超大型積體電路信號處理設計期末報告

題目:

2-D 8-Point Discrete Cosine Transform

系所:光電所碩士班

學號:M1124009

姓名:林家齊

1. 電路架構與設計

Final Project – DCT Design

- **設計描述**

設計一個 2-D 8-point Discrete Cosine Transform (DCT)，1-D DCT 猶如一個矩陣運算，下面即是設計描述：

$$Z(n) = \sqrt{\frac{2}{N}} c(n) \sum_{m=0}^{N-1} x(m) \times \cos\left(\frac{(2m+1)n\pi}{2N}\right)$$

where

$$C(n) = \begin{cases} 1/\sqrt{2} & \text{for } n = 0 \\ 1 & \text{for } others \end{cases}$$

利用矩陣展開

$$Z = \begin{bmatrix} Z_0 \\ Z_1 \\ Z_2 \\ Z_3 \\ Z_4 \\ Z_5 \\ Z_6 \\ Z_7 \end{bmatrix} = \begin{bmatrix} \cos 4\theta & \cos 4\theta & \cos 4\theta & \cos 4\theta & \cos 4\theta & \cos 4\theta & \cos 4\theta & \cos 4\theta \\ \cos\theta & \cos 3\theta & \cos 5\theta & \cos 7\theta & -\cos 7\theta & -\cos 5\theta & -\cos 3\theta & -\cos\theta \\ \cos 2\theta & \cos 6\theta & -\cos 6\theta & -\cos 2\theta & -\cos 2\theta & -\cos 6\theta & \cos 6\theta & \cos 2\theta \\ \cos 3\theta & -\cos 7\theta & -\cos\theta & -\cos 5\theta & \cos 5\theta & \cos\theta & \cos 7\theta & -\cos 3\theta \\ \cos 4\theta & -\cos 4\theta & -\cos 4\theta & \cos 4\theta & \cos 4\theta & -\cos 4\theta & -\cos 4\theta & \cos 4\theta \\ \cos 5\theta & -\cos\theta & \cos 7\theta & \cos 3\theta & -\cos 3\theta & -\cos 7\theta & \cos\theta & -\cos 5\theta \\ \cos 6\theta & -\cos 2\theta & \cos 2\theta & -\cos 6\theta & -\cos 6\theta & \cos 2\theta & -\cos 2\theta & \cos 6\theta \\ \cos 7\theta & -\cos 5\theta & \cos 3\theta & -\cos\theta & \cos\theta & -\cos 3\theta & \cos 5\theta & -\cos 7\theta \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix}$$

where $\theta = \pi/16$

2-D DCT 運算為

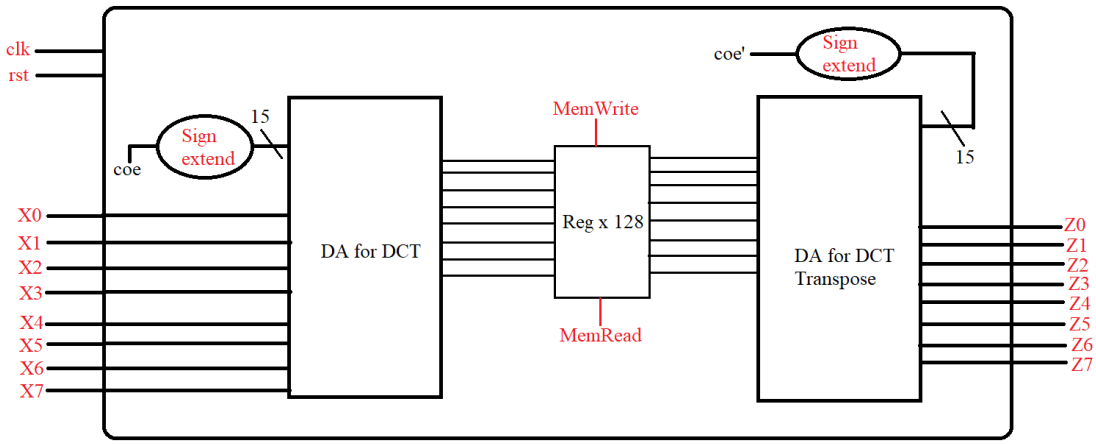
$$Y(u,v) = \frac{1}{4} C(u) C(v) \sum_{i=0}^{7} \sum_{j=0}^{7} X(i,j) \times \cos\left(\frac{(2i+1)u\pi}{16}\right) \cos\left(\frac{(2j+1)v\pi}{16}\right)$$

