數位積體電路設計期中報告 - 1d DCT

第三組

學號:411086006 姓名:張愷和

學號:411086022 姓名:張君愷

ー、 DCT 介紹

DCT,discrete cosine transform,一種類似傅立葉轉換的方法,主要目的應用在影像壓縮(JPEG 壓縮)、音訊編碼、圖片編碼等等。由於之前有修過多媒體訊號處理的課程,也有實作過模擬 JPEG 壓縮,因此來說明一下DCT 在 JPEG 中的應用。

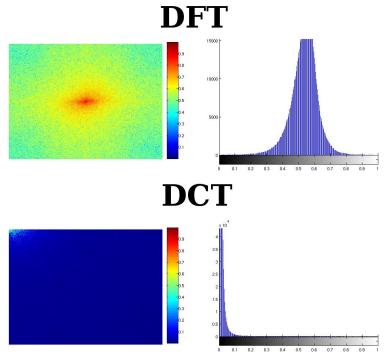
在圖片和聲波一樣,有分高頻和低頻的區域,如果像素數值與附近像 素數值相差較大,可以稱區域為高頻區域,如果像素數值與附近像素數值 相差較小,可以稱區域為低頻區域(圖一)。由於人眼對於高頻訊號的改變



圖一:圖片高頻區域(紅圈)、圖片低頻區域(藍圈)

沒有那麼敏感,因此可以將圖片由像素數值,轉換為頻域數值,之後對高頻區域做量化,刪減高頻的數值,這樣可以達到壓縮的效果,並且不太會影響到圖片視覺上的體驗。

不過可以使用 DFT(discrete fourier transform)或 FFT(fast fourier transform)來轉換頻域,為什麼要用 DCT 來轉換,從圖二可以看到 DCT 轉換可以對低頻保留較多的訊息,轉換範圍較小,儲存訊息量較多,但是 DFT 轉換後分布的範圍較廣,低頻儲存量較小,做量化會刪掉太多的細節,因此 DCT 來轉換是比較好的選擇。



圖二:DFT和 DCT轉換結果差異

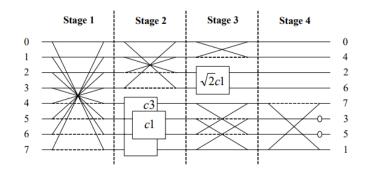
以下是 DCT(式一)和 inverse DCT(式二)的數學公式:

$$F(k) = \begin{cases} \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} f(i), k = 0\\ \sqrt{\frac{2}{N}} \sum_{i=0}^{N-1} f(i) \cos \frac{(2i+1)k\pi}{2N}, k = 1, 2, 3, ..., N-1 \end{cases}$$
 (\$\frac{1}{2}\$)

$$f(i) = \frac{1}{\sqrt{N}}F(0) + \sqrt{\frac{2}{N}} \sum_{k=0}^{N-1} F(k) \cos \frac{(2i+1)k\pi}{2N}$$
 (式 2)

上面兩個公式如果要直接實作在 verilog 中的話,會用到比較多的加法器以及乘法器,使用到較多的加法器和乘法器,對應設計的功耗以及效率會較差,因此有人簡化了 DCT 演算法,得到了另一個 loeffler DCT algorithm,這個演算法總共會使用 29 個加法器以及 11 個乘法器。以下圖片是 loeffler DCT 演算法計算的階段,以及各個 block 計算的方式。在圖四中可以看到 kcn symbol equation 上會使用到四個乘法和兩個加法,但是 effect 上顯示的乘法器和加法器數量都是 3,會減少一個乘法器是因為 equation 可以改成另一種公式寫法(式三),所以結果來說可以在減少一個乘法器。

$$\begin{cases}
O_0 = aI_0 + bI_1 = (b - a)I_1 + a(I_0 + I_1) \\
O_1 = -bI_0 + aI_1 = -(a + b)I_0 + a(I_0 + I_1)
\end{cases}$$
(式三)



圖三: loeffler's DCT algorithm 計算 stage

Symbol	Equation	Effect
I_0 O_0 O_1	$O_0 = I_0 + I_1$ $O_1 = I_0 - I_1$	2 add
I_0 kcn O_0	$\begin{split} O_0 &= I_0(k\cos\frac{n\pi}{16}) + I_1(k\sin\frac{n\pi}{16}) \\ O_1 &= -I_0(k\sin\frac{n\pi}{16}) + I_1(k\cos\frac{n\pi}{16}) \end{split}$	3 mult. +3 add
<i>I</i> —	$O = \sqrt{2}I$	1 mult.

圖四:各個 symbol 代表的計算方法

二、 1d-DCT 實作 verilog 程式碼說明

程式碼有實作兩個版本,第一個版本是按照上方 loeffler's DCT algorithm 一步一步按照 stage 的方法計算的,總共使用了 29 個 adder 以及 11 個 multiplier,但是可能是我 verilog 學藝不精,以結果來說用 design compiler 產出的面積和時間都不太理想,使用 100MHz clk 來模擬 這個程式,slack time 非常接近 0,使用 primetime 測出來的時間報告會 變成負數。

第二個版本是直接使用 always blocking 寫法,這裡就可以用像是 c 語言的寫法,直接將教授給的 c 語言範例翻譯成 verilog 語言。結果來說 design compiler 合成出來的面積和計算的 power 比第一個版本還要小,但是 slack time 差不多,使用 primetime 測出來的時間報告也是會變成負數。

