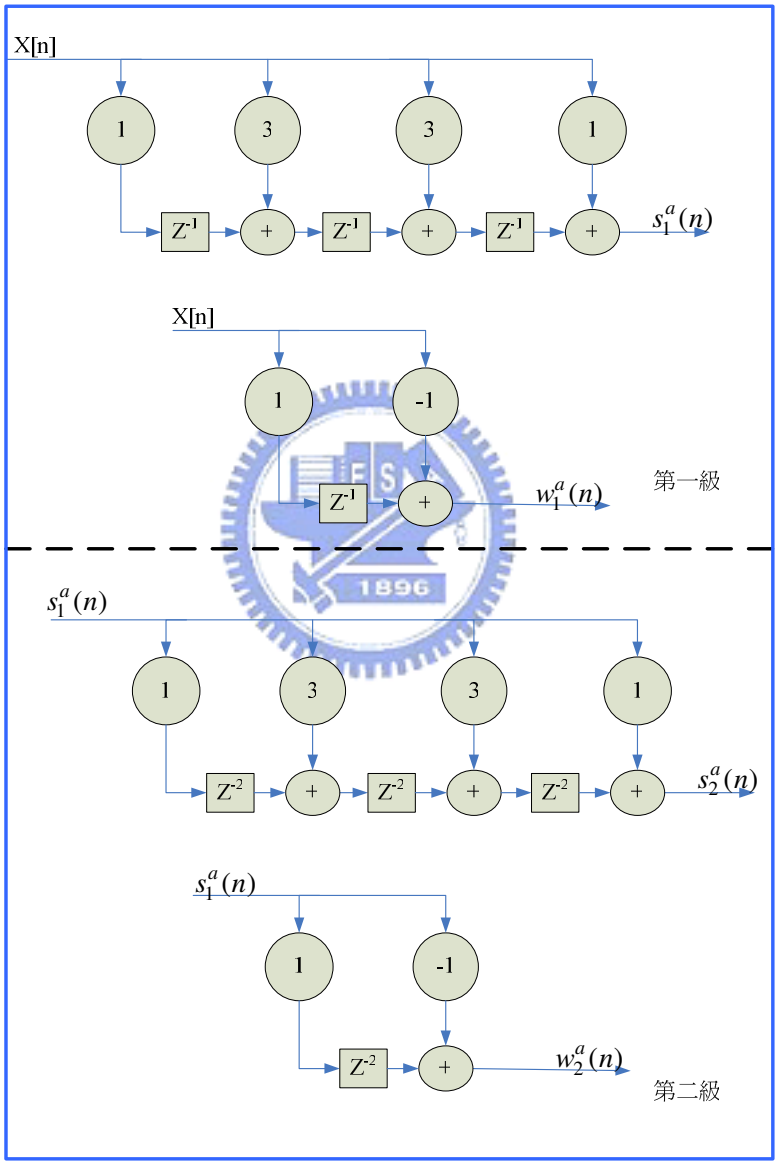

圖 40:整數過完備小波分解

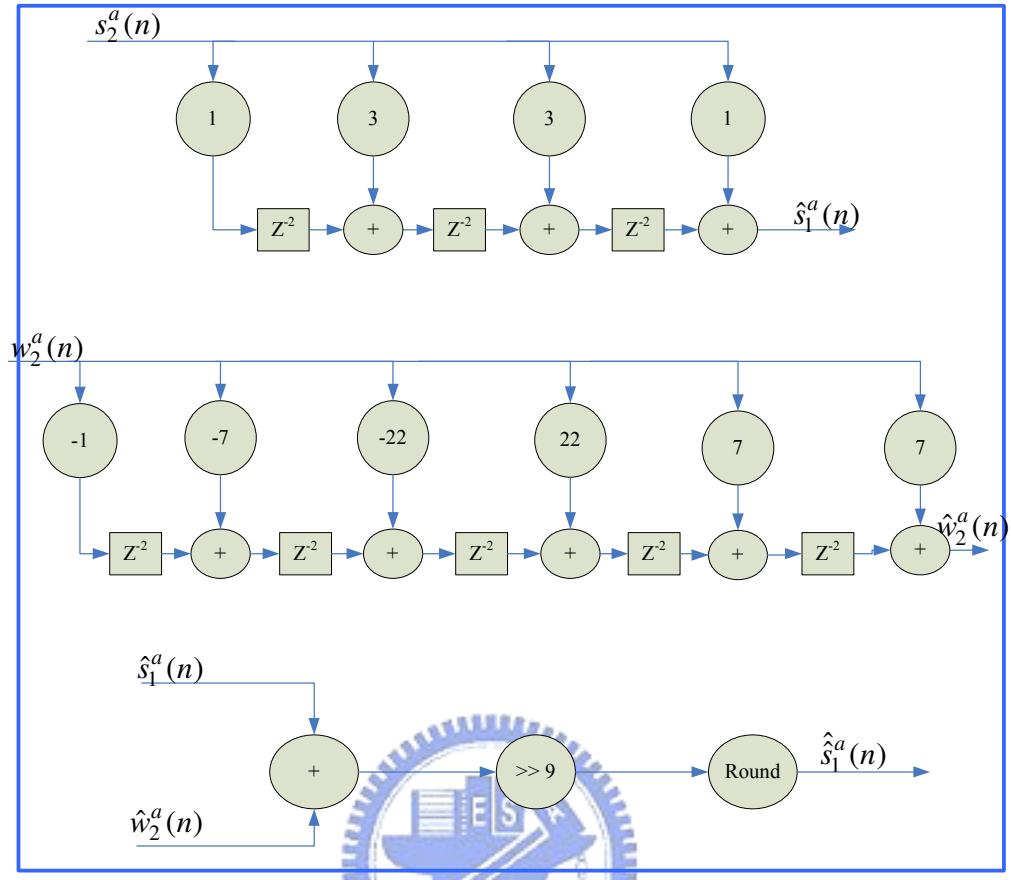

圖 41:第二級整數過完備小波合成

在圖41和圖42中,可以看出,過完備小波轉換在硬體上最長的時間路徑就是在合成時的小波細節係數,其濾波器的係數共有6個,但是這在現在VLSI的技術上,並不是很大的問題,經過Timing Analyze後,其滿足我們對100MHz時脈的要求.

