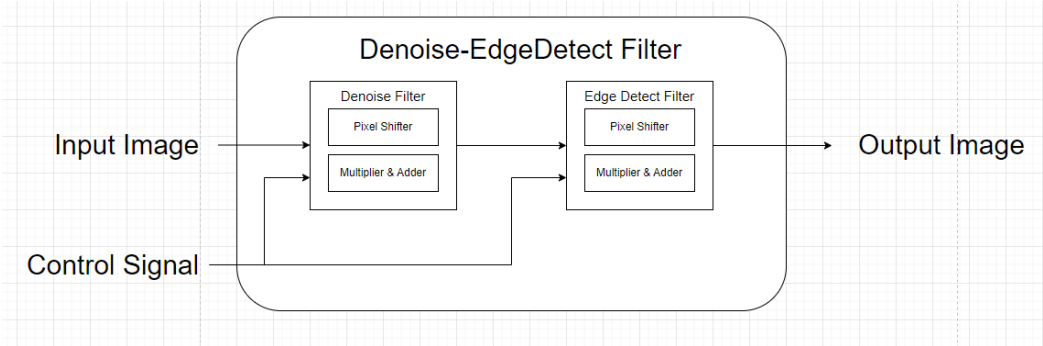
Advanced SoC Final Project

Topic: Denoise-EdgeDetect Filter

組別: 第四組　　組員: 周聖平、蔡以心、張煒侖、李承澔

一、Catapult HLS

1. System Block

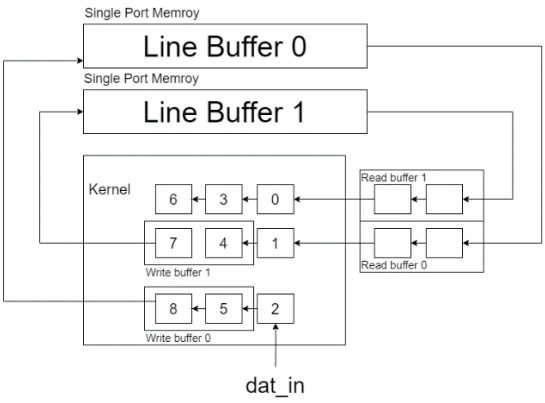


Denoise:　　Bypass、Gaussian Filter、Medium Filter

EdgeDetect:　　Bypass、Laplacian Filter、Sobel Filter

Pixel Shifter:　　Shifts the input image pixel

Pixel Shifter Architecture:



Pixel Shifter have two Line Buffers serving as temporary storage containers for pixels. These buffers will output to the Kernel at the appropriate time during processing.

Multiplier & Adder: Performs mathematical operations to apply the kernel to the image pixel

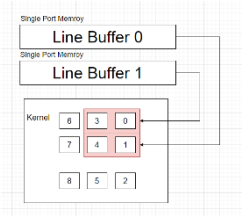
2. Process of pixel shifter

Step 1. Line buffer access

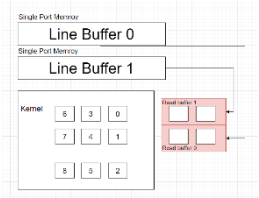
```
// LineBuffer Access
// first col
if(x==0){
  rdbuf0_pix = line_buf0[x/2];
  rdbuf1_pix = line_buf1[x/2];

  pix[4] = rdbuf0_pix.slc<8>(0);
  pix[1] = rdbuf0_pix.slc<8>(8);

  pix[3] = rdbuf1_pix.slc<8>(0);
  pix[0] = rdbuf1_pix.slc<8>(8);
}else{
  if ( (x&1)==1 ) {
  // ReadLineBuffer in x=1, x=3, ... x=odd
    rdbuf0_pix = line_buf0[(x+1)/2];
    rdbuf1_pix = line_buf1[(x+1)/2];
  } else {
  // WriteLineBuffer in x=2, x=4, ... x=even
    if(y==0){    // first row => write into buf1
      line_buf0[(x/2)-1] = wrbuf0_pix;
    }else{
      line_buf0[(x/2)-1] = wrbuf0_pix;
      line_buf1[(x/2)-1] = wrbuf1_pix;
    }
  }
}
```

In each cycle, we need to load from or write back to the Line Buffer. However, since the Line Buffer uses single-port memory and cannot read and write simultaneously, we therefore need to alternate the read and write operations as shown below.

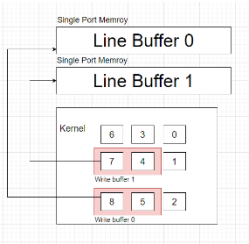
X=0 > Load Line buffer into kernel



X=1, 3, ..., odd > Load Line buffer into read buffer



X=2, 4, ..., even > Write back to line buffer



Step 2. Load pix0, pix1, pix2

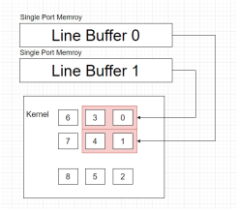
```
// Read Pixel // For uint_pix[0]
pix[0] =  (x==0 || x==1)?  pix[0]:
          ((x&1)==0)?      rdbuf1_pix.slc<8>(0):   // even 2, 4, 6, ...
                           rdbuf1_pix.slc<8>(8);   // odd  3, 5, 7, ...

pix[1] = (x==0 || x==1)? pix[1]:
         ((x&1)==0)?      rdbuf0_pix.slc<8>(0):    // even 2, 4, 6, ...
                          rdbuf0_pix.slc<8>(8);    // odd  3, 5, 7, ...


// Read dat_in into pix[2]
if (y <= heightIn-1 && x <= widthIn-1) {
  pix[2] = dat_in.read(); // Read streaming interface
}else{
  pix[2] = pix[2];
}
```

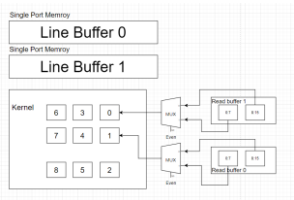
X = 0 or 1> Remain the same

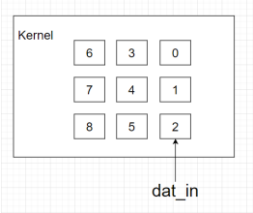
X = Even > load Read_buffer[0:7]

X = Odd > load Read_buffer[8:15]