1. Version 1 code

首先需要設定一些常數項,這裡的常數項設定唯一些已經計算好的 cos 和 sin 值,不過 verilog 中不能直接計算小數,所以需要將計算好的常數乘上 2 的次方項,次方越大之後計算出來的結果精度會越好,但是計算過程中需要開較大空間的 wire 來避免計算出現overflow,最後結果需要將之前乘上的 256 除回來,實作上使用右移 8 位元實現除法,圖五是設定各項 sin、cos 計算常數。

```
//b sin //a cos //中間的a和m代表加和減 //b-a sin - cos //sq表示有根號2 //b+a sin + cos //數字表示對應的kcn n數值 wire signed [10:0] clas1 = 9'd301; wire signed [10:0] slmc1 = -201; wire signed [10:0] c1 = 9'd251; wire signed [10:0] c3as3 = 9'd355; wire signed [10:0] s3mc3 = -71; wire signed [10:0] c3 = 9'd213; wire signed [10:0] sqc1as1 = 9'd470; wire signed [10:0] sqc1as1 = 9'd196; wire signed [10:0] sqc1 = 9'd137; wire signed [10:0] sqc1 = 9'd362;
```

圖五:各項計算常數設定,都有在乘上 256(2^8)

接著就可以將各項數值通過各個 stage 來計算結果,再將結果丟到下個 stage 計算。

Stage 1:

圖六: stage 1 計算程式碼

Stage 2:

圖七: stage 2 計算程式碼

```
assign tmp1 = (in1 * in2) >> 8;
assign tmp2 = (in5 * in6) >> 8;
assign out1 = tmp1 + tmp2;
assign tmp3 = (in3 * in4) >> 8;
assign out2 = tmp2 - tmp3;
```

圖八:對應的 mult_kcn module 計算方法(式 3 計算方法)

Stage 3:

```
assign s30 = s20 + s21;

assign s31 = s20 - s21;

assign tmp03 = s22 + s23;

mult_kcn m3(s23, sqs1mc1, s22, sqc1as1, tmp03, sqc1, s32, s33);

assign s34 = s24 + s26;

assign s36 = s24 - s26;

assign s37 = s27 + s25;

assign s35 = s27 - s25;
```

圖九: stage 3 計算過程, mult kcn 可參照圖八

Stage 4:

```
assign o0 = s30;
assign o4 = s31;
assign o2 = s32;
assign o6 = s33;
assign o1 = s37 + s34;
assign o7 = s37 - s34;
assign o3 = (s35 * sq2) >> 8;
assign o5 = (s36 * sq2) >> 8;
```

圖十:最後 stage 4 計算過程

首先使用 vsim 先模擬計算是否正確,以下比較 verilog 和 c 語言計算 出來的結果。

```
| Marco Debaut: | Marco | Marc
```

圖十一: c 語言計算和 verilog 計算數據差異

由圖十一可以看到計算公式上基本上應該是沒有出錯,沒有看到太離譜的數值錯誤,至於計算精度上面,最差有到正負 3,看起來只乘上 2 的 8 次方計算精度還是沒有那麼好。接下來使用 design compiler 來看 area、time 和 power 的結果。

圖十二:合成 area 結果, total cell area 為 105690.736132

```
      mult_107/U173/0 (NR2)
      0.065
      14.291 f

      mult 107/U168/0 (NR2)
      0.148
      14.439 r

      mult_107/U207/0 (INV1S)
      0.076
      14.515 f

      mult_107/U207/0 (A0112HS)
      0.191
      14.706 r

      mult_107/U223/0 (A0112HS)
      0.091
      14.796 f

      mult_107/U229/0 (X0R2HS)
      0.104
      14.900 r

      mult_107/product[18] (loeffler_1d_DW_mult_tc_5)
      0.000
      14.900 r

      out3_reg[10]/D (QDFFRBT)
      0.000
      14.900 r

      clock clk (rise edge)
      15.000
      15.000

      clock network delay (ideal)
      0.500
      15.500

      clock uncertainty
      -0.500
      15.000 r

      out3_reg[10]/CK (QDFFRBT)
      0.000
      14.901 r

      data required time
      14.901

      data required time
      14.901

      data arrival time
      -0.099
      14.901

      data arrival time
      -14.900
```

圖十三: time 結果, slack time 為 0.001

從圖十三的結果來看, clock 設定為 100MHz 的話會差一點點就 violate slack time。

Power Group	Internal Power	Switching Power	Leakage Power	Total Power			Attrs
io pad	0.0000	0.0000	0.0000	0.0000	(0.00%)	
memory	0.0000	0.0000	0.0000	0.0000		0.00%)	
black box	0.0000	0.0000	0.0000	0.0000		0.00%)	
clock network	0.0000	0.0000	0.0000	0.0000		0.00%)	
register	1.1014	1.2252	2.8724e+04	2.3266		18.26%)	
seguential	0.0000	0.0000	0.0000	0.0000		0.00%)	
combinational	5.2794	5.1374	3.7010e+05	10.4171		81.74%)	
Total	6.3807 mW	6.3626 mW	3.9883e+05 pW	12.7437 1	nW		

圖十四:預測的 total power 為 12.7437mW

版本一的程式碼測試到這邊後,就開始改良程式碼架構,一開始式認為說中間宣告的 wire 太多占太多空間,於是就把中間的線一個一個拔掉和合併計算公式,但是出來的結果 slack time 只多了 0.01,跑primetime slack time 還是會爆掉,面積跟 power 也沒有變得多好,之後使用了 always blocking 的寫法,以面積跟 power 來說有比較好,但是 slack time 還是一樣,不過最後還是決定用第二個版本的程式碼來跑primetime,這樣 power 的數據比較好看。所以來看一下第二個版本的程式碼吧。