每一級的合成,其結果都會四捨五入(Rounding)成整數,因為最後的輸出都是整數,沒有任何小數的存在.

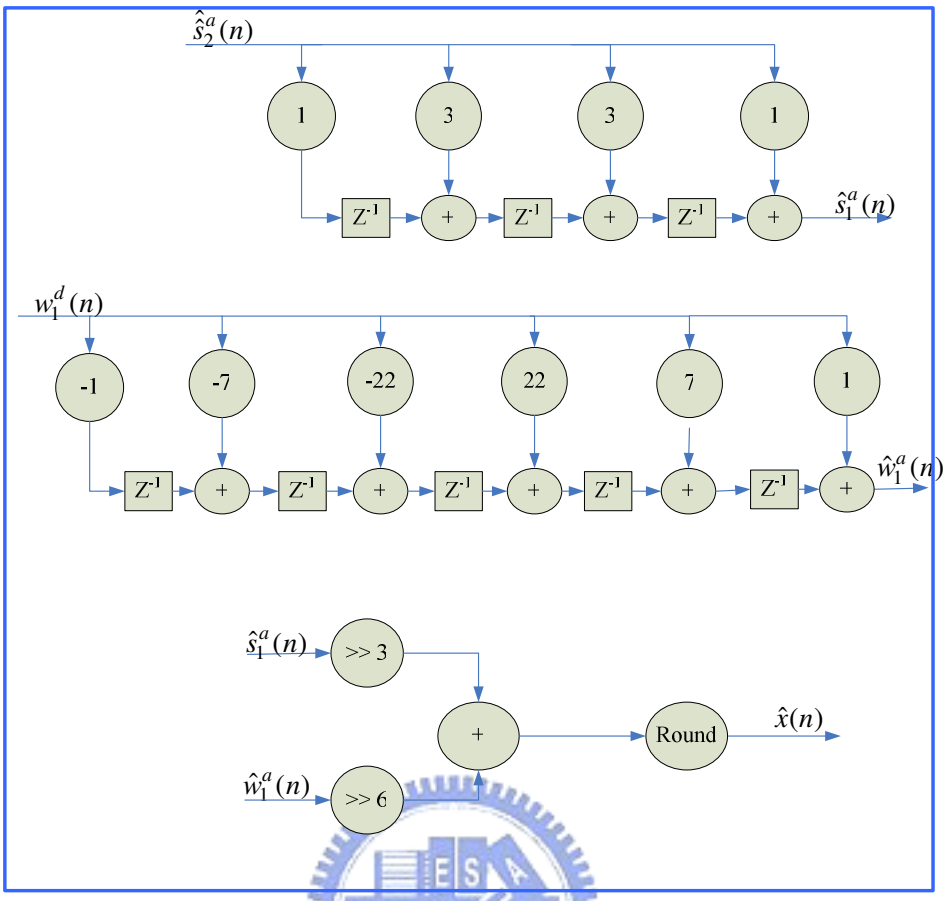

圖 42:第一級整數過小波合成

由上述可以知道以一維過完備小波轉換來做硬體的話,其在濾波器上的硬體複雜度將會是:12 加法器(分解)+ 22 加法器[1] (合成) =34個 加法器.

另外, 在圖41中,因為過完備小波轉換的第二級的小波近似係數在硬體中不會被做任何處理的,所以在這一級中我們為了縮小資料量,會將過完備小波轉換的第二級的小波近似係數先做好合成的動作,其節省的資料量會由原來的60 bits,變成48 bits.節省20%的資料量.對硬體而言會有很大的幫助.

而在第一級中,我們也要算出其判斷是否為邊緣(edge),突波(pulse ,blocking) 或是平滑(smooth)區域的閥值,其演算法如下:

[1] Rounding 也是一個加法器.

$$\sigma^2_{w_1^d} = \mathrm{var}[w_1^d f_b(mM)] - \mathrm{var}[w_1^d f_b(mM - M/2)]$$

$$Th = 2Th1 = c\sigma_{w_1^d}; c = 5.5; M = 8, 1 \le m \le \frac{Length(Signal)}{M} - 1 \qquad (5\text{-}1)$$

由上述(5-1)可以看出,須要的元件為2個變異數一個開根號再加上一個乘法器,其中開根號可以由Y.Li 和W. Chu 所提出的開根號硬體[26]來做出來,在我們的硬體設計中,為了一定的精準度,所以我們共須要13個加法器來做出來.而求變異數則是由公式

$$\sigma^2(Variance) = \frac{1}{N}\sum_{k=1}^{N}(x_k - u)^2$$

$$u : mean \qquad (5\text{-}2)$$

$$N : length \text{ of Signal}$$

$$x_k : input$$

來看,須要的是一個除法器(1/N)和一個乘法器,再加上2個加法器所組成,因為在這的除法器可以用乘法器來代替,所以在這個變異數的算法共須要2個乘法器和2個加法器.

所以在計算閥值上,我們要付出的硬體成本總共是2個乘法器和15個加法器.

綜合上述,在第一級中我們所要付出的硬體成本是**2個乘法器和49個加法器**.

當然,我們都用整數來做的話,確實會有很多冗餘bit產生,如果是用fix point的方法來做的話,再適當的捨棄小數點後第n個bit的方式來做的話,是會節省很多不必要的bit數,但是如果用fix point來做的話,像計算上式5-2的變異數值,會因為fix point的精確度而影響.接下來就會影響判斷blocking , edge或是

smooth region.這當然會在處理完的影像上得到代價,所以在不考慮硬體成本之下,且加上我們所產生的冗餘bit還是可以接受的程度之下,我們是用整數的濾波器係數來做,而沒有用fix point來做運算的精簡.