簡單說就是一個 8×8 的 2-D DCT 轉換硬體實現，而我的做法首先就是先將我要在電路上做的事列出，則會如下所示

```
for i = 1:w*h/64
    YB2(:,:,i) = bw*lena8x8(:,:,i)*bw';
end
```

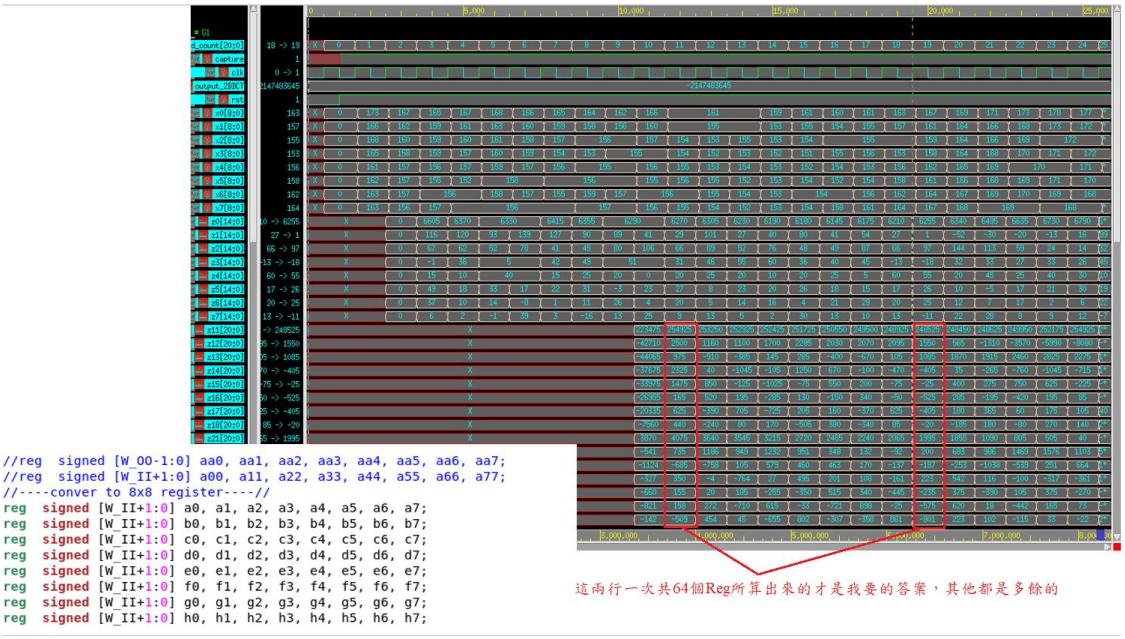
Lena8x8 為我們這次使用的圖片，則我用的 for 迴圈會把這張 256x256 的圖片切成 1024 塊，而其中每塊裡面又有 8x8 個 pixel，因此我們就可以直接將這一塊一塊總共 1024 塊的區塊拿去餵給 DCT 轉換，而在電路上我們需要做的就只有將 pixel 區塊乘以 DCT 的矩陣以及轉置矩陣而已。

在做完電路演算法的軟體模擬之後，我就開始做 Fix-Pointed，我一開始選擇將 coe 做 3 個 bit 的左移，而轉置過後的 DCT 矩陣一樣做 3bit 的左移，輸入為老師規定的 8bit + signed bit = 9bit、輸出為 16bit，那下圖是我的電路圖。



但我在將電路寫出來後發現位移的 bit 數遠遠不夠，導致我最後的 PSNR 遠遠不夠達到正 40dB 的標準，所以上面這張圖是我在 1/05 期末 Demo 之後所做的修改，我將 coe 的位移提高到 14Bit，而我在軟體上測試過後發現最後最少需要位移 coe + coe^T 共至少 28Bit 才能達到 40dB，所以事實上這是我報告完後修改的電路圖。