2. Version 2 code

這個版本使用 always blocking 方法實作,以下是程式碼架構,計算過程是放在 always block 裡面。

```
tmp = (vp0 + vp1 + vp2 + vp3);
out0 = tmp;
tmp = (vp0 - vp1 - vp2 + vp3);
out4 = tmp;
tmp = ((vp1-vp2)*a + (vp0-vp3)*b) >> 10;
out2 = tmp;
tmp = ((vp1-vp1)*b + (vp0-vp3)*a) >> 10;
out6 = tmp;
tmp = ((((-vn3)*d+vn0*c)+(vn2*e+vn1*f))-((vn3*c+vn0*d) + ((-vn2)*f+vn1*e))) >> 10;
out7 = tmp;
tmp = (PT*(((-vn3)*d+vn0*c)-(vn2*e+vn1*f))) >> 20;
out3 = tmp;
tmp = (PT*((vn3*c+vn0*d)-((-vn2)*f+vn1*e))) >> 20;
out5 = tmp;
tmp = ((((-vn3)*d+vn0*c)+(vn2*e+vn1*f))+((vn3*c+vn0*d) + ((-vn2)*f+vn1*e))) >> 10;
out1 = tmp;
```

圖十五: loeffler DCT 合併計算方式

圖十五的程式碼中可以想成是 c 語言的順序執行方法,上面算完換下面計算,計算精度從 8 次方改成 10 次方,上面計算的公式可以認為說是將版本 1 的程式碼,stage 計算全部拉在一起寫成一個給結果的等式,直接來看 adder 和 multiplier 應該是比版本 1 還要多,但是結果來看這個寫法面積和 power 卻比版本 1 好了不少,目前還不太清楚是為甚麼。