5.2. 第二級的硬體功能

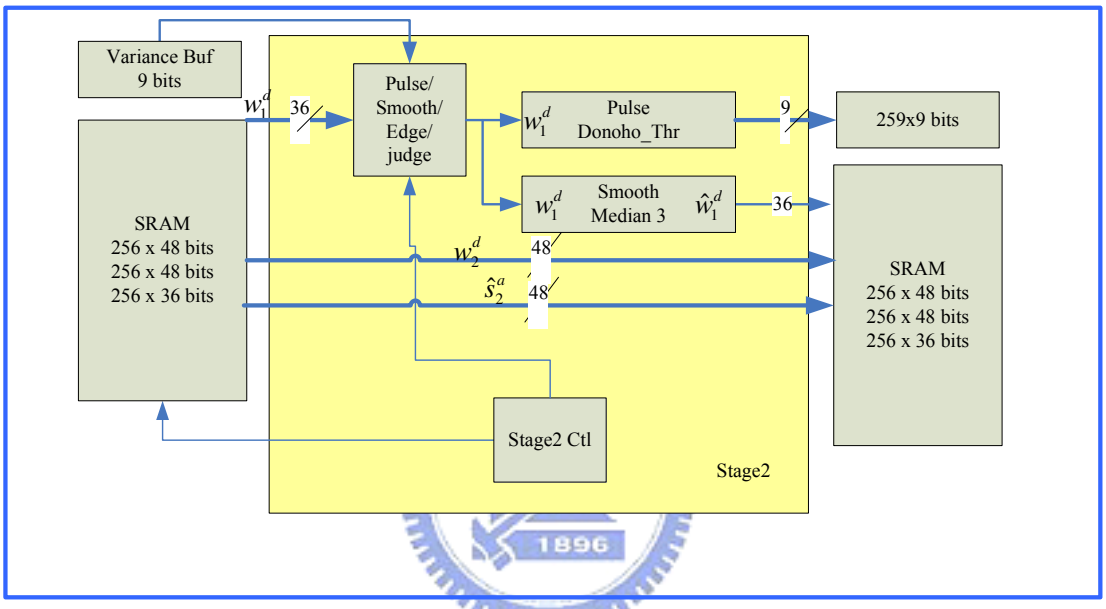

圖 43:第二級功能示意圖

圖43秀出第二級硬體中的功能示意圖.由圖43可以看出本功能主要的功能是而求中位數值(MAV)的實踐方法則是採用[27]。中所用的**Rank-Order Filtering (ROF)**硬體架構,可以在一群無依序排列的數目中,算出想要的次序的結果,只是不採用原文中的DCRAM(dual-cell RAM)的架構,而是用DFF來取代,在[27] 中主要是利用bit-sliced 的方法來求出答案,我們以7個4 bits的數值來求其由大至小的排序中第一個位置的數值來做例子來說明此硬體是如何動作的,見圖44.在圖44中,yi是我們想要的答案.步驟1 是做全部7個數值的最高位元(MSB)的加法,步驟2 再判斷結果是不是大於想要的排序位置(在例中為4,也就是

7組數字的中位數),如果是(y[3]=1),則將7個數值的最高位元(MSB)不是1的,全部設為0,步驟3 再做7個數值的第三位元的加法,再重複步驟2,直到全部4個位元都做過,最後的y就是我們想要的結果.

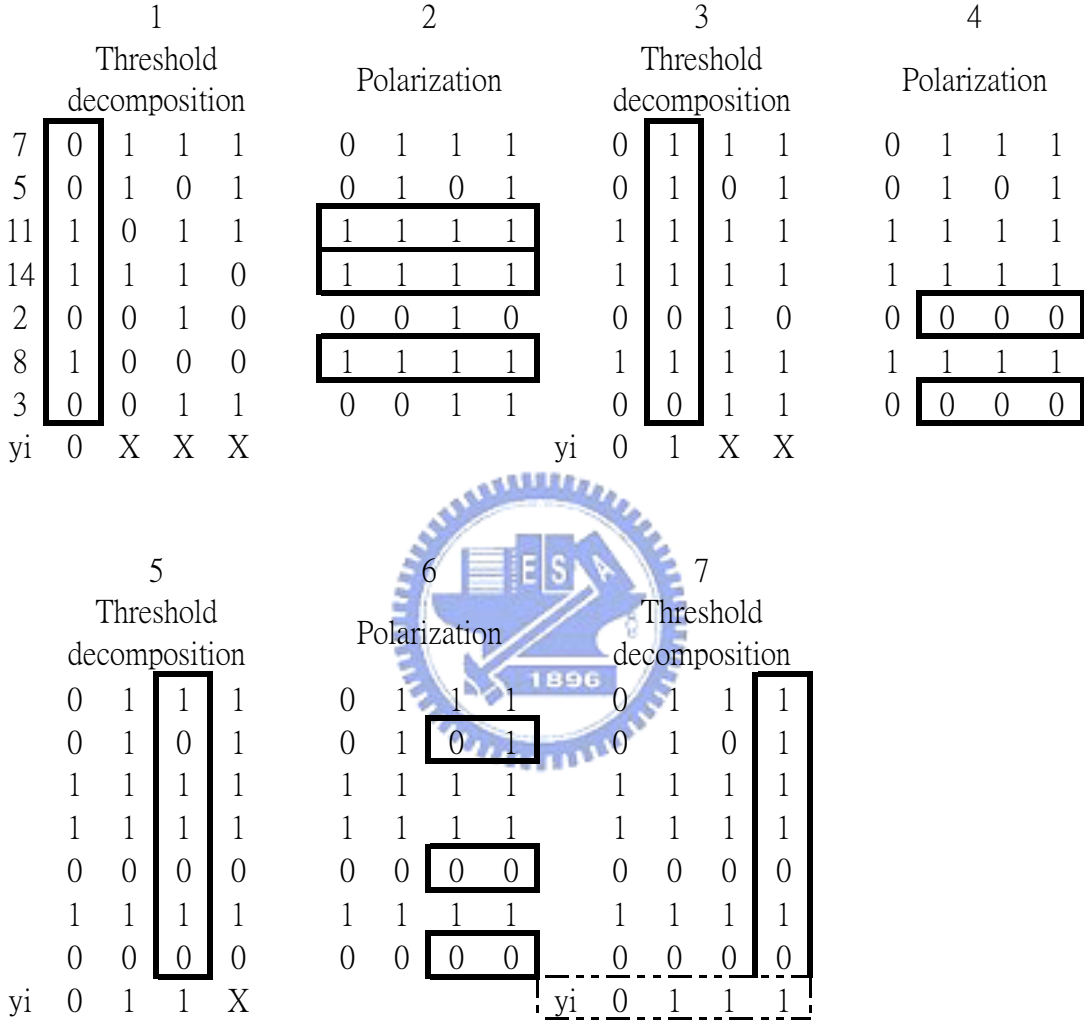

圖 44:ROF 硬體求值的例子

而在求中位數值時,硬體資源花費最大的就是在DFF的地方,因為須要暫存9筆資料在DFF之間,每筆資料是9 bits,所以總共須要9x9=81個DFF.而在求中位數時,先將9筆資料存在DFF中,再經過9次的運算比較,就可以得到我們想要的中位數值了.其中我們須要用到6個加法器來做中位數值運算的判斷.圖45中是其示意

圖.