而在設計上我還有修改的部分是在算完第一次 DCT 之後該如何把答案的矩陣轉置成我可以一次八個輸出的形式，也就是中間 128 個 Reg 的部分，在這部分我原本用的方法是設 8 條串起來的 Reg，每條 8 個總共 64 個 Reg，分別對到八個第一層 DCT 的輸出，這樣串起來就可以讓 Reg 依照 clk 一個一個傳下去暫時保留剛剛算完的值不被洗掉，但這樣做有一個很大的缺點，就是我只能讓它一直傳下去然後再去跟 DCT 的轉置矩陣做運算，這會讓我產生很多我不需要的 data，造成我在最後寫進 txt 檔的時候要抓取我需要的答案會有困難，但真的要抓答案還是可以回到 matlab 寫一個 for 迴圈把我要的答案抓出來，但這樣就失去了我挪到硬體上做的意義。




```
//reg  signed [W_OO-1:0] aa0, aa1, aa2, aa3, aa4, aa5, aa6, aa7;
//reg  signed [W_II+1:0] a00, a11, a22, a33, a44, a55, a66, a77;
//----conver to 8x8 register----//
reg  signed [W_II+1:0] a0, a1, a2, a3, a4, a5, a6, a7;
reg  signed [W_II+1:0] b0, b1, b2, b3, b4, b5, b6, b7;
reg  signed [W_II+1:0] c0, c1, c2, c3, c4, c5, c6, c7;
reg  signed [W_II+1:0] d0, d1, d2, d3, d4, d5, d6, d7;
reg  signed [W_II+1:0] e0, e1, e2, e3, e4, e5, e6, e7;
reg  signed [W_II+1:0] f0, f1, f2, f3, f4, f5, f6, f7;
reg  signed [W_II+1:0] g0, g1, g2, g3, g4, g5, g6, g7;
reg  signed [W_II+1:0] h0, h1, h2, h3, h4, h5, h6, h7;
```