接下來使用 vsim 來看計算結果是否有誤,首先用範例資料來檢查。

<u>`</u>		Msgs									
stimulus 1ddct/o	12'd1192	(C	532	1924	(1193	1118	1203	1095	11110	11192	
stimulus 1ddct/o	12'd79	(0	1419	(511	150	(33	(34	7-17	13	79	
± ♦ /stimulus 1ddct/o	-12'd54	(0) (-57	(-6	7-81	X-16	159	17	(-47	X-54	
± ♦ /stimulus 1ddct/o	-12'd3	(C) X-179	-227	7 1-48	7-29	.30	39	.25	1-3	
stimulus 1ddct/o	12'd38	(0	196	200		1-18	37	135	44	38	
stimulus 1ddct/o	12'd50	(0	1198	62	10	38	1-12	7-13	27	50	
stimulus 1ddct/o	12'd22	(0	162	13	138	X-16	119	7-31	13	22	
stimulus 1ddct/o	12'd30	(0) \ \ \ \ \ \ \ \ \ \ \ -72	7-60	(-61	X-14	7-15	χ-8	1-45	(30	
♦ /stimulus 1ddct/rstn	-1'd1										
stimulus 1ddct/w	16'd166		137	200	156	147	166	142	137	166	
stimulus 1ddct/w	16'd137		81	166		152		137		137	
stimulus 1ddct/w	16'd166		137	176		156		117	137	166	
stimulus 1ddct/w	16'd166		157						166		
+ 🔷 /stimulus 1ddct/w	16'd156		10		137					156	
stimulus_1ddct/w	16'd137		ō	10		141		137			
stimulus 1ddct/w	16'd137		0						127	137	
+ > /stimulus 1ddct/w	16'd127		10			137				127	
 .tch1030813@mitcl .tch1030813@mitcl	h:~/ubunt h:~/ubunt	tu-fold/t	:hird_yea:	r/DIC/M							
itch1030813@mitcl itch1030813@mitcl peffer_DCT loef- itch1030813@mitcl = 0.541196 = 1.306563 = 0.831470	h:~/ubunt h:~/ubunt fer_DCT.c	tu-fold/t c loeffe	:hird_yea: er_DCT.ex	r/DIC/M e	lidterm/	exampl	e\$ ls	effer_l	ОСТ		
ttch1030813@mitcl ttch1030813@mitcl teffer_DCT loef: ttch1030813@mitcl = 0.541196 = 1.306563 = 0.831470 = 0.555570	h:~/ubunt h:~/ubunt fer_DCT.c	tu-fold/t c loeffe	:hird_yea: er_DCT.ex	r/DIC/M e	lidterm/	exampl	e\$ ls	effer_l	ост		
tch1030813@mitcl tch1030813@mitcl peffer_DCT	h:~/ubunt h:~/ubunt fer_DCT.c	tu-fold/t c loeffe	:hird_yea: er_DCT.ex	r/DIC/M e	lidterm/	exampl	e\$ ls	effer_I	ост		
tch1030813@mitcl tch1030813@mitcl peffer_DCT loef- tch1030813@mitcl e	h:~/ubunt h:~/ubunt fer_DCT.c	tu-fold/t c loeffe	:hird_yea: er_DCT.ex	r/DIC/M e	lidterm/	exampl	e\$ ls	effer_I	ост		
ttch10308130mitcl ttch10308130mitcl ttch10308130mitcl effer_DCT loef- ttch10308130mitcl = 0.541196 = 1.306563 = 0.831470 = 0.555570 = 0.980785 = 0.195090	h: ~/ubunt h: ~/ubunt fer_DCT h: ~/ubunt	tu-fold/t c loeffe tu-fold/t	:hird_yea: er_DCT.ex: :hird_yea:	r/DIC/M e r/DIC/M	lidterm/	exampl	e\$'ls'				
itch10308130mitcl ttch10308130mitcl teffer_DCT loef- itch10308130mitcl = 0.541196 = 1.306563 = 0.831479 = 0.555570 = 0.980785 = 0.195090 DCT 532.000000 924	h:~/ubunt h:~/ubunt fer_DCT.c h:~/ubunt	tu-fold/t c loeffe tu-fold/t	chird_yea er_DCT.exc chird_yea 000 1182.	r/DIC/Me e r/DIC/M	lidterm/	exampl exampl	e\$'ls' e\$./lo	000 11:	LO . 00000		
itch10308130mitcl ttch10308130mitcl teffer_DCT loef- itch10308130mitcl = 0.541196 = 1.306563 = 0.831479 = 0.555570 = 0.980785 = 0.195090 DCT 532.000000 924	h: ~/ubunt h: ~/ubunt fer_DCT h: ~/ubunt	tu-fold/t c loeffe tu-fold/t 1193.0000 150.7071	chird_yea er_DCT.exc chird_yea 000 1182.	r/DIC/M e r/DIC/M	lidterm/ lidterm/ 1203.00 34.52	exampl (exampl) 00000 1	e\$'ls'	000 11: 525 :	LO . 00000	50 79.	00000 63870
tch10308130mitcl tch10308130mitcl tch10308130mitcl effer_DCT loef- tch10308130mitcl = 0.541196 = 1.306563 = 0.831470 = 0.555570 = 0.980785 = 0.195090 DCT 532.000000 224 419.597402 511	h:~/ubunt h:~/ubunt fer_DCT.c h:~/ubunt	tu-fold/t c loeffe tu-fold/t	chird_yea er_DCT.exc chird_yea 000 1182.	r/DIC/Me e r/DIC/M	lidterm/	exampl (exampl) 00000 1	e\$'ls' e\$./lo	000 11: 525 :	LO . 00000	50 79.	
itch10308130mitcl tch10308130mitcl tch10308130mitcl effer_DCT loef- tch10308130mitcl = 0.541196 = 1.306563 = 0.831470 = 0.555570 = 0.980785 = 0.195090 DCT 532.000000 924 419.597402 511 -56.438241 -5		tu-fold/t c loeffe tu-fold/t 1193.0000 150.7071	hird_yea: er_DCT.ex: hird_yea: 000 1182. 95 33.	r/DIC/M e r/DIC/M 900000	lidterm/ lidterm/ 1203.00 34.52	exampl (exampl)00000 1 18970 .2359	e\$ ls e\$./lo 095.000 -16.663	000 11: 525 :	LO . 00000	50 79. 57 -53.	63876
ttch10308130mitcl ttch10308130mitcl teffer_DCT loef- ttch10308130mitcl = 0.541196 = 1.306563 = 0.831479 = 0.555570 = 0.980785 = 0.195090 DCT 532.000000 924 419.597402 511 -56.438241 -5-		tu-fold/t c loeffe tu-fold/t 1193.0006 150.7071 -80.3118	hird_yea: er_DCT.ex: chird_yea: 000 1182. 95 33. 661 -15. 577 -28.	r/DIC/M e r/DIC/M 900000 453284 944705	lidterm/ lidterm/ 1203.00 34.52 59.11	exampl (exampl) 00000 1 18970 .2359 13516	e\$./lo e\$./lo 095.000 -16.663 7.986	000 11: 525 : 625 -4	L0 . 00000 L3 . 88040 46 . 00820	50 79. 57 –53. 51 –2.	63876 58501
tch10308130mitcl tch10308130mitcl tch10308130mitcl tch10308130mitcl effer_DCT loef- tch10308130mitcl = 0.541196 = 1.306563 = 0.831470 = 0.555570 = 0.1980785 = 0.198090 DCT 55CT 53CT 53CT 53CT 54CT 54CT 54CT 55CT 55CT 55CT 55CT 55	h:~/ubunt h:~/ubunt fer_DCT h:~/ubunt .0000000 : .751725 .411961 .857421	tu-fold/t c loeffe tu-fold/t 1193.0000 150.7071 -80.3118 -47.9135 -5.0000	hird_yea: r_DCT.ex: hird_yea: 000 1182. 95 33. 361 -15. 377 -28.	000000 000000 053284 044705 179420 000000	lidterm/ lidterm/ 1203.06 34.52 59.11 -29.74 37.06	(exampl (exampl 00000 1 8970 2359 3516 00000	e\$./lo 095.000 -16.663 7.986 39.677 35.000	000 11.525 : 625 - 4172 - 5000 : 1	L0.00000 L3.88040 46.00820 24.16465 44.00000	50 79. 57 -53. 51 -2. 90 38.	63876 58501 22195 00006
itch10308130mitcl ttch10308130mitcl ttch10308130mitcl coeffer_DCT loef- itch10308130mitcl = 0.541196 = 1.306563 = 0.831479 = 0.555579 = 0.980785 = 0.195090 DCT 532.0000000 924 419.597402 511 -56.438241 -5 -178.533266 -226 92.0000000 920 198.084961 62		tu-fold/t c loeffe tu-fold/t 1193.0000 150.7071 -80.3118 -47.9135	hird_yea: hird_yea: hird_yea: 000 1182.	r/DIC/M e r/DIC/M 000000 453284 044705 179420	lidterm/ lidterm/ 1203.00 34.52 59.11 -29.74	exampl 000000 1 88970 22359 3516 00000 84208	e\$./lo e\$./lo 095.000 -16.663 7.986 39.677	000 11: 525 : 625 - 172 -: 000 !	L0.00000 L3.88040 46.00820 24.1646	50 79. 57 -53. 51 -2. 90 38. 93 50.	63876 58501 22195