綜合上述, 在第二級中, 我們共用到了**6個加法器.**

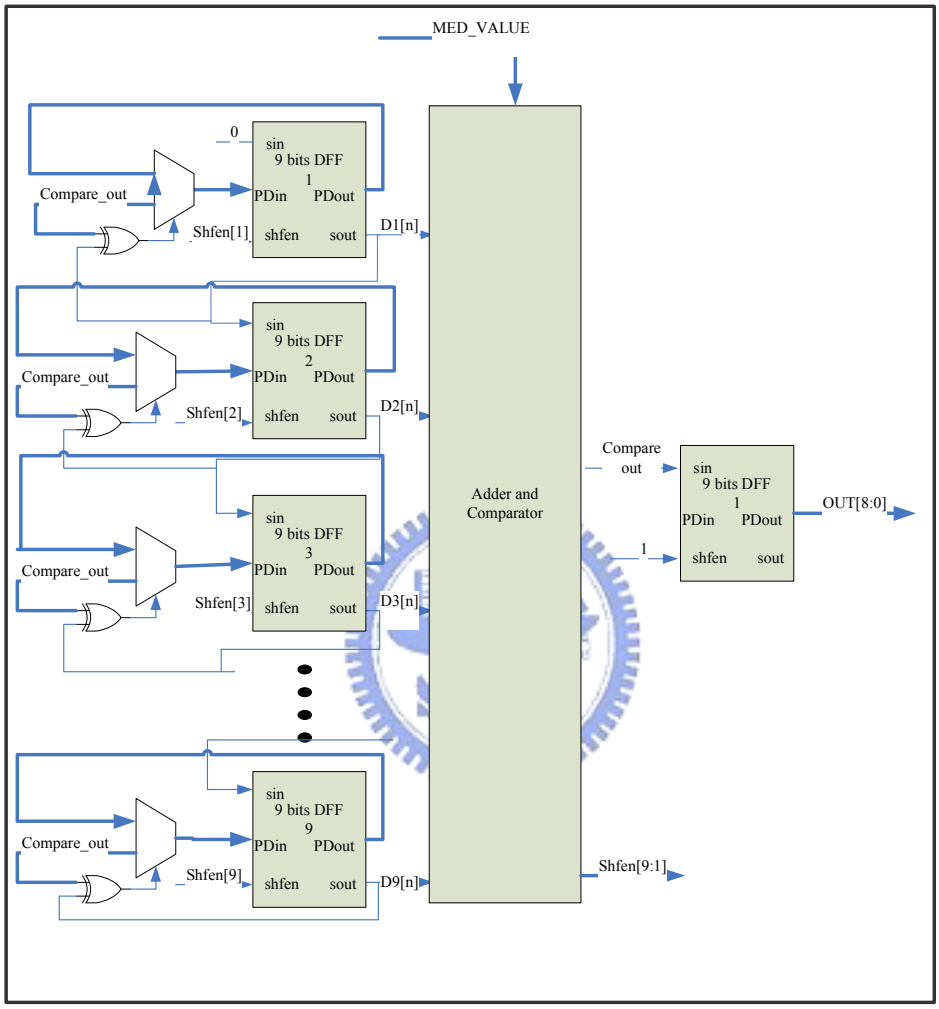

圖 45: 中位數求值之硬體

5.3. 第三級的硬體功能

圖46是第三級的硬體方塊圖, 第三級主要的有二個功能, 一個是做第一和第二級小波細節係數的軟閥值計算, 二是做過完備小波的合成.

軟閥值演算法的閥值是由第二級的donoho閥值計算的結果.

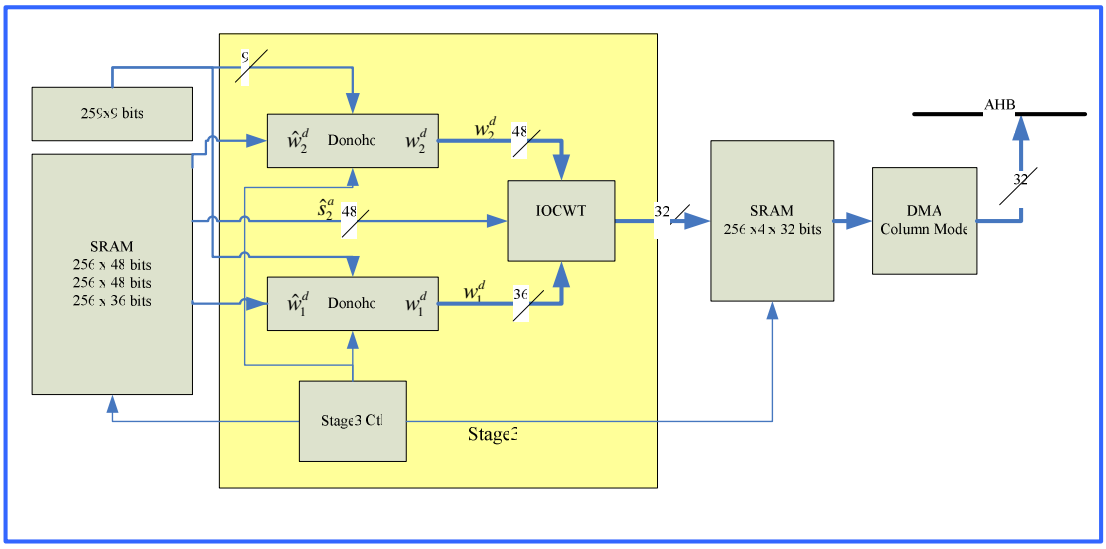

圖 46:第三級硬體方塊圖

在第三級過完備小波合成完成之後,我們要將一維的去方塊完成的資料放到DRAM之中,所以最後還有一個DMA來負責這一件事,為了使我們的硬體可重復性高,我們在把資料放到DRAM時,會轉成行(Column)排放的方式放回DRAM,這個好處是當所有列(ROW)資料都處理完之後,馬上就可以利用原有的硬體來對行的資料做一次去方塊的運算了.而整體的資料流程如圖47.