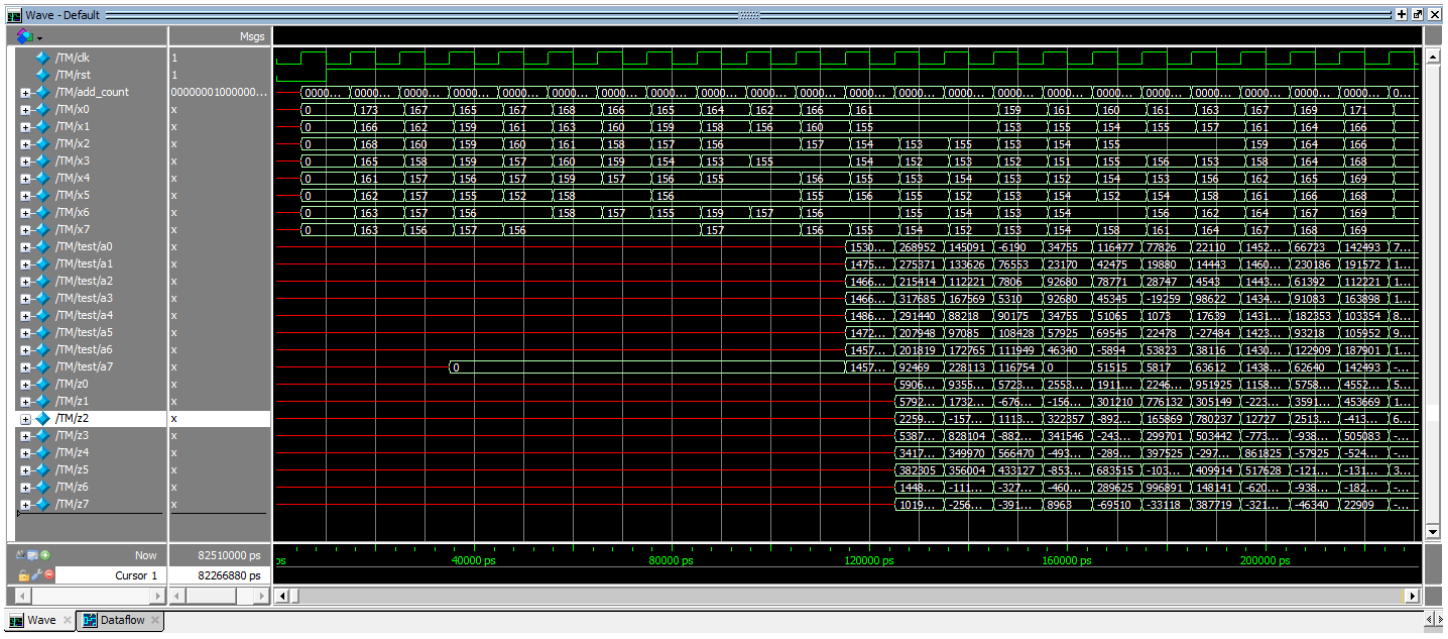
▲原始設計

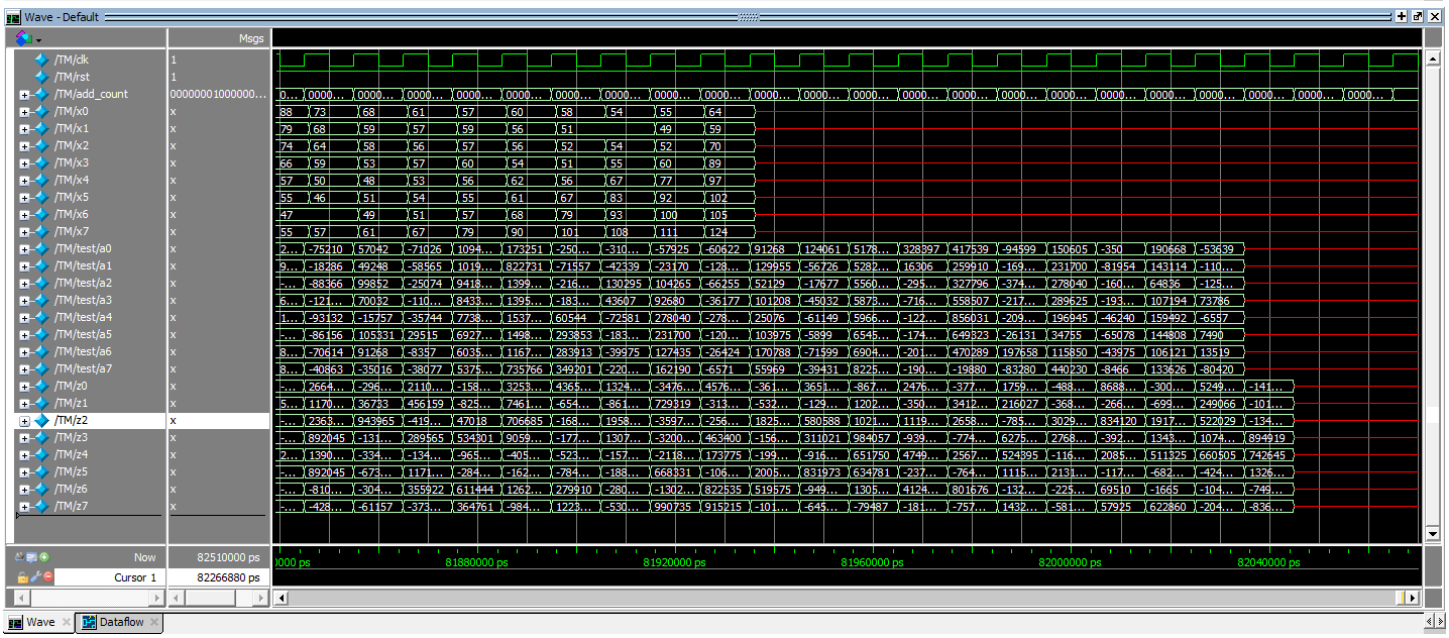
因此雖然此方法可行但我還是決定在報告完後把它修改掉，修改的方法就是我用兩倍的 Reg 量然後配合電路中我自己設的計數器來做資料的暫存，實際的 Vcode 如下兩圖所示

```
144     //set a trigger in DCT circuit
145     always@(posedge clk or negedge rst)begin
146         if(!rst)
147                 trg_in <= 2'd0;
148         else begin
149             if(trg_in == 2'd1)
150                 trg_in <= 2'd1;
151             else
152                 trg_in <= trg_in + 1;
153         end
154     end
155
156     always@(posedge clk or negedge rst)begin
157         if(!rst)
158                 mem_count <= 0;
159         else begin
160                 if(trg_in == 2'd1)                  //when trg=3 then mem_cunt will work
161                     mem_count <= mem_count + 1;
162         end
163     end
164
165
166     //----assign mem_reg[n] for DCT1 result----//
167     always@(posedge clk)begin
168             if(mem_count == 4'd0)begin
169                 memory[16*8-8] <= zz0;
170                 memory[16*8-7] <= zz1;
171                 memory[16*8-6] <= zz2;
172                 memory[16*8-5] <= zz3;
173                 memory[16*8-4] <= zz4;
174                 memory[16*8-3] <= zz5;
175                 memory[16*8-2] <= zz6;
176                 memory[16*8-1] <= zz7;
177             end
178             else begin
179                 if(mem_count >= 4'd9)begin
180                     memory[(mem_count-8)*16-8] <= zz0; //8_24_40..._120
181                     memory[(mem_count-8)*16-7] <= zz1; //9_25_41..._121
182                     memory[(mem_count-8)*16-6] <= zz2; //10_26_42..._122
183                     memory[(mem_count-8)*16-5] <= zz3; //11_27_43..._123
184                     memory[(mem_count-8)*16-4] <= zz4; //12_28_44..._124
185                     memory[(mem_count-8)*16-3] <= zz5; //13_29_45..._125
186                     memory[(mem_count-8)*16-2] <= zz6; //14_30_46..._126