Load dat_in into pix2

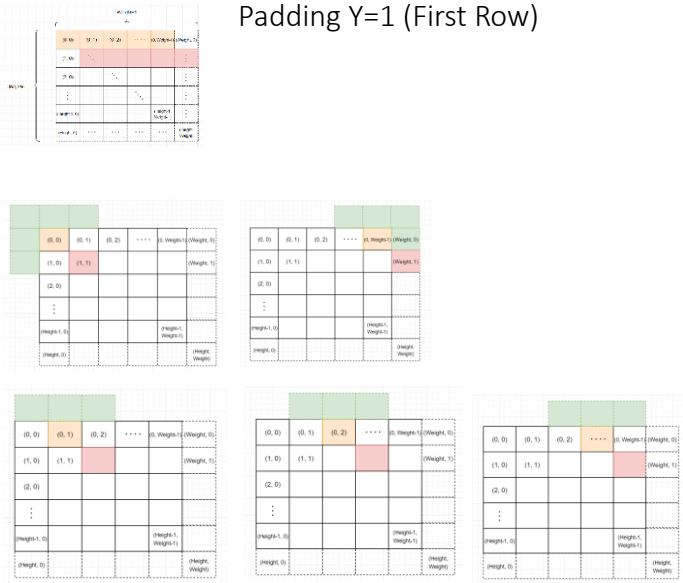






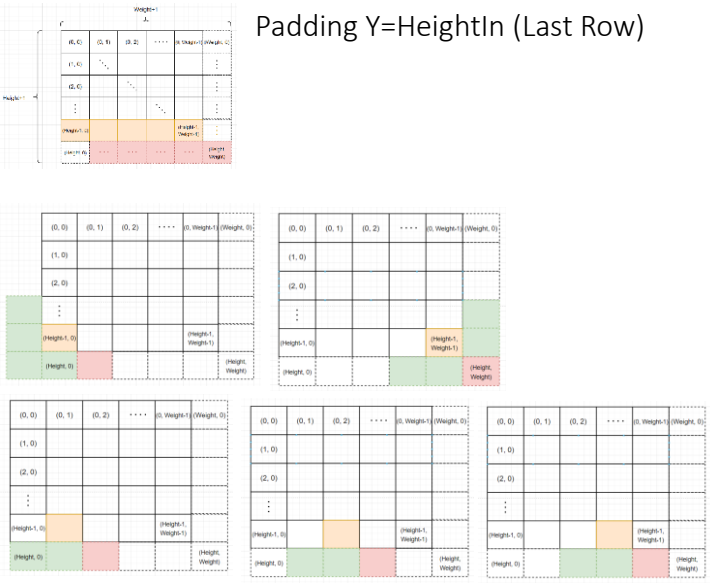
Step3. Padding

```
if (y == 1) {
    // Top edge
    if (x == 1) {
        // Top-left corner
        pix[7] = pix[4];
        pix[8] = pix[5];
    } else if (x == widthIn) {
        // Top-right corner
        pix[2] = pix[5];
        pix[1] = pix[4];
    }
    pix[0] = pix[1];
    pix[3] = pix[4];
    pix[6] = pix[7];
} else if (y == heightIn) {
```



Padding Y=1 (First Row)

```
} else if (y == heightIn) {
    // Bottom edge
    if (x == 1) {
        // Bottom-left corner
        pix[6] = pix[3];
        pix[7] = pix[4];
    } else if (x == widthIn) {
        // Bottom-right corner
        pix[0] = pix[3];
        pix[1] = pix[4];
    }
    pix[2] = pix[1];
    pix[5] = pix[4];
    pix[8] = pix[7];
} else {
```



Padding Y=HeightIn (Last Row)

```
} else {
    if (x == 1) {
        // Left edge
        pix[6] = pix[3];
        pix[7] = pix[4];
        pix[8] = pix[5];
    } else if (x == widthIn) {
        // Right edge
        pix[0] = pix[3];
        pix[1] = pix[4];
        pix[2] = pix[5];
    }
}
```

Padding X=1 & X=WidthIn
(Both side)

Step4. Multiplier & Adder – Calculate

Gaussian Filter

```
}else if(ctrl_signal == 1){
  // Gaussian Filter
  pix_result = 0;
  for(i=0; i<9; i++){
    pix_multi_float = pix[i] * Gaussian_kernel[i];
    pix_result += pix_multi_float / 16;
  }
}else if(ctrl_signal == 2){
```

Laplacian Filter

```
}else if(ctrl_signal == 1){
  // Laplacian Filter
  pix_result = 0;
  for(i=0; i<9; i++){
    pix_int = pix[i];
    pix_result += pix_int * Laplacian_kernel[i];
  }
}else if(ctrl_signal == 2){
```

Sobel Filter

```
}else if(ctrl_signal == 2){
  // Sobel Filter
  pix_result_dx = 0;
  for(i=0; i<9; i++){
    pix_int = pix[i];
    pix_result_dx += pix_int * Sobeldx_kernel[i];
  }
  pix_result_dy = 0;
  for(i=0; i<9; i++){
    pix_int = pix[i];
    pix_result_dy += pix_int * Sobeldy_kernel[i];
  }
  //printf(" ======== dy_buf=%2d ========= ",dy_buf.to
  pix_result = pix_result_dx + pix_result_dy;
  //printf(" ======== pix_resul=%2d ========= \n",pix_
}
```

Use one multiplier and one adder to finish the calculation

Medium Filter

```
}else if(ctrl_signal == 2){
  // Medium Filter
  for(i=0; i<9; i++){
    tmp_pix[i] = pix[i];
  }

  pix_result = 0;
  for(j=0; j<5; j++){
    max_val = tmp_pix[j];
    max_i = j;
    for(i=j+1; i<9; i++){
      if(tmp_pix[i] > max_val){
        max_val = tmp_pix[i];
        max_i = i;
      }
    }
    tmp_pix[max_i] = tmp_pix[j];
  }
  pix_result = max_val;
}
```

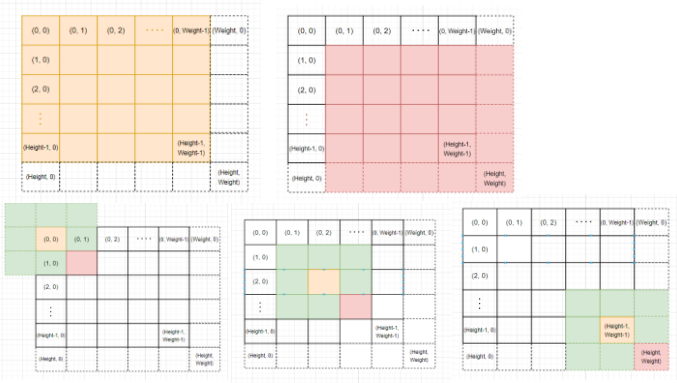
1. Starting from the jth position, find the maximum value between positions j+1 and END.

2. Once the maximum value is found, swap its position with the jth position to ensure that the values from j+1 to END are less than the value at the jth position (since the values from 0 to jth are not used, there is no need to assign a new value to jth).

3. After five iterations, the maximum value will be the median of the array.

Step 5. Output

```
// Output dat_out
if (y!=0 && x!=0) {
  // Saturation
  //printf(" ======== pix_result =========
  //printf(" ======== x=%u, y=%u =========
  //printf(" ======== pix_result_sat=%2d, p

  if(pix_result < 0){
    pix_result_sat = 0;
  }else if(pix_result > 255){
    pix_result_sat = 255;
  }else{
    pix_result_sat = pix_result.slc<8>(0);
  }
  dat_out.write(pix_result_sat);
}
```

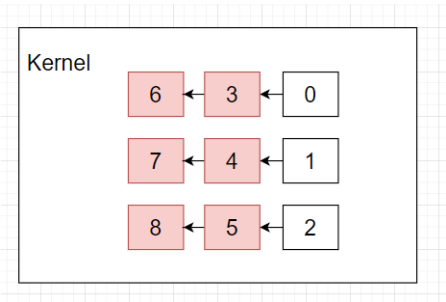


Green: 3X3 Kernel; Orange: Processing Pixel

Red: For loop idx (x, y)

Step6. Shift the pixel

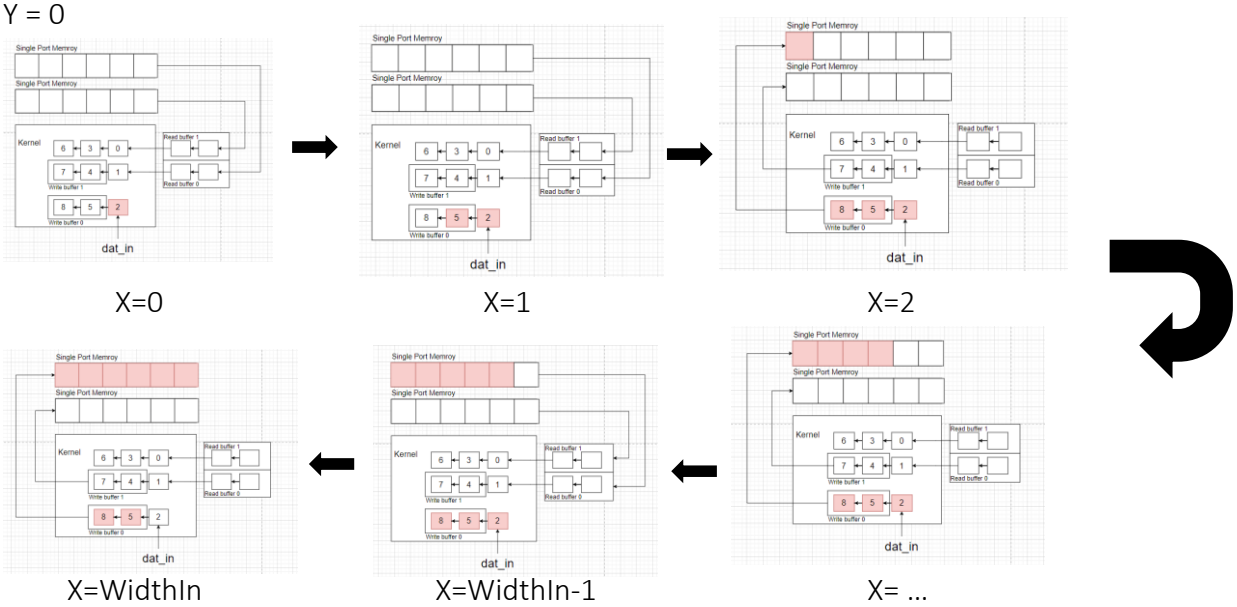
```
// Pixel Shift
pix[6] = pix[3]; pix[7] = pix[4]; pix[8] = pix[5];
if(x==0){
  // maintain
  pix[3] = pix[3]; pix[4] = pix[4];
}else{
  // shift
  pix[3] = pix[0]; pix[4] = pix[1];
}
pix[5] = pix[2];
```