圖十六:比較版本二程式碼計算 DCT 和 c 語言計算 DCT 精度

從圖十六模擬結果來看,誤差幾本上在正負1,看起來乘上10次方 會比乘上8次方的精度比起來好太多了,結果看來程式碼也沒有計算 上的錯誤。

接著使用 design compiler 來看合成的 time、area 和 power。

```
Raport: area
Design: loeffler 1d
Vorsion: R-2020.00
Date: Mon Apr 15 14:51:47 2024

Library(s) Used:

fsa@m_a_generic_core_ttlp8v25c (File: /home/cad/tech/CBDK018_UMC_Faraday_v1.1/CIC/SynopsysDC/db/fsa@m_a_generic_core_ttlp8v25c.db)

Number of ports: 2695
Number of nets: 5376
Number of nets: 5376
Number of cells: 2219
Number of combinational cells: 2076
Number of sequential cells: 96
Number of sequential cells: 96
Number of macros/black boxes: 0
Number of macros/black boxes: 0
Number of references: 66
Combinational area: 78048.130208
But/Inv area: 5140.295927
Noncombinational area: 6299.596436
Net Interconnect area: undefined (Wire load has zero net area)
Total cell area: 94347.726643
Total area: undefined
```

圖十七:合成 area 結果, total cell area 為 84347.726643

從圖十七結果來看,版本二程式碼 area 比版本一程式碼 area(圖十二) 少了將近 20000 cell area。

```
      clock clk (rise edge)
      15.000
      15.000

      clock network delay (ideal)
      0.500
      15.500

      clock uncertainty
      -0.500
      15.000

      out5_reg[11]/CK (DFCRBN)
      0.000
      15.000 r

      library setup time
      -0.116
      14.884

      data required time
      14.884

      data arrival time
      -14.882

      slack (MET)
      0.002
```

圖十八: time 結果, slack time 為 0.002

slack time 結果比版本一程式碼多了 0.001(圖十三)。

Power Group	Internal Power	Switching Power	Leakage Power	Total Power (%) Attrs
io pad	0.0000	0.0000	0.0000	0.0000 (0.00%)
memory	0.0000	0.0000	0.0000	0.0000 (0.00%)
black box	0.0000	0.0000	0.0000	0.0000 (0.00%)
clock_network	0.0000	0.0000	0.0000	0.0000 (0.00%)
register	0.9460	0.1075	2.6525e+04	1.0535 (9.77%)
sequential	0.0000	0.0000	0.0000	0.0000 (0.00%)
combinational	3.8194	5.9088	2.8898e+05	9.7285 (90.23%)
Total	4.7653 mW	6.0163 mW	3.1550e+05 pW	10.7820 mW

圖十九: power 結果, total power 預測為 10.7820mW

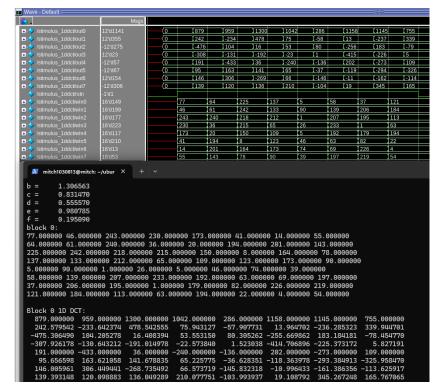
Power 結果比版本一程式碼少了約 2mW(圖十四)。 結論來說版本二合成後的效果較好,接下來更改 testbench 測試數據,然後生成 vcd 檔案給 primetime 計算 slack time 和 total power。 以下是使用 python 生成數據的程式碼:

```
import numpy as np

fp1 = open("hex1.txt", 'w')
fp2 = open("dec1.txt", 'w')
l = np.random.randint(0, 255, 6400)
for i in l:
    fp1.write(hex(i)[2:])
    fp1.write('\n')
    fp2.write('{}'.format(i))
    fp2.write('\n')
fp1.close()
fp2.close()
```

圖二十:生成 100 組 8*8 block 數據程式碼

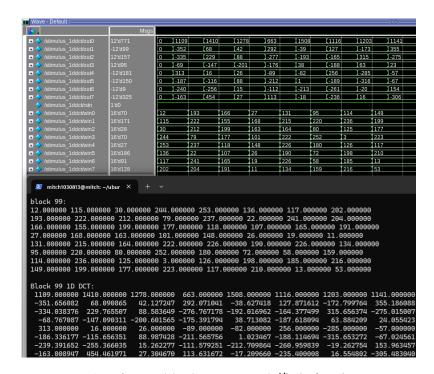
以上程式碼給定一個 0~255(unsigned char,影像像素資料型態)範圍並 隨機生成 6400 筆數據,之後會以類似 8*8 block 的方式寫入 mem 中,會有 hex 和 dec 是因為 c 語言那邊也要檢測一下計算是否正確。接著使用 vsim 加入 sdf 檔案進行時序仿真,這裡會生成 vcd 檔案並且驗證一下剛剛生成的數據計算 DCT 是否有誤。