187                     memory[(mem_count-8)*16-1] <= zz7; //15_31_47..._127
188                 end
189                 else begin
190                     memory[mem_count*16-16] <= zz0;     //0_16_32..._112
191                     memory[mem_count*16-15] <= zz1;     //1_17_33..._113
192                     memory[mem_count*16-14] <= zz2;     //2_18_34..._114
193                     memory[mem_count*16-13] <= zz3;     //3_19_35..._115
194                     memory[mem_count*16-12] <= zz4;     //4_20_36..._116
195                     memory[mem_count*16-11] <= zz5;     //5_21_37..._117
196                     memory[mem_count*16-10] <= zz6;     //6_22_38..._118
197                     memory[mem_count*16-9] <= zz7;      //7_23_39..._119
198                 end
199             end
200     end
```

```
206     //----output from 8x8 convert register----//
207     always@(posedge clk or negedge rst)begin
208         if(!rst)
209                 trg_o <= 4'd0;
210         else begin
211             if(trg_o == 4'd9)
212                 trg_o <= 4'd9;
213             else
214                 trg_o <= trg_o + 1;
215         end
216     end
217
218     always@(posedge clk or negedge rst)begin
219         if(!rst)
220                 readmem_count <= -1;
221         else begin
222                 if(trg_o == 4'd9)                 //when trg_o=11 then mem_cunt will work
223                     readmem_count <= readmem_count + 1;
224         end
225     end
226
227     always@(posedge clk)begin
228     a0 <= memory[(readmem_count)];
229     a1 <= memory[(readmem_count + 16)];
230     a2 <= memory[(readmem_count + 32)];
231     a3 <= memory[(readmem_count + 48)];
232     a4 <= memory[(readmem_count + 64)];
233     a5 <= memory[(readmem_count + 80)];
234     a6 <= memory[(readmem_count + 96)];
235     a7 <= memory[(readmem_count + 112)];
236     end
237     //-----------------------------------------------------------------
```