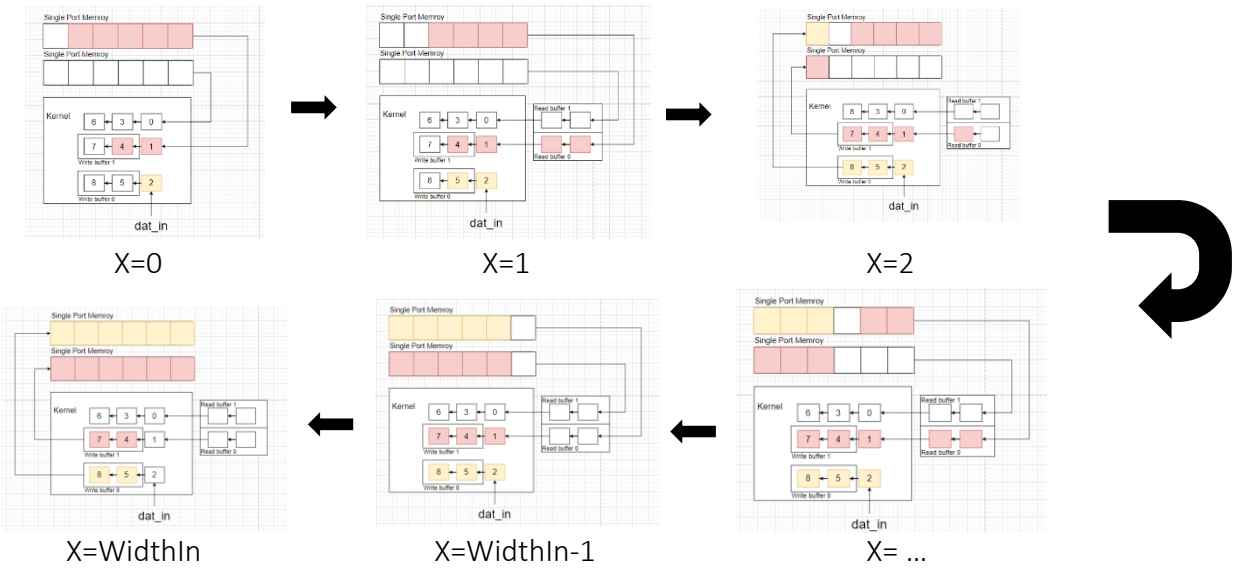
Step 7. Repeat Steps 1–6 until the for loop is complete.