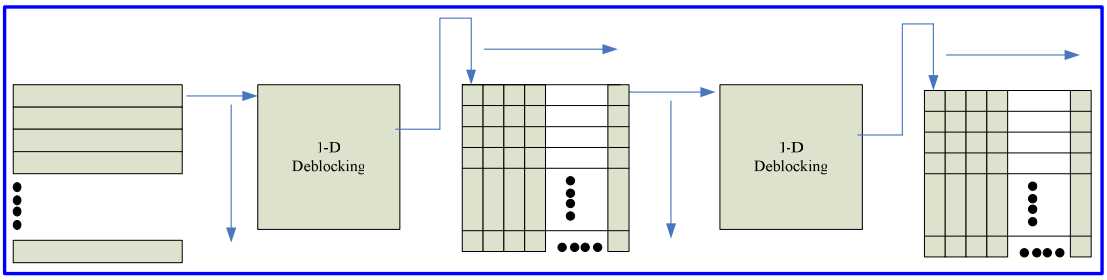

圖 47:本文資料流程圖

特別要注意的是,最後一級的DMA是須要先暫存4個列資料才可以放到DRAM,因為在AHB 上的資料量是以32bits為基礎,而我們要收後4個列(row)的資料才開始以行(column)排放的方式放在DRAM上,但此時的AHB Burst 長度是為1,如果為

了促進AHB的使用率,那麼可以用更多暫存器存更多列資料,再放到DRAM上,但缺點是會須要更多暫存器來使硬體資源增加.所以在AHB的使用率和硬體資源上其實我們可以求得一個平衡點,由圖48可以看出,平衡點應該是當暫存器是以16個列資料(即Burst Length=4)時會有一個極佳的選擇,但為了節省本硬體的空間,在這裡只用了4個列暫存器(4 x 256 x 32,Burst Length=1).如果以後要擴充,只要修改列暫存器的大小和最後DMA的一些地方(Burst Length和存取計數器(Access counter))即可.這也增加了未來的可擴充性.表7列出了AHB的Burst Length和相對的列暫存器的大小對DRAM 的使用率的關係.其中DRAM的使用數在這是指把512個列放到DRAM時所須的DMA的次數.

Burst Length	1	4	8	16
Buffer Size	4x256x32	16x256x32	32x256x32	64x256x32
DRAM 使用數	128x512	32x512	16x512	8x512

表格 7:AHB Burst Length vs Buffer Size

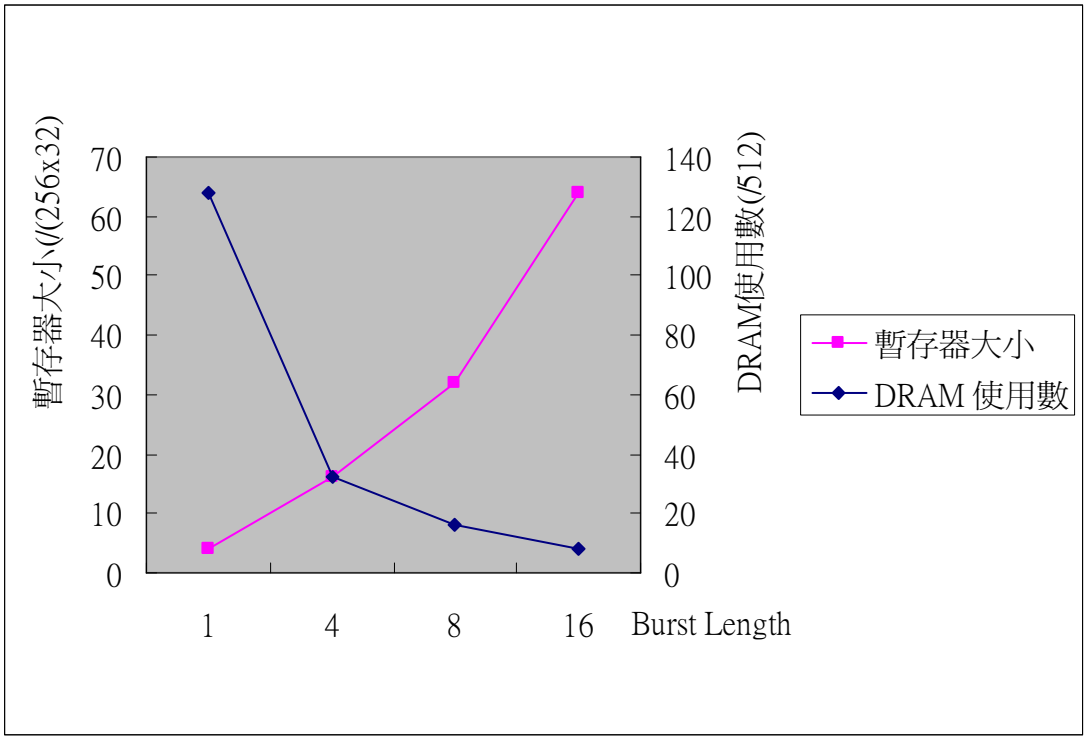

圖 48:暫存器和 Burst Length 大小的關係圖

5.4. 結論和比較

在這一節中,我們將MPEG4 中[45]的去方塊效應的硬體複雜度和本文所提出的硬體複雜度做一個比較.

以5-1,5-2和5-3節可以知道,就本文所提出的硬體改進方式,所用到的基本元件是:2個乘法器和54個加法器.但因為過完備小波的資料量較大,在本文中的小波階數為2,所以其所須的資料量複雜度為O(3N).