圖二十一: block 0 DCT 計算結果比較。

/ /stimulus 1ddct/out0	12'd771	0 1745	X911	¥1400	V110F	V1200	V1100	X 1074	V1000
	-12'd99	0 1205		11493			(1123 (-235		-
/stimulus_1ddct/out1				Ĭ 187 Ĭ 72	X316 X71		X-235 X19		
/stimulus_1ddct/out2	12'd157 12'd95					-72		X-190	
/stimulus_1ddct/out3				85	-206		383	7-252	
/stimulus_1ddct/out4	-12'd181	0 (139		I-169		(-28	X 525		
/stimulus_1ddct/out5	-12'd150	0 353		I -88	X-168		8	X-185	
/stimulus_1ddct/out6	-12'd9	0 7-21		1267 127	X-60	.382	196 1-64		133 1140
/stimulus_1ddct/out7	-12'd325 1'd0	0 1-29	λ-84	121	(-136	407	J-64	(,-57	1140
/stimulus_1ddct/rstn					100	1474			1000
/stimulus_1ddct/win0	16'd70	144		233	198	174	237	46	228
/stimulus_1ddct/win1	16'd171	51		206	205	226	1	216	33
/stimulus_1ddct/win2	16'd28	132			121		7	232	49
/stimulus_1ddct/win3	16'd70	203 45		116 155	209	170 245	156	143 117	149 1191
/stimulus_1ddct/win4	16'd27				149		223		
/stimulus_1ddct/win5	16'd186	23		242	39	6	204	142	231
/stimulus_1ddct/win6	16'd91	97			13 191	243 91	87	75 103	233
4 letimulue 1 ddethain7									
≥ mitch1030813@mitch		50 + ~ 00 203.0					9000 50.		1223
## .ximulus_1ddctwin7 ## mitch1030813@mitch block 18: 1444.096096 51. 0006 60.090600 238. 000 233. 096090 265. 00 198. 096090 265. 00 237. 096090 216. 00 46. 090909 216. 00 46. 090909 213. 000	000 132.0000 000 67.00000 000 67.00000 0000 121.000 0000 123.000 000 7.000000 000 232.0000	00 203.0 0 40.000 000 116.0 000 209.0 000 170.0 156.0000 00 143.0	00000 45 000 36.00 000000 19 000000 20 00 223.00	.000000 00000 12 55.00000 49.00000 45.00000 00000 20	23.00006 0.000006 0 242.06 0 39.006 0 6.0006 4.000006	90 97.000 9 158.000 90000 13.0 9000 13.0 900 243.0 9 87.0000	0000 50. 0000 142 2.000000 000000 1 000000 9	000000 1.000000 1.58.000 91.00000 000000	9000 90 9

圖二十二: block 18 DCT 計算結果比較



圖二十三: block 99 DCT 計算結果比較

從上面三個 block 看起來應該計算是沒有太大的問題,接下來可以用 primetime 來計算 power。

clock clk (rise edge) clock network delay (propagated) clock reconvergence pessimism clock uncertainty out5_reg[11]/CK (DFCRBN) library setup time data required time	15.00 0.00 0.00 -0.50 -0.12	15.00 15.00 15.00 14.50 14.50 r 14.39 14.39
data required time data arrival time		14.39 -14.40
slack (VIOLATED)		-0.01

圖二十四: primetime 計算 slack time 為-0.01

slack time 來說以 100MHz clock 來跑會出問題。

Power Group	Internal Power	Switching Power	Leakage Power	Total Power	(%)	Attrs
clock_network register combinational sequential memory io_pad black_box	2.702e-03 0.0000 0.0000	1.909e-04 5.853e-03 0.0000 0.0000	2.666e-08 2.883e-07 0.0000 0.0000	8.555e-03 0.0000 0.0000 0.0000	(9.72%) (85.30%) (0.00%) (0.00%) (0.00%)	
Net Switching Power Cell Internal Power Cell Leakage Power		6.044e-03 3.985e-03 3.150e-07				
Total Power		0.0100	(100.00%)			
X Transition Power CAPP Estimated Glitchin	= g Power =	0.0000 0.0000				
Peak Power Peak Time		0.0140 4380				

圖二十五: primetime 計算 total power 為 10mW

Primetime 計算的 power 和 design compiler 預測的 power(圖十九)幾乎一樣。

結論來說,版本二程式碼 area 和 power 都比版本一程式碼還要好,但是 slack time 還是一樣,兩個都很差。

三、 問題與解決方法

Slack time 目前是最大的問題,第一個問題可能是我不太會寫程式, 改了兩天還改不好,但是目前時間緊迫,因此用第二個方法處理,就是 將 clock 的切換頻率降低降到 50MHz 或是其他筆 100MHz 還要小的頻 率。以下會將合成 clock 和 testbench clock 頻率降低至 50MHz,然後再 看 design compiler time report 和 primetime time report。

```
      clock clk (rise edge)
      25.000
      25.000

      clock network delay (ideal)
      0.500
      25.500

      clock uncertainty
      -0.500
      25.000

      out3_reg[11]/CK (DFCRBN)
      0.000
      25.000 r

      library setup time
      -0.107
      24.893

      data required time
      24.893

      data arrival time
      -18.116

      slack (MET)
      6.777
```

圖二十六:降低 clock 頻率, slack time 為 6.777。

可以看到降低 clock 頻率後, slcak time 變得比較正常了, 不過由於改變 頻率, power 和 area 也有跟著改變。

Power Group	Internal Power	Switching Power	Leakage Power	Total Power (%) Attrs
io_pad	0.0000	0.0000	0.0000	0.0000 (0.00%)
memory	0.0000	0.0000	0.0000	0.0000 (0.00%)
black box	0.0000	0.0000	0.0000	0.0000 (0.00%)
clock network	0.0000	0.0000	0.0000	0.0000 (0.00%)
register	0.4901	5.3719e-02	2.5703e+04	0.5439 (10.63%)
sequential	0.0000	0.0000	0.0000	0.0000 (0.00%)
combinational	1.7392	2.8329	2.7766e+05	4.5724 (89.37%)
Total	2.2293 mW	2.8866 mW	3.0336e+05 pW	5.1162 mW

圖二十七:降低 clock 頻率,預估 total power 為 5.1162mW

```
Number of ports: 2695
Number of nets: 5145
Number of cells: 1911
Number of combinational cells: 1768
Number of sequential cells: 96
Number of macros/black boxes: 0
Number of buf/inv: 453
Number of references: 59

Combinational area: 76035.758997
Buf/Inv area: 4018.492735
Noncombinational area: 6299.596436
Macro/Black Box area: 0.000000
Net Interconnect area: undefined (Wire load has zero net area)

Total cell area: 82335.355433
Total area: undefined
```