3. Visualization of process

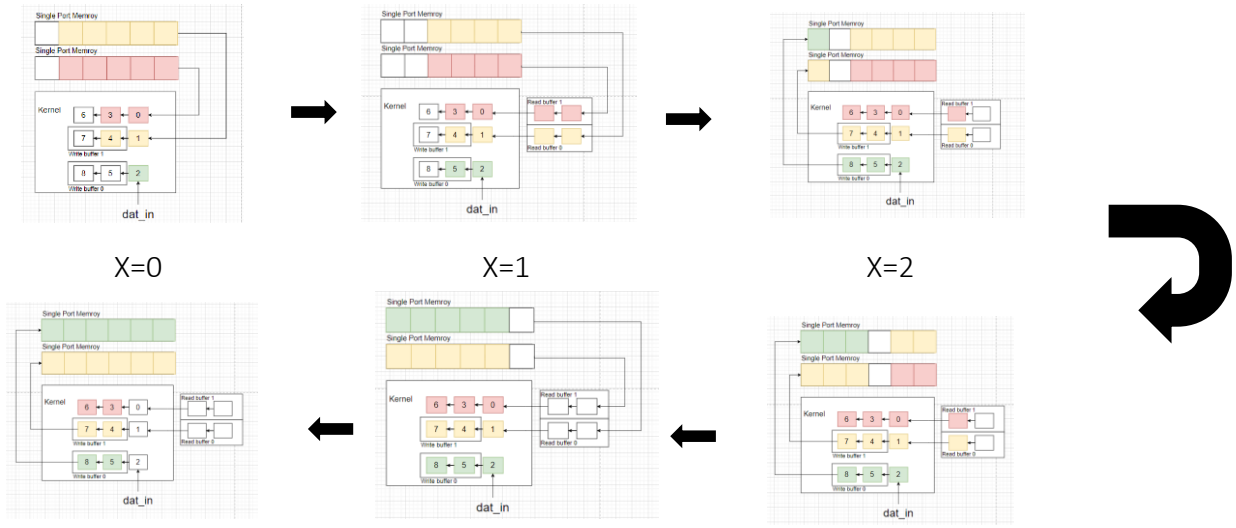
Y = 0



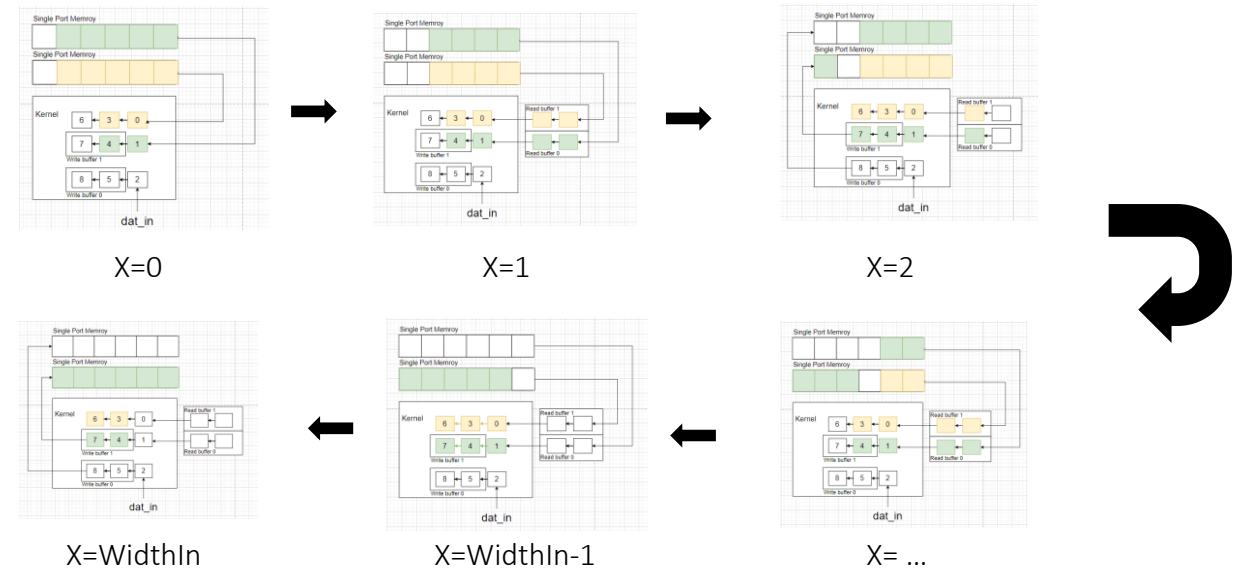
Y = 1



Y = 2



Y = HeightIn



4. Result

Image with Gaussian noise

Gaussian Filter

Medium Filter

Gaussian Filter 的圖片看起來相對平滑，噪聲殘留較少，細節和邊緣保持得較好。

Medium Filter 的圖片仍有一些殘留噪聲，邊緣也略顯模糊。

Image with SaltPepper noise

Gaussian Filter

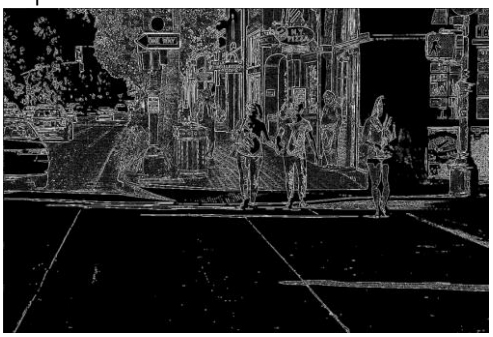
Medium Filter

Gaussian Filter 的圖片還殘留許多 SaltPepper 噪音，看起來較為模糊。

Medium Filter 的圖片較為乾淨，且將大部分的 SaltPepper 噪音給去除掉。

Image with large edge variation

Laplacian Filter



Sobel Filter

Laplacian Filter 很多區域都檢測出了不必要的邊緣，特別是在平坦或均勻的區域，整體來看，它的邊緣較不明顯。

Sobel Filter 在細節保留方面表現良好，相對 Laplacian 來說，人物和建築物的輪廓較為明顯。

Image with small edge variation

Laplacian Filter

Sobel Filter

Laplacian Filter 的邊緣較為清晰且銳利，對於建築物和人物的輪廓上有不錯的效果。

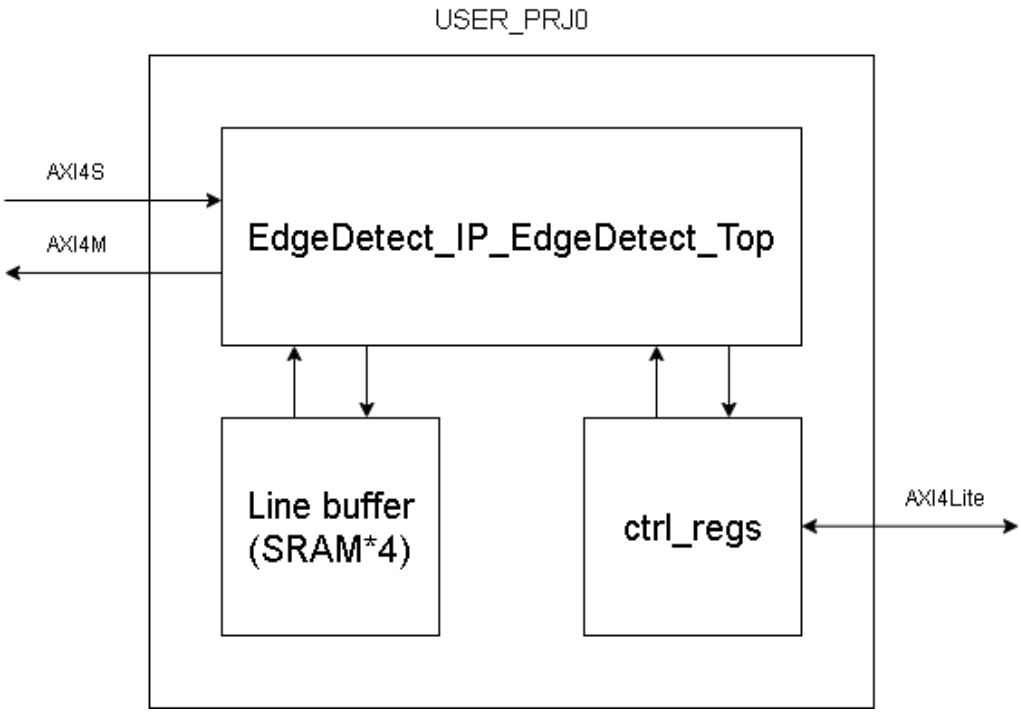
Sobel Filter 的某些邊緣存在斷裂的現象，相較於 Laplacian filter 來說，較難以識別建築物和人物的輪廓

5. Synthesis Report

Solution	Latency Cycles	Latency Time	Throughput Cycles	Throughput Time	Slack	Total Area
EdgeDetect_IP::EdgeDetect_Top.v19 (extract)	12	120.00	14	140.00	3.92	10246.53

二、Integration with FSIC

1. user_prj block diagram



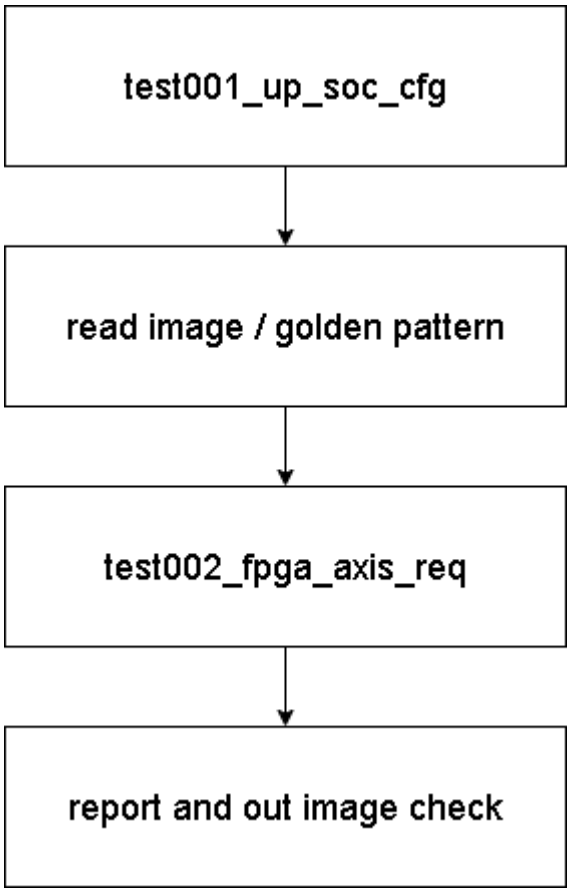
我們將 HLS 合成的.v 檔放入 user project0，並加入 4 個 SRAM，因為總共有 2 個 filter，每個 filter 都需要 2 個 line buffer。此外，我們還需要加入 ctrl_reg 對我們的 Edge_dect IP 進行配置。

ctrl_reg 是透過 AXI4Lite 進行傳輸，而 Edge_dect 透過 AXI4S 接收輸入、透過 AXI4M 傳送輸出。

Register Map

Register	WordOffset	StartBit	Length	ResetVal	Access	HADDRS
reg_rst	0	0	[0:0]	0	R/W	00
reg_widthIn	1	0	[10:0]	640	R/W	04
reg_heightIn	2	0	[9:0]	480	R/W	08
reg_ctrl_denoise	3	0	[1:0]	0	R/W	0C
reg_ctrl_edgedect	4	0	[1:0]	0	R/W	10
reg_IP_done	5	0	[0:0]	0	R/W	14

2. Caravel-FSIC FPGA Simulation (without userdma)



FSIC 的驗證流程如上圖，首先 test001_up_soc_cfg 會對 control register 進行讀寫，依序進行 reset、寫 width、heigjt、ctrl_denoise、ctrl_edgedect，然後讀取輸入圖片的 hex 檔和在 c++ simulation 時產生的 golden pattern，test002_fpga_axis_req 就是透過 AXIS 傳送輸入圖片或透過 AXIM 接收 Edge_dect IP 產生的輸出，最後將接收到的輸出與先前讀取的 golden pattern 進行比較。