這樣設計出來就會是乾淨的 8 進 8 出波形，也不會產生我不需要的答案。





2. 電路 performance

原始設計上我的第二層 DA 全部分開來算，因此最後 Output 會有 64 個，也就是一次 64 個 pixel 的量，這樣我第二層的 DA 架構就需要分 8 次去計算，而且是一次算完所以無法共用加法器，如下圖所示，這會讓我的面積變得很大。

```
//-----------------------------------------
//----Transpose_DA_Left(1,12345678)----//
assign ts0 = 0;
assign ts1 = a3;
assign ts2 = a2;
assign ts3 = a2 + a3;  //x2 + x3

assign ts4 = a1;
assign ts5 = a1 + a3;  //x1 + x3
assign ts6 = a1 + a2;  //x1 + x2
assign ts7 = a1 + ts3;   //x1 + x2 + x3

assign ts8 = a0;
assign ts9 = a0 + a3;
assign ts10 = a0 + a2;
assign ts11 = a0 + ts3;  //x0 + x2 + x3

assign ts12 = a0 + a1;
assign ts13 = a0 + ts5;  //x0 + x1 + x3
assign ts14 = a0 + ts6;  //x0 + x1 + x2
assign ts15 = ts12 + ts3;

//----Transpose_DA_Right(1,12345678)----//
assign tss0 = 0;
assign tss1 = a7;
assign tss2 = a6;
assign tss3 = a6 + a7;

assign tss4 = a5;
assign tss5 = a5 + a7;
assign tss6 = a5 + a6;
assign tss6 = a5 + a6;
assign tss7 = a5 + tss3;

assign tss8 = a4;
assign tss9 = a4 + a7;
assign tss10 = a4 + a6;
assign tss11 = a4 + tss3;

assign tss12 = a4 + a5;
assign tss13 = a4 + tss5;
assign tss14 = a4 + tss6;
assign tss15 = tss12 + tss3;

//--------Add--------//
assign u11 = (ts15<<2) + ts15 + (tss15<<2) + tss15 + a7*0;
assign u12 = -((ts14<<2) + (ts12<<1) + ts9 + (-tss15<<3) + (tss12<<2) + (tss10<<1) + tss9 + a7*0);
assign u13 = (-ts3<<3) + (ts10<<2) + (ts12<<1) + ts15 + (-tss12<<3) + (tss5<<2) + (tss3<<1) + tss15 + a7*0;
assign u14 = -((-ts7<<3) + (ts13<<2) + (ts12<<1) + ts6 + (-tss1<<3) + (tss12<<2) + (tss5<<1) + tss6 + a7*0);
assign u15 = (-ts6<<3) + (ts9<<2) + (ts6<<1) + ts15 + (-tss6<<3) + (tss9<<2) + (tss6<<1) + tss15 + a7*0;
assign u16 = -((-ts4<<3) + (ts9<<2) + (ts1<<1) + ts6 + (-tss13<<3) + (tss7<<2) + (tss14<<1) + tss6 + a7*0);
assign u17 = (-ts5<<3) + (ts3<<2) + (ts10<<1) + ts15 + (-tss10<<3) + (tss12<<2) + (tss5<<1) + tss15 + a7*0;
assign u18 = -((-ts5<<3) + (ts6<<2) + (ts2<<1) + ts9 + (-tss5<<3) + (tss11<<2) + (tss13<<1) + tss9 + a7*0);

always@(posedge clk)begin
z11 <= u11;
z12 <= u12;
z13 <= u13;
z14 <= u14;
z15 <= u15;
z16 <= u16;
z17 <= u17;
z18 <= u18;
end
//-----------------------------------------------------
```

Verilog ▾ Tab Width: 8 ▾ Ln 289, Col 26 ▾ INS

下面是我原始電路設計的 performance

```
****************************************
Report : timing
         -path full
         -delay max
         -max_paths 1
Design : DCT
Version: S-2021.06-SP1
Date   : Thu Jan  5 12:52:23 2023
****************************************

 # A fanout number of 1000 was used for high fanout net computations.

Operating Conditions: slow   Library: slow
Wire Load Model Mode: top

  Startpoint: e7_reg[2] (rising edge-triggered flip-flop clocked by clk)
  Endpoint: z53_reg[20]
            (rising edge-triggered flip-flop clocked by clk)
  Path Group: clk
  Path Type: max

  Des/Clust/Port     Wire Load Model       Library
  ------------------------------------------------
  DCT                tsmc13_wl10           slow
```