圖二十八: 降低 clock 頻率, total cell area 為 82335.35433

跟之前 clock 100MHz 相比,power 和 cell area 都有再下降,power 預估 少了一半(圖十九),cell area 少了 2000 多(圖十七),可知降低 clock 頻率 可以把面積時間跟 power 都用的比較好,但是做的速度會比較慢。 最後再丟去 primetime 做分析。

clock clk (rise edge) clock network delay (propagated) clock reconvergence pessimism clock uncertainty out3_reg[11]/CK (DFCRBN) library setup time data required time	25.00 0.00 0.00 -0.50 -0.11	25.00 25.00 25.00 24.50 24.50 r 24.39 24.39
data required time data arrival time		24.39 -17.63
slack (MET)		6.77

圖二十九: primetime 計算 slack time 為 6.77

```
| Internal | Switching | Leakage | Power | Pow
```

圖三十: primetime 計算 power 為 5.195mW

從圖二九和圖三十來看 power 和 time 都有變好,而且跟 design compiler 估計的數值差不多。

最後總結一下,降低 clock 頻率可以讓面積、power 和 slack time 變好, 缺點我看了一下 vcd 檔案,總執行時間變長了。還有寫程式碼好像計算 能寫再一起就寫一起,電腦好像比較會排上面那種寫法,雖然版本一可 讀性比較好但是 area、power 還有 time 就比較爛(這裡有點小 bug,因為 沒有重新跑過 50MHz 的 clock,但是後來跑了一下發現真的比較爛)。

四、 FINAL CODE

這段程式碼是在截止日期前一個小時半完成的,所以報告可能不完全,不過解決了版本一和版本二程式碼的問題,area、time 和 power 都有再進步,程式碼主要融合了版本一的 stage 計算方法和版本二的 blocking and non-blocking 寫法,還有將版本一的計算精度更改為乘上 2 的 10 次方,計算出來的差距為正負一。之前兩個版本的寫法是一個 clock 就要想辦法全部計算完成,現在改成需要幾個 clock 才可以把結果計算完成。總的來說可以在 clock 100MHz 下完成目標,只是會多出來大約 100ns 時間完成 100 筆 block 的計算目標。

程式碼:

```
always @(posedge clk) begin

s10 <= win0 + win7;

s17 <= win0 - win7;

s11 <= win1 + win6;

s16 <= win1 - win6;

s12 <= win2 + win5;

s15 <= win2 - win5;

s13 <= win3 + win4;

s14 <= win3 - win4;

end
```

圖三十一: stage 1 計算, non-blocking

```
assign tmp4 = (s17 * s3mc3);
assign tmp5 = (s14 * c3as3);
assign tmp7 = (s16 * s1mc1);
assign tmp8 = (s15 * c1as1);
always @(*) begin
    tmp02 = s15 + s16;
    tmp9 = (tmp02 * c1);
always @(*) begin
   tmp01 = s17 + s14;
    tmp6 = (tmp01 * c3);
always @(posedge clk) begin
   s20 <= s10 + s13;
    s23 <= s10 - s13;
    s21 <= s11 + s12;
    s22 <= s11 - s12;
    s24 <= (tmp4 + tmp6) >> 10;
    s27 <= (tmp6 - tmp5) >> 10;
    s25 <= (tmp7 + tmp9) >> 10;
    s26 <= (tmp9 - tmp8) >> 10;
```

圖三十二: stage 2 計算, non-blocking

```
assign tmp1 = (s23 * sqs1mc1);
assign tmp2 = (s22 * sqc1as1);

always @(*) begin
    tmp03 = s22 + s23;
    tmp3 = (tmp03 * sqc1);
end

always @(posedge clk) begin
    s30 <= s20 + s21;
    s31 <= s20 - s21;
    s32 <= (tmp1 + tmp3) >> 10;
    s33 <= (tmp3 - tmp2) >> 10;
    s34 <= s24 + s26;
    s36 <= s24 - s26;
    s37 <= s27 + s25;
end</pre>
```

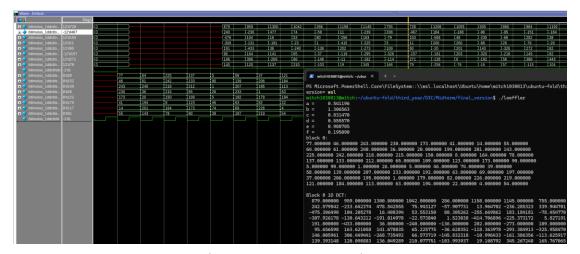
圖三十三: stage 3 計算, non-blocking

```
always @(posedge clk) begin

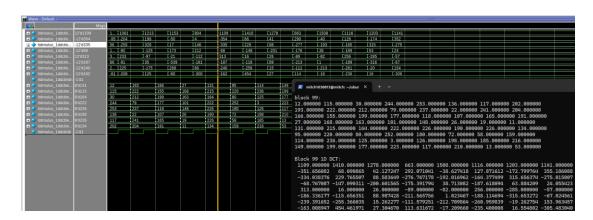
o0 <= s30;
o4 <= s31;
o2 <= s32;
o6 <= s33;
o1 <= s37 + s34;
o7 <= s37 - s34;
o3 <= (s35 * sq2) >> 10;
o5 <= (s36 * sq2) >> 10;
end
```

圖三十四:stage 4 計算, non-blocking

上方程式碼 always 中會同步好每條 reg 接收到資料,並且下個 stage 就會開始計算,直到下個 clock 到更新資料,上面會有看到 blocking 的寫法是因為在實作 ken block 的時候會需要使用到前一個計算的 wire,所以需要有順序下去做,因此用 blocking,但是想了一下好像把 tmp 的計算等式直接合併到下面 tmp 的等式就好了。最後可以知道從一開始資料輸入到最後計算完成輸出中間會間隔 5 個 posedge clk,上面有四個 clock 再包含最後輸入到 output 的一個 clock,所以最後執行時間會是 8 * 100 * 10 + 50 = block 列數量 * block 數量 * posedge clk 週期 + 5 個 posedge clk 週期 = 8050ns。以下是展示一些計算結果,由於輸入和輸出沒有太同步,因此我只 verilog計算出來的輸出,和 c 語言計算出來的輸出。