下圖為 test001_up_soc_cfg 的 log

```
test001_up_soc_cfg: soc cfg read/write test
                1665=> soc_up_cfg_write : wbs_adr=30000000, wbs_sel=0001, wbs_wdata=00000001
                1945=> soc_up_cfg_read : wbs_adr=30000000, wbs_sel=0001
                1945=> soc wishbone read data result : send soc_cfg_read_event
                1945=> soc_up_cfg_read : got soc_cfg_read_event
                1945=> test001_up_soc_cfg [PASS] cfg_read_data_expect_value=00000001, cfg_read_data_captured=00000001
-----------------
                2185=> soc_up_cfg_write : wbs_adr=30000000, wbs_sel=0001, wbs_wdata=00000000
                2465=> soc_up_cfg_read : wbs_adr=30000000, wbs_sel=0001
                2465=> soc wishbone read data result : send soc_cfg_read_event
                2465=> soc_up_cfg_read : got soc_cfg_read_event
                2465=> test001_up_soc_cfg [PASS] cfg_read_data_expect_value=00000000, cfg_read_data_captured=00000000
-----------------
                2705=> soc_up_cfg_write : wbs_adr=30000004, wbs_sel=0111, wbs_wdata=00000080
                2985=> soc_up_cfg_read : wbs_adr=30000004, wbs_sel=0111
                2985=> soc wishbone read data result : send soc_cfg_read_event
                2985=> soc_up_cfg_read : got soc_cfg_read_event
                2985=> test001_up_soc_cfg [PASS] cfg_read_data_expect_value=00000080, cfg_read_data_captured=00000080
-----------------
                3225=> soc_up_cfg_write : wbs_adr=30000008, wbs_sel=0111, wbs_wdata=00000080
                3505=> soc_up_cfg_read : wbs_adr=30000008, wbs_sel=0111
                3505=> soc wishbone read data result : send soc_cfg_read_event
                3505=> soc_up_cfg_read : got soc_cfg_read_event
                3505=> test001_up_soc_cfg [PASS] cfg_read_data_expect_value=00000080, cfg_read_data_captured=00000080
-----------------
                3745=> soc_up_cfg_write : wbs_adr=3000000c, wbs_sel=0001, wbs_wdata=00000000
                4025=> soc_up_cfg_read : wbs_adr=3000000c, wbs_sel=0001
                4025=> soc wishbone read data result : send soc_cfg_read_event
                4025=> soc_up_cfg_read : got soc_cfg_read_event
                4025=> test001_up_soc_cfg [PASS] cfg_read_data_expect_value=00000000, cfg_read_data_captured=00000000
-----------------
                4265=> soc_up_cfg_write : wbs_adr=30000010, wbs_sel=0001, wbs_wdata=00000002
                4545=> soc_up_cfg_read : wbs_adr=30000010, wbs_sel=0001
                4545=> soc wishbone read data result : send soc_cfg_read_event
                4545=> soc_up_cfg_read : got soc_cfg_read_event
                4545=> test001_up_soc_cfg [PASS] cfg_read_data_expect_value=00000002, cfg_read_data_captured=00000002
```

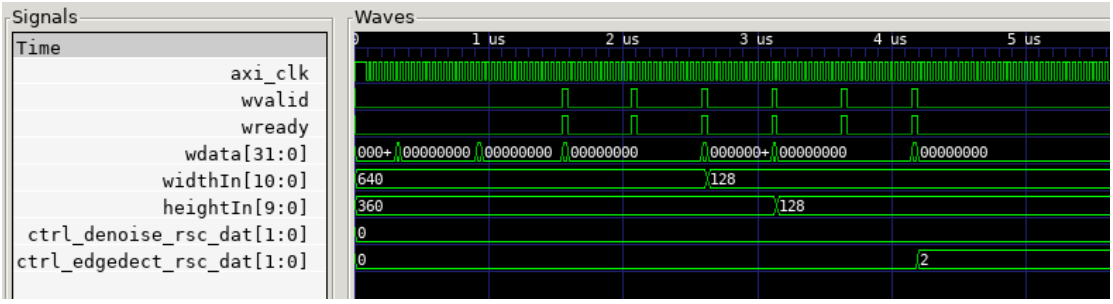
下圖為 test002_fpga_axis_req 的 log

```
 5450985=> fpga_axis_req send data, vcnt =        117
 5498185=> fpga_axis_req send data, vcnt =        118
 5545385=> fpga_axis_req send data, vcnt =        119
 5589705=> fpga_axis_req send data, vcnt =        120
 5636585=> fpga_axis_req send data, vcnt =        121
 5683785=> fpga_axis_req send data, vcnt =        122
 5730985=> fpga_axis_req send data, vcnt =        123
 5778185=> fpga_axis_req send data, vcnt =        124
 5822505=> fpga_axis_req send data, vcnt =        125
 5869385=> fpga_axis_req send data, vcnt =        126
 5916585=> fpga_axis_req send data, vcnt =        127
 5916585=> test002_fpga_axis_req done
 5916585=> soc_to_fpga_axis_expect_count =      16384
 5916585=> soc_to_fpga_axis_captured_count =      16109
 5916585=> fpga_axis_test_length =      16384
 5916585=> check_fpga_cnt =      16384
 6016225=> soc_to_fpga_axis_captured : send soc_to_fpga_axiis_event
-----
 6016225=> test002_up_soc_rpt [PASS] cfg_read_data_expect_value=00000002, cfg_read_data_captured=00000002
-----
 6016225=> test002 [PASS] soc_to_fpga_axis_expect_count =      16384, soc_to_fpga_axis_captured_count =      16384
```