Report_Timing

	0.00	10.54 f
z53_reg[20]/D (DFFQX1)	0.00	10.54 f
data arrival time		10.54
clock clk (rise edge)	9.00	9.00
clock network delay (ideal)	2.00	11.00
clock uncertainty	-0.20	10.80
z53_reg[20]/CK (DFFQX1)	0.00	10.80 r
library setup time	-0.26	10.54
data required time		10.54
data required time		10.54
data arrival time		-10.54
slack (MET)		0.00

Report_Power

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	7.7390	3.0290	6.0874e+07	10.8289	(21.44%)	
sequential	0.0000	0.0000	0.0000	0.0000	(0.00%)	
combinational	20.7664	18.1278	7.9438e+08	39.6886	(78.56%)	
Total	28.5054 mW	21.1568 mW	8.5526e+08 pW	50.5175 mW		

```
****************************************
Report : area
Design : DCT
Version: S-2021.06-SP1
Date   : Thu Jan  5 12:52:24 2023
****************************************

Library(s) Used:

    slow (File: /home/Course/VLSIDSP/tech/db/slow.db)

Number of ports:                        38142
Number of nets:                         70059
Number of cells:                        26477
Number of combinational cells:          22249
Number of sequential cells:              2624
Number of macros/black boxes:               0
Number of buf/inv:                       4458
Number of references:                     692

Combinational area:             639520.914768
Buf/Inv area:                    19750.946208
Noncombinational area:           73881.033678
Macro/Black Box area:                0.000000
Net Interconnect area:         2697347.082977

Total cell area:                 713401.948446
Total area:                     3410749.031423
```

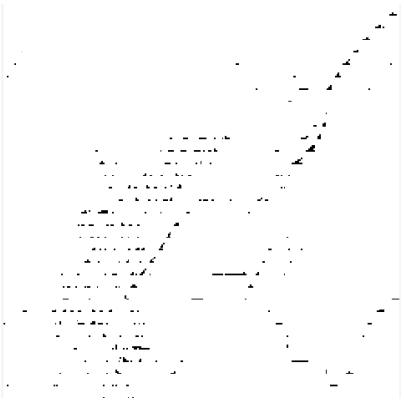
Report_Area

修改過後的設計本來想進合成看看面積多少，但是工作站好像有問題一直當機，所以就來不及上去合成了。

3. 還原後的 PSNR

A. 原始設計

```
Command Window
==================================
Original mode
PSNR = + 260.86 dB
MSE =  8.2082e-27
PSNR = + -110.63 dB
MSE =  115545727999.888
---------------------------------
fx >>
```



B. 修改後設計

```
Command Window
==========================
Original mode
PSNR = + 260.83 db
MSE =  8.2685e-27
------------------------
Fixed-point mode
PSNR = + 44.99 db
MSE =  3.1666e-05
fx >>
```

4. 心得

這次的期末 project 真的沒有多少時間做，如果可以跟期中的 1-D DCT 對調時間就好了，因為我看有成功達標 40dB 的也只有一個人而已，其他 3 個有做出來的人都沒有到 40dB，可能也是 coefficient 的 bit 數移的不夠多吧，但儘管如此也算是有做出來，我就是沒做出來的其中一個，硬體的值有算對，但就是 coe 的 bit 數真的太少是最主要的原因，再來就是中間要如何儲存和讀出 data 也花了我很多時間在想和 debug，但我想這些都不是藉口因為這個電路並不難，這表示我還有非常多要學的地方。在 Demo 完後我又花了一些時間進行架構的修改，就是前面提到的增加 bit 數提高精確度還有重新設計暫存器存資料的方法，本來我是想寫寫看 SRAM 的，但是工作站似乎沒辦法用 memory compiler，最後要感謝老同學兼實驗室夥伴靖天，我在這門課中有什麼不懂的幾乎都是和他討論才有這些成果，也感謝老師辛苦的教我們這門課。