圖三十五: block 0 DCT 計算, 黃線為 block 0 計算後範圍, vsim 介面上面是輸出 DCT, 下面是輸入數據,終端機介面會顯示 C 語言計算對應的 DCT



圖三十六: block 99 DCT 計算, 黃線為 block 99 計算前範圍, vsim 介面上面是輸出 DCT, 下面是輸入數據,終端機介面會顯示 C 語言計算對應的 DCT

接著就來看一下 design compiler 合成 area、time 和 power。

<pre>clock clk (rise edge) clock network delay (ideal) clock uncertainty s27_reg[11]/CK (QDFFN) library setup time data required time</pre>	15.000 0.500 -0.500 0.000 -0.078	15.000 15.500 15.000 15.000 r 14.922 14.922
data required time data arrival time		14.922 -13.598
slack (MET)		1.324

圖三十五: clock 100MHz 下 slack time 為 1.324

```
Report: area
Design: loeffler 1d
Version: R-2020.09
Date : Mon Apr 15 22:17:38 2024

Library(s) Used:

fsa0m_a_generic_core_tt1p8v25c (File: /home/cad/tech/CBDK018_UMC_Faraday_v1.1/CIC/SynopsysDC/db/fsa0m_a_generic_core_tt1p8v25c.db)

Number of ports: 1761
Number of nets: 4369
Number of cells: 2430
Number of combinational cells: 1933
Number of combinational cells: 461
Number of sequential cells: 461
Number of baracros/black boxes: 0
Number of the combinational cells: 401
Number of baracros/black boxes: 0
Number of berferences: 49
Combinational area: 51827, 933029
But/Inv area: 9690, 068705
Noncombinational area: 25992, 086504
Narcro/Black Box area: 9, 0000000
Net Interconnect area: undefined

Total cell area: 77820.019622
Total area: undefined
```

圖三十六: clock 100MHz 下 total cell area 為 77820.019622

```
Global Operating Voltage = 1.8
Power-specific unit information :
Voltage Units = 1V
Capacitance Units = 1.000000pf
Time Units = 1ns
Dynamic Power Units = 1mW
Leakage Power Units = 1pW
                                                                (derived from V,C,T units)
   Cell Internal Power
Net Switching Power
Total Dynamic Power
                                            = 8.8237 mW (100%)
Cell Leakage Power
                                            = 299.2046 nW
                                                                  Switching
Power
                                 Internal
                                                                                                          Leakage
Power
                                                                                                                                                Total
Power
Power Group
                                                                                                                                                                                      Attrs
io_pad
                                                                                                            0.0000
to_pau
memory
black_box
clock_network
register
sequential
combinational
                                                                                                           0.0000
                                                                                                    1.1726e+05
                                                                                                                                                5.6208
                                                                        3.3100 mW
```

圖三十七: clock 100MHz 下 total power 預估為 8.8240mW

從結果來看,可知沒有一次做完 slack time、area,會比一個 clock 直接做 完還要好,上面的結果跟版本二程式碼 clock 設定在 50MHz 相比較,area 少了約 5000 cell area,雖然 slack time 沒有很好看,但是至少不用花兩倍的 時間去完成計算。

最後生成 vcd 檔案然後使用 primetime 來計算 time 和 power。

```
clock clk (rise edge)
                                              15.00
                                                          15.00
                                                          15.00
clock network delay (propagated)
                                              0.00
clock reconvergence pessimism
                                              0.00
                                                          15.00
                                              -0.50
clock uncertainty
                                                          14.50
s27_reg[11]/CK (Q̈DFFN)
                                                          14.50
library setup time
data required time
                                              -0.08
                                                          14.43
                                                          14.43
data required time
                                                          14.43
data arrival time
                                                          13.12
slack (MET)
                                                           1.31
```

圖三十七: clock 100MHz 下 primetime 計算 slack time 為 1.31

```
Internal Switching Leakage
                                                                                              Total
clock_network
                                        2.232e-03
                                                                              0.0000 2.232e-03 (99.81%)
register
                                         -1.977e-06
                                                         5.414e-06 1.265e-07 3.564e-06 ( 0.16%)
combinational
                                        -2.616e-07
                                                        7
6.364e-07 2.013e-07 5.761e-07 ( 0.03%)
0.0000 0.0000 0.0000 ( 0.00%)
0.0000 0.0000 0.0000 ( 0.00%)
0.0000 0.0000 0.0000 ( 0.00%)
0.0000 0.0000 0.0000 ( 0.00%)
                                            0.0000
0.0000
sequential
memory
io_pad
black_box
   Net Switching Power
Cell Internal Power
Cell Leakage Power
                                                   = 6.050e-06
                                                                           ( 0.27%)
(99.71%)
( 0.01%)
                                                   = 2.230e-03
= 3.278e-07
Total Power
                                                   = 2.236e-03 (100.00%)
X Transition Power =
CAPP Estimated Glitching Power =
                                                    = 3.437e-06
= 0.0000
                                                   = 5.364e-03
= 10
```

圖三十八: clock 100MHz 下 primetime 計算 total power 為 2.236mW

Information: Total simulation time = 8215.000000 ns

圖三十九:VCD 檔案模擬時長

以結果來看,primetime 計算的 power 比 clock 設定為 50MHz 的還要少約 3mW(圖三十),然後圖三十九的 vcd 檔案模擬時長也跟剛剛計算出來的模擬時長(8050ns)差不多,因為 testbench 檔案裡有設定一些#延時。總結上述的程式碼在計算精度上差距為正負一,area、time 和 power report 都比版本一和版本二的程式碼還要更好。