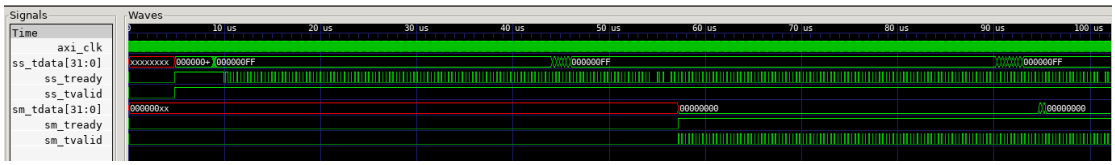
下圖為 report and out image check 的 log

```
 6016225=> test002 [PASS] idx3=     16361, soc_to_fpga_axis_expect_value[     16361] = 0000xxxxxx00, soc_to_fpga_axis_captured[     16361]  = 000X00000000
 6016225=> test002 [PASS] idx3=     16362, soc_to_fpga_axis_expect_value[     16362] = 0000xxxxxxff, soc_to_fpga_axis_captured[     16362]  = 000X000000ff
 6016225=> test002 [PASS] idx3=     16363, soc_to_fpga_axis_expect_value[     16363] = 0000xxxxxxff, soc_to_fpga_axis_captured[     16363]  = 000X000000ff
 6016225=> test002 [PASS] idx3=     16364, soc_to_fpga_axis_expect_value[     16364] = 0000xxxxxxff, soc_to_fpga_axis_captured[     16364]  = 000X000000ff
 6016225=> test002 [PASS] idx3=     16365, soc_to_fpga_axis_expect_value[     16365] = 0000xxxxxx00, soc_to_fpga_axis_captured[     16365]  = 000X00000000
 6016225=> test002 [PASS] idx3=     16366, soc_to_fpga_axis_expect_value[     16366] = 0000xxxxxx0c, soc_to_fpga_axis_captured[     16366]  = 000X0000000c
 6016225=> test002 [PASS] idx3=     16367, soc_to_fpga_axis_expect_value[     16367] = 0000xxxxxx00, soc_to_fpga_axis_captured[     16367]  = 000X00000000
 6016225=> test002 [PASS] idx3=     16368, soc_to_fpga_axis_expect_value[     16368] = 0000xxxxxx00, soc_to_fpga_axis_captured[     16368]  = 000X00000000
 6016225=> test002 [PASS] idx3=     16369, soc_to_fpga_axis_expect_value[     16369] = 0000xxxxxx00, soc_to_fpga_axis_captured[     16369]  = 000X00000000
 6016225=> test002 [PASS] idx3=     16370, soc_to_fpga_axis_expect_value[     16370] = 0000xxxxxx00, soc_to_fpga_axis_captured[     16370]  = 000X00000000
 6016225=> test002 [PASS] idx3=     16371, soc_to_fpga_axis_expect_value[     16371] = 0000xxxxxx00, soc_to_fpga_axis_captured[     16371]  = 000X00000000
 6016225=> test002 [PASS] idx3=     16372, soc_to_fpga_axis_expect_value[     16372] = 0000xxxxxx00, soc_to_fpga_axis_captured[     16372]  = 000X00000000
 6016225=> test002 [PASS] idx3=     16373, soc_to_fpga_axis_expect_value[     16373] = 0000xxxxxx00, soc_to_fpga_axis_captured[     16373]  = 000X00000000
 6016225=> test002 [PASS] idx3=     16374, soc_to_fpga_axis_expect_value[     16374] = 0000xxxxxx00, soc_to_fpga_axis_captured[     16374]  = 000X00000000
 6016225=> test002 [PASS] idx3=     16375, soc_to_fpga_axis_expect_value[     16375] = 0000xxxxxx00, soc_to_fpga_axis_captured[     16375]  = 000X00000000
 6016225=> test002 [PASS] idx3=     16376, soc_to_fpga_axis_expect_value[     16376] = 0000xxxxxx00, soc_to_fpga_axis_captured[     16376]  = 000X00000000
 6016225=> test002 [PASS] idx3=     16377, soc_to_fpga_axis_expect_value[     16377] = 0000xxxxxx00, soc_to_fpga_axis_captured[     16377]  = 000X00000000
 6016225=> test002 [PASS] idx3=     16378, soc_to_fpga_axis_expect_value[     16378] = 0000xxxxxx00, soc_to_fpga_axis_captured[     16378]  = 000X00000000
 6016225=> test002 [PASS] idx3=     16379, soc_to_fpga_axis_expect_value[     16379] = 0000xxxxxx00, soc_to_fpga_axis_captured[     16379]  = 000X00000000
 6016225=> test002 [PASS] idx3=     16380, soc_to_fpga_axis_expect_value[     16380] = 0000xxxxxx00, soc_to_fpga_axis_captured[     16380]  = 000X00000000
 6016225=> test002 [PASS] idx3=     16381, soc_to_fpga_axis_expect_value[     16381] = 0000xxxxxx00, soc_to_fpga_axis_captured[     16381]  = 000X00000000
 6016225=> test002 [PASS] idx3=     16382, soc_to_fpga_axis_expect_value[     16382] = 0000xxxxxx00, soc_to_fpga_axis_captured[     16382]  = 000X00000000
 6016225=> test002 [PASS] idx3=     16383, soc_to_fpga_axis_expect_value[     16383] = 0001xxxxxx00, soc_to_fpga_axis_captured[     16383]  = 000X00000000
 6016625=> soc_up_cfg_write : wbs_adr=30000018, wbs_sel=0001, wbs_wdata=00000001
-----
==========================================================================
==========================================================================
==========================================================================
 6017025=> Final result [PASS], check_cnt = 16392, error_cnt = 0000
```

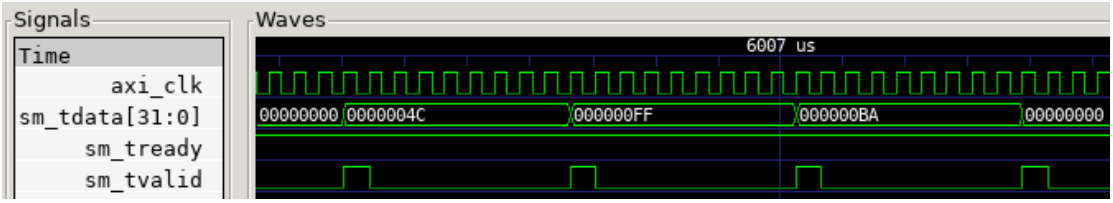
下圖為寫入 ctrl_reg 的波形

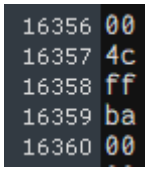


下圖為透過 AXIS/AXIM 接收和傳送的波形



下圖為 AXIM 傳送的詳細波形，這是跟 golden pattern 確認是否相同





Result

這是用 python 將 hex 檔轉為影像的結果

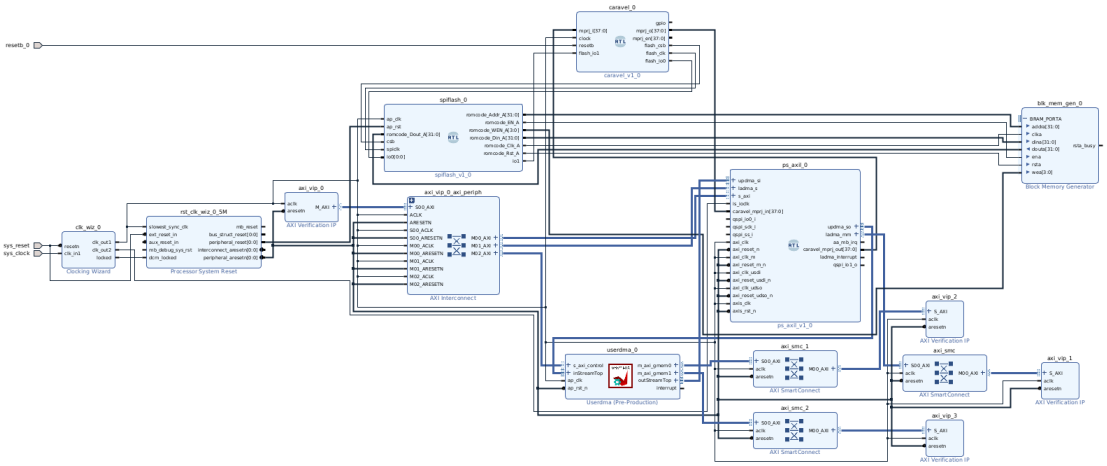
ctrl_denoise = 0	Bypass	ctrl_edgedect = 0	Bypass
ctrl_denoise = 1	Gaussian Filter	ctrl_edgedect = 1	Laplacian Filter
ctrl_denoise = 2	Medium Filter	ctrl_edgedect = 2	Sobel Filter

(ctrl_denoise ,ctrl_edgedect)		
(0,0)	(1,0)	(2,0)
		
(0,1)	(1,1)	(2,1)
		
(0,2)	(1,2)	(2,2)
		