而MPEG4的去方塊效應的方法簡述如下(圖49):

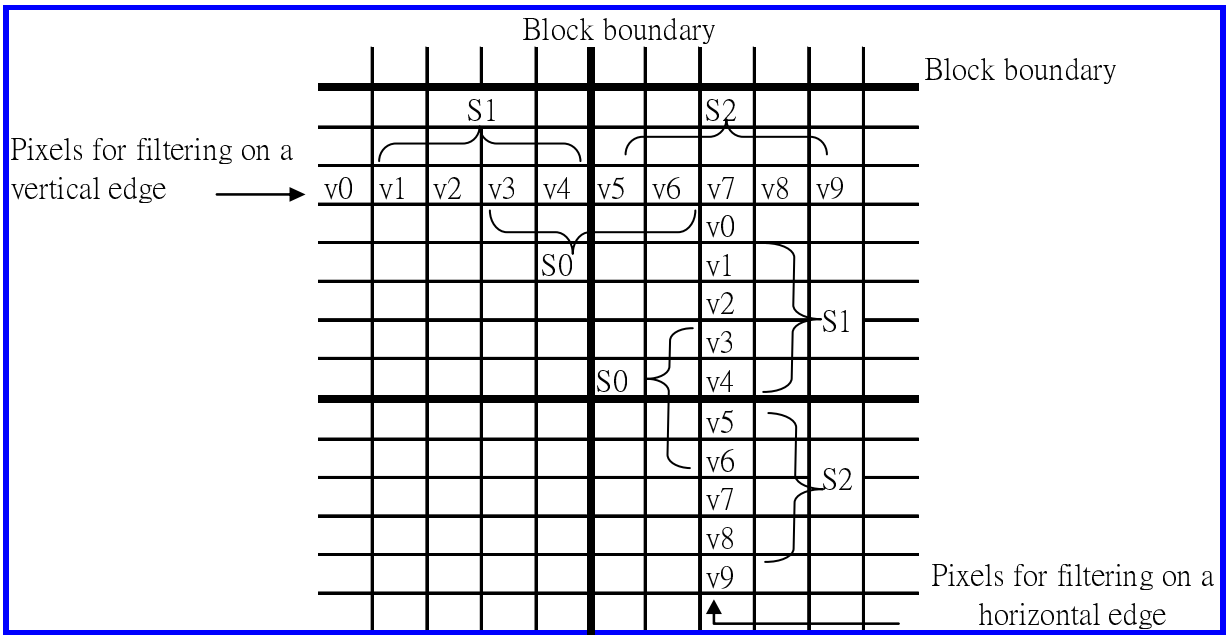

圖 49:MPEG4 deblocking Method

$$eq\_cnt = \phi(v0-v1)+\phi(v1-v2)+\phi(v2-v3)+\phi(v3-v4)$$
$$+\phi(v4-v5)+\phi(v5-v6)+\phi(v6-v7)+\phi(v7-v8)+\phi(v8-v9),$$
$$where \quad \phi(\gamma)=1 \; if \; |\gamma| \le THR1 \text{ and } 0 \text{ otherwise.}$$

If $(eq\_cnt \ge THR2)$

DC offset mode is applied, **(5-5)**

else

Default mode is applied.

endif

THR1=2 and THR2=6.

式5-5是用來判斷方塊效應發生的位置是否在灰階變化緩慢(smooth region)之處,如果方塊效應不是在灰階變化緩慢的區域,只要用可調性的平滑濾波器(adaptive smoothing filter,即Default Mode)來做即可,見式5-6,由式5-5可以看出,判斷式所須的硬體為15個加法器.

Default mode

$v_4' = v_4 - d,$

$v_5' = v_5 - d,$

$d = CLIP(5 \cdot (a_{3,0}' - a_{3,0})//8, 0, (v_4 - v_5)/2) \cdot \delta(|a_{3,0}| < QP)$

$where\ a_{3,0}' = SIGN(a_{3,0}) \cdot MIN(|a_{3,0}|, |a_{3,1}|, |a_{3,2}|).$

$a_{3,0} = ([2,-5,5,-2] \bullet [v_3, v_4, v_5, v_6]^T)//8,$

$a_{3,1} = ([2,-5,5,-2] \bullet [v_1, v_2, v_3, v_4]^T)//8,$

$a_{3,2} = ([2,-5,5,-2] \bullet [v_5, v_6, v_7, v_8]^T)//8,$

$CLIP(x, p, q)$: *clips* x to a value between p and q,

QP:quantisation parameter of the macroblock where pixel v_5 belongs.

δ(condition)=1,if the "condition" is true and 0 otherwise.

// : rounding the the nearest integer.

(5-6)

式5-6是針對方塊效應發生在非灰階變化緩慢的區域來做平滑化,而所花的硬體為16個加法器,其中計算$a_{3,0}, a_{3,1}, a_{3,2}$就須要12個加法器.

如果是在灰階變化緩慢的地區,則須採用DC offset mode來做去方塊效應.見式5-7:

由式5-7可以看出,其須要9x8=72個加法器來做.

綜上所述,MPEG4 去方塊效應其須要72+16+15=103個加法器來達成,但因為MPEG4是時域上的處理,所以資料複雜度為O(N).

DC offset mode:

$$\max = MAX(v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8),$$

$$\min = MIN(v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8),$$

$$if\ (|\max - \min\} < 2 \cdot QP)\{$$

$$v'_n = \sum_{k=-4}^{4} b_k \cdot p_{n+k}, 1 \le n \le 8,$$

$$p_m = \begin{cases} (|v_1 - v_0| < QP)\ ?\ v_0 : v_1, if\ \text{m<1} \\ v_m, if\ 1 \le \text{m} \le 8 \\ (|v_8 - v_9| < QP)\ ?\ v_9 : v_8, if\ \text{m>8} \end{cases}$$

$$\{b_k : -4 \le k \le 4\} = \{1,1,2,2,4,2,2,1,1\}//16$$

$\}$

$else$

No change will be done (5-7)

表8為MPEG4去方塊效應和本文的去方塊效應硬體的乘法器和加法器比較表.