2. Caravel-FSIC FPGA Simulation (with userdma)

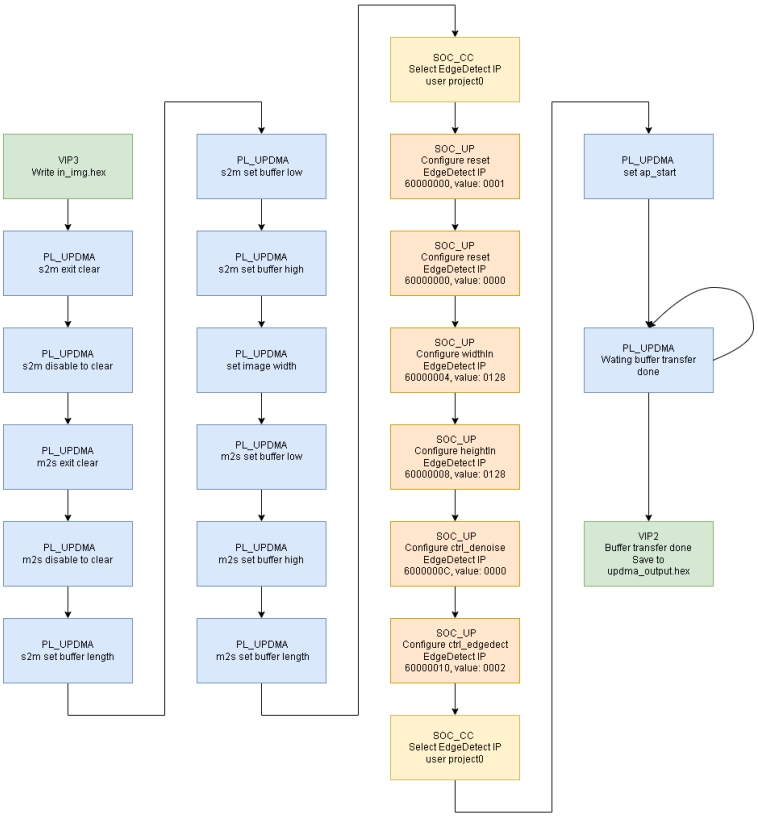
Block diagram

我們將 userdma 的 BUF_LEN 調整為圖片的大小，再使用 hls 重新合成。



Testbench

在 fsic_tb.sv 當中只有進行 Task SocUp2DmaPath。



下圖為透過 AXIlite 寫入 ctrl_reg 的 log

```
8225658=> AXI4LITE_WRITE_BURST 60000000, value: 0001, resp: 00
8237858=> AXI4LITE_READ_BURST 60000000, value: 0001, resp: 00
8237858=> Fpga2Soc_Write SOC_UP offset 000 = 00000001, PASS
8237858=> Fpga2Soc_Write: SOC_UP
8240458=> AXI4LITE_WRITE_BURST 60000000, value: 0000, resp: 00
8252658=> AXI4LITE_READ_BURST 60000000, value: 0000, resp: 00
8252658=> Fpga2Soc_Write SOC_UP offset 000 = 00000000, PASS
8252658=> Fpga2Soc_Write: SOC_UP
8255258=> AXI4LITE_WRITE_BURST 60000004, value: 0080, resp: 00
8267458=> AXI4LITE_READ_BURST 60000004, value: 0080, resp: 00
8267458=> Fpga2Soc_Write SOC_UP offset 004 = 00000080, PASS
8267458=> Fpga2Soc_Write: SOC_UP
8270058=> AXI4LITE_WRITE_BURST 60000008, value: 0080, resp: 00
8282258=> AXI4LITE_READ_BURST 60000008, value: 0080, resp: 00
8282258=> Fpga2Soc_Write SOC_UP offset 008 = 00000080, PASS
8282258=> Fpga2Soc_Write: SOC_UP
8284858=> AXI4LITE_WRITE_BURST 6000000c, value: 0000, resp: 00
8297058=> AXI4LITE_READ_BURST 6000000c, value: 0000, resp: 00
8297058=> Fpga2Soc_Write SOC_UP offset 00c = 00000000, PASS
8297058=> Fpga2Soc_Write: SOC_UP
8299658=> AXI4LITE_WRITE_BURST 60000010, value: 0002, resp: 00
8311858=> AXI4LITE_READ_BURST 60000010, value: 0002, resp: 00
8311858=> Fpga2Soc_Write SOC_UP offset 010 = 00000002, PASS
```

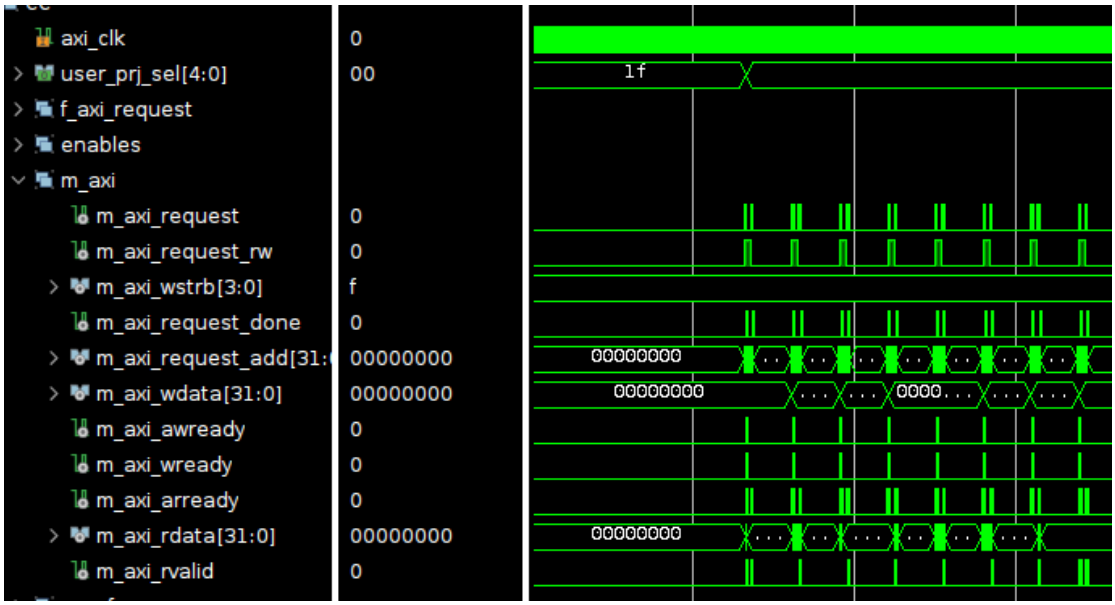
DMA 開始傳送輸入圖片的 log

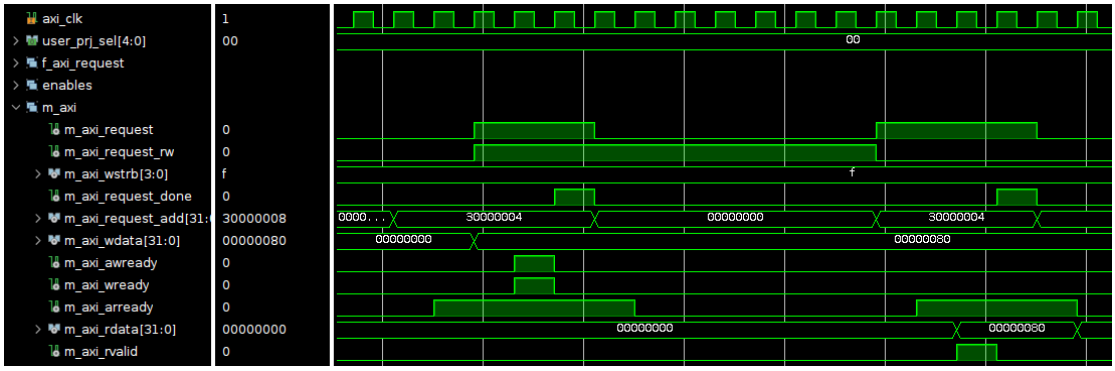
```
8328258=> Starting CheckuserDMADone()...
8328258=> ==========================================================
8328258=> FpgaLocal_Read: PL_UPDMA
8328258=> Wating buffer transfer done...
```

DMA 接收輸出圖片的 log

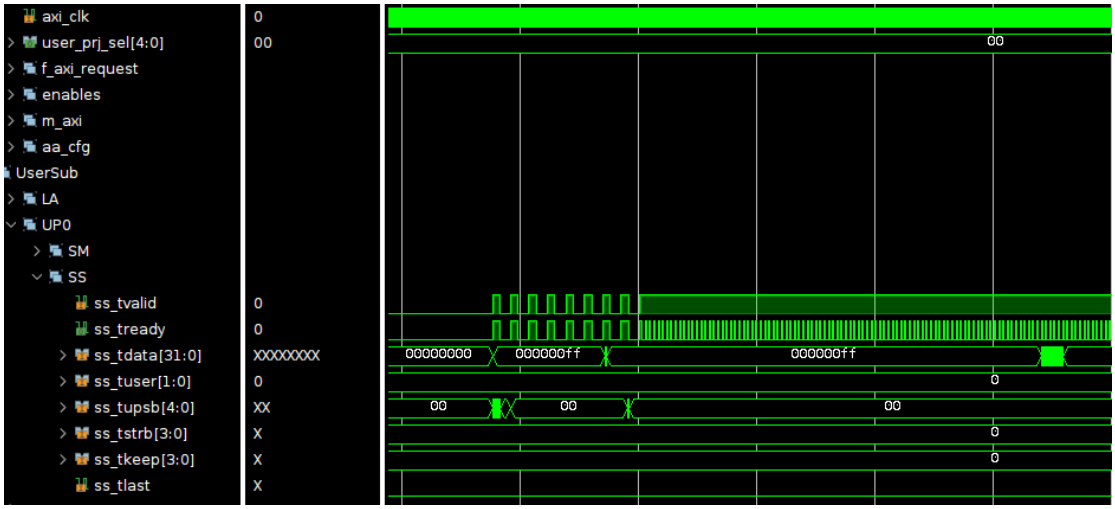
```
38457258=> Buffer transfer done. offset 010 = 00000001, PASS
38457258=> End CheckuserDMADone()...
38457258=> ==========================================================
38457258=> End SocUp2DmaPath() test...
38457258=> ==========================================================
```

下圖為透過 AXIlite 寫入 ctrl_reg 的波形

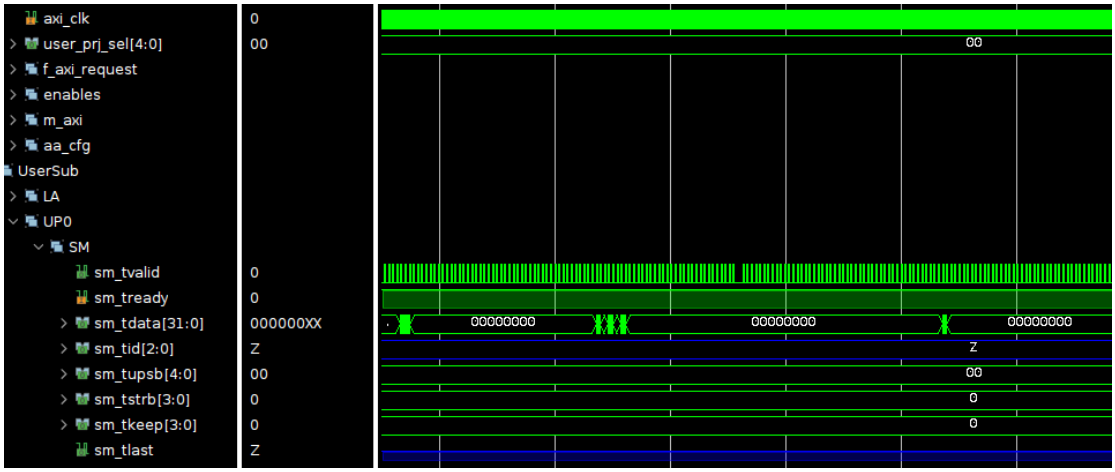




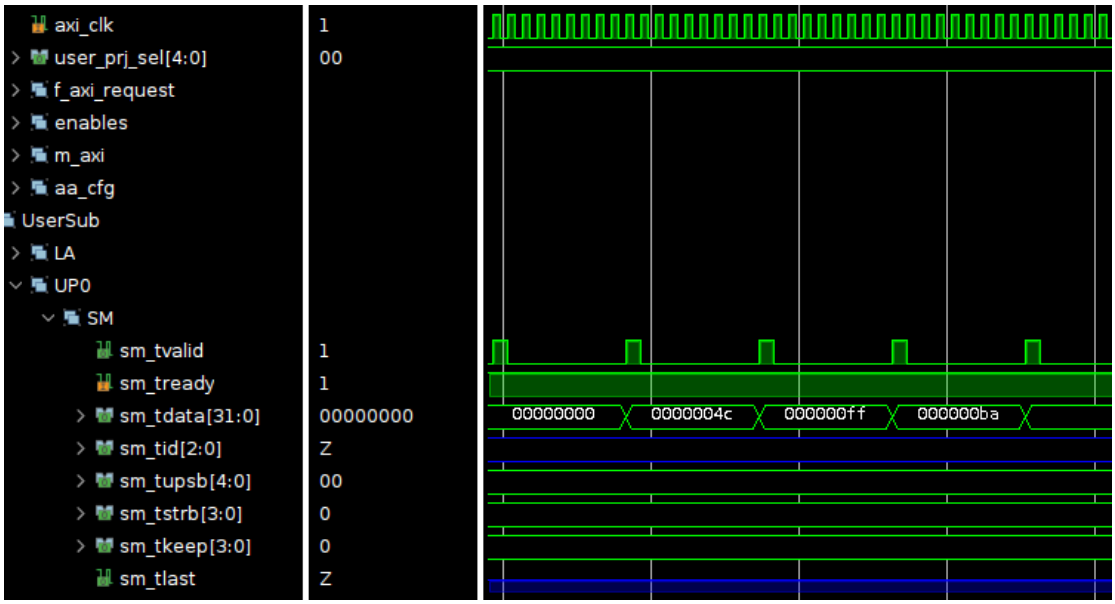
下圖為透過 AXIS 接收輸入的波形

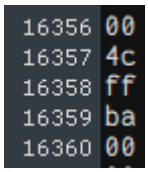


下圖為透過 AXIM 傳送輸出的波形



下圖為 AXIM 傳送的詳細波形，這是跟 golden pattern 比較是否相同





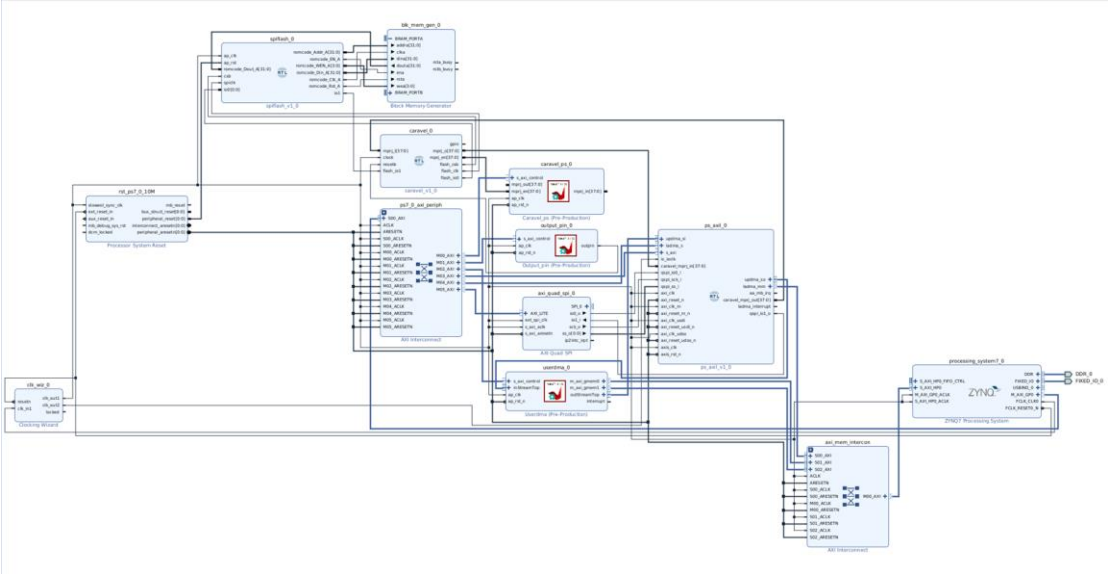
這是用 python 將 hex 檔轉為影像的結果

ctrl_denoise = 0	Bypass	ctrl_edgedect = 0	Bypass
ctrl_denoise = 1	Gaussian Filter	ctrl_edgedect = 1	Laplacian Filter
ctrl_denoise = 2	Medium Filter	ctrl_edgedect = 2	Sobel Filter