	Multiplier	Adder	Data
MPEG4	--	103	N
Ours	2	54	3N

表格 8:硬體複雜度比較表

由表8可知,因VLSI的進步,如果不計資料的處理量,則採本文所做的去方塊效應硬體的複雜度其實並沒有比MPEG4增加很多,但是去方塊效應的效果卻比MPEG4的還不錯.在合成方面,我們採用的UMC .13 製程的library來合成,圖50是DC 合成完的cell area的報告,結果為本文的去塊效應共用了45K的Gate count,而圖51是max path 的報告表示在設定時脈週期為9ns時,全部的電路都符合我們的要求.表9是硬體的Gate Count 和內部Sram的大小說明.其中第３項和第２項主要的意義為以最後一級Column DMA為例，在D1的大小之下，我們實際上用到的只有180x32x4，但因為在Memory Compiler的限制之下，我們只能用256x32x4

的memory,也就是有29.7%是沒有用到的.

1	2	3
Gate Count	HW Sram Size(Byte)	Sram Size(use)(byte)
45938	27.5625K	19.29K

表格9:硬體規格大小

```
information, use the "link" command. (UID-341)
Information: Updating design information... (UID-85)
Warning: Design 'deblocking_top' contains 2 high-fanout nets. A fanout number of 1000 will be
used for delay calculations involving these nets. (TIM-134)

****************************************
Report : area
Design : deblocking_top
Version: W-2004.12-SP5-1
Date   : Fri Jan 19 10:44:05 2007
****************************************

Library(s) Used:

    fsc0h_d_sc_wc (File: /home/user/designer/leochen/UMC13lib/std_cell/fsc0h_d/2006Q1v2.0/SC/
FrontEnd/synopsys/fsc0h_d_sc_wc.db)

Number of ports:              432
Number of nets:              1361
Number of cells:               20
Number of references:          12

Combinational area:       162233.000000
Noncombinational area:    71594.000000
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:          233827.000000
Total area:                 undefined

Information: This design contains black box (unknown) components. (RPT-8)
```

Gate count=
233827/5.09=45938

圖50:DC的cell area report

```
  stage1_inst0/B4_variance_inst0/add_198/U1_16/CO (FA1DHD)
                                                  0.26       7.63 f
  stage1_inst0/B4_variance_inst0/add_198/SUM[17]
(variance_out_dt1_DIN_SIZE9_FIX_BIT_NOO_0_DW01_add_18_0)
                                                  0.00       7.63 f
  stage1_inst0/B4_variance_inst0/U138/O (AN2EHD)  0.14       7.78 f
  stage1_inst0/B4_variance_inst0/U139/O (AN2EHD)  0.14       7.92 f
  stage1_inst0/B4_variance_inst0/U140/O (AN2EHD)  0.14       8.06 f
  stage1_inst0/B4_variance_inst0/U141/O (AN2EHD)  0.14       8.20 f
  stage1_inst0/B4_variance_inst0/U142/O (AN2EHD)  0.14       8.34 f
  stage1_inst0/B4_variance_inst0/U173/O (AN2CHD)  0.14       8.48 f
  stage1_inst0/B4_variance_inst0/U125/O (XNR2EHD) 0.17       8.65 r
  stage1_inst0/B4_variance_inst0/U124/O (NR2CHD)  0.05       8.70 f
  stage1_inst0/B4_variance_inst0/sum_out_reg[23]/D (DFERBCHD)
                                                  0.00       8.70 f
  data arrival time                                          8.70

  clock sys_clk (rise edge)                       9.00       9.00
  clock network delay (ideal)                     0.00       9.00
  stage1_inst0/B4_variance_inst0/sum_out_reg[23]/CK (DFERBCHD)
                                                  0.00       9.00 r
  library setup time                             -0.30       8.70
  data required time                                         8.70
  ---------------------------------------------------------------------------
  data required time                                         8.70
  data arrival time                                         -8.70
  ---------------------------------------------------------------------------
  slack (MET)                                                0.00


  Startpoint: stage1_inst0/B4_variance_inst0/mean_sum_out_reg[15]
```

圖51:DC的max path report

表10則是把內部的SRAM佔所有全部的SRAM的比例和大小做一個統計.

	DMA in	OCWT	DMA out	Others
Size(bit)	256x32x2 (16x1024=16k)	256x48x8 (96k) 256x36x4 (36k)	256x32x4x2 (64k)	256x9x2 (4.5k)
%	8%	60%	30%	2%

表格10:SRAM 統計表

由表10可以看出,最大比例是在過完備小波係數的儲存,這也是過完備小波的缺點,資料量會比傳統的快速小波轉換大.不過這些Sram大小對現在VLSI而言還是可以接受的.

本文也將實際實現的MPEG4 deblocking 演算法用verilog來實現,且經過simulation, synthesis之後的結果來和本文的deblocking硬體在Gate counter, sram size 和 power estimation來做比較(表格11).

	Gate Count	Sram Size(byte)	Dynamic Power
Proposal	45938	27.5625K	11.886mW
MPEG4	15523	5.012K	2.2864mW

表格11:MPEG4 和本文deblocking演算法的比較

由上表可以看出,本文的演算法的Gate Count是約為MPEG4的deblocking演算法的3倍,也就是可以看出其實spatial domain(MPEG4 deblocking)的deblocking 演算法的硬體大小還是比較小的,但是其時以現在0.13um的製程,本文的Gate Count 也還是在可以接受的範圍之內.而Sram Size上的比較就可以看

出過完備小波轉換的內部暫存確實須要很大,這也是過完備小波轉換的缺點.因為本文的Gate count 和Sram Size 故比MPEG4 deblocking都大,所以power 的消耗上會比MPEG4還大,但是還是可以被接受的.所以雖然本文的硬體成本比MPEG4 deblocking 大,但就處理影像的成果上,是比MPEG4還大的.

下圖(圖52)秀出本FPGA的實際圖,FPGA我們採用的是Altera STRATIXII EP2S180 FC1020 -5.而在FPGA內部除了deblocking硬體之外,還會有DSP的core會被放在FPGA內.

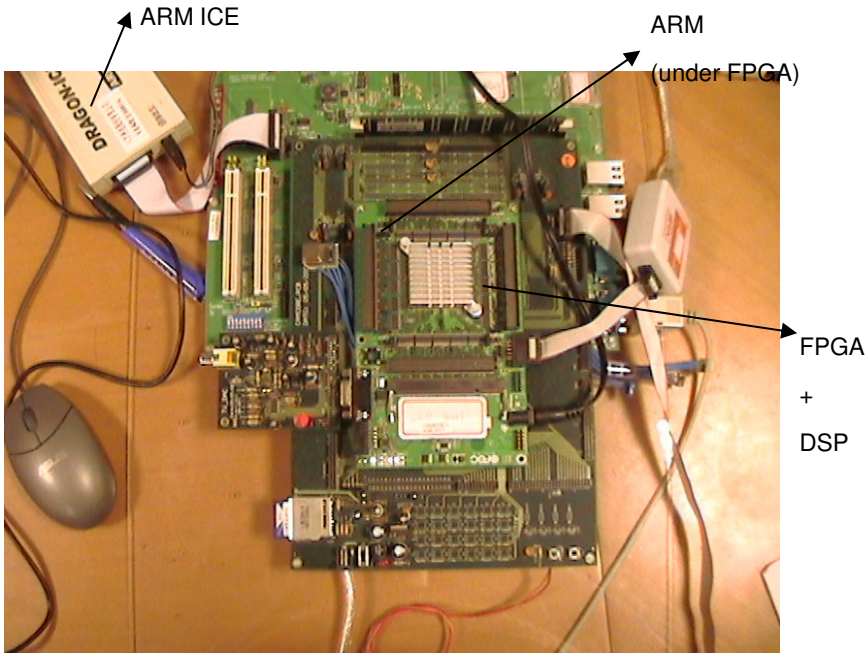

圖52:FPGA board圖

而在外部Dram的存取是應用ARM的AHB來對外部的DRAM做存取的動作,而在我們的系統之中AHB 和硬體內部頻率是30MHz.由於整個系統會是由內部的DSP來控制deblocking硬體,所以在我們的系統中,啟動的程序會是由USER下命令給ARM,再由ARM下命令給DSP,最後DSP會啟動整個deblocking硬體.原因是當我們的系統

在另一套硬體應用時,如MPEG4 encoder時,DSP可以用來做Motion estimation 的rate control,還有在做AV整合時,DSP也可以用來做Audio的編解碼.所以我們的硬體是由DSP來做啟動的動作,而非直接由ARM來做控制.

在處理速度上,我們利用軟體來計算每秒可以處理的圖像速度(fps),在FPGA上的頻率是30MHz,以512x512的圖片來做測試,每一次處理完512個列(行)須要的是23ms(在30MHz之下),所以完成處理完一張512x512的圖片是須要46ms,即在30MHz之下,可以做到的是1000/46=21fps,換成如果是利用100MHz來做的話,就是可以做到21x3.3=69fps.而在D1的大小之下(720x480,含420YUV)其資料量是512x512的720x480x1.5/(512x512)~=2,所以就算是在D1的大小之下,我們還是可以做到30fps的規格的.只是這是AHB在只有deblocking硬體之下的結果,如果搭配其他的硬體的話,那麼可能須要利用5.3節提到的調整column DAM buffer size來做到每秒可以30fps.圖53是利用軟體求出的處理速度.

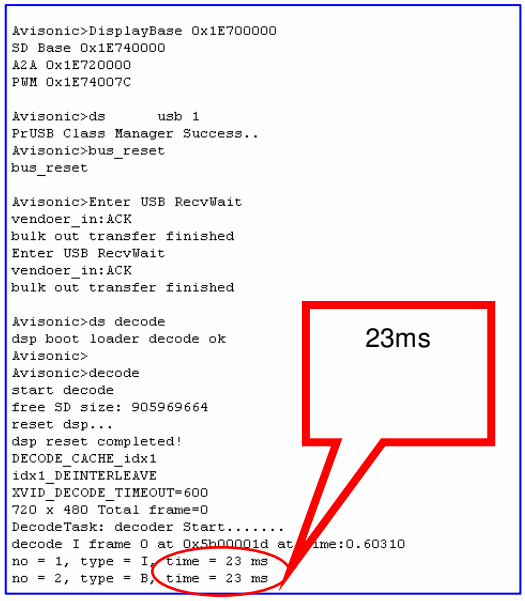

圖53:硬體處理速度

第六章、結論和未來改進

在本文中,所提出的去方塊效應演算法可以有效的去除具有方塊效應的位置,對經由因為高壓縮比而有方塊效應的影像,不僅在人眼主觀判斷有明顯的改進,在 PSNR 量化的量測上也有有效改善的效果.主要是利用過完備小波和 Donoho 去噪演算法,而在過完備小波的硬體上,提出了一個幾乎和小數過完備小波有著一樣的精準度的整數化過完備小波架構,除了精準度外,也只用到了加法器和移位器來減少硬體的複雜度.

在 Donoho 去噪演算法中,提出了分段式 Donoho 去噪演算法,籍由適當的分段可以達到無分段的 Donoho 去噪演算法的效果,且大大的降低傳統無分段的 Donoho 去噪演算法,這在未來想實踐 Donoho 演算法在其他的應用上,可以利用相關的方法來做成硬體.

基本上,利用過完備小波和 Donoho 去噪演算法來做去方塊效應雖然看起來硬體的設計上會十分龐大複雜,但利用上述的方法之後,其實以現有的 VLSI IC 技術其實是夠小且可以到實用的地步.

接下來,可以再改善的就是因為去方塊和高壓縮比的結果,影像會有所謂 ring effect 的發生,要解決 ring effect 首要的是要做影像的邊緣(edge)判定,因為 ring effect 主要是發生在影像的邊緣附近,而要判斷 ring effect 在時域上常用的就是 Sobel 和 Canny 算子,在小波上也有所謂的 WTMM(wavelet

transform modulus maxima),利用過完備小波的Lipschitz regularity特性[14][20]來做影像的邊緣判斷,且可以得到很好的效果.所以未來可以考慮用過完備小波一起處理去方塊效應和ring effect的應用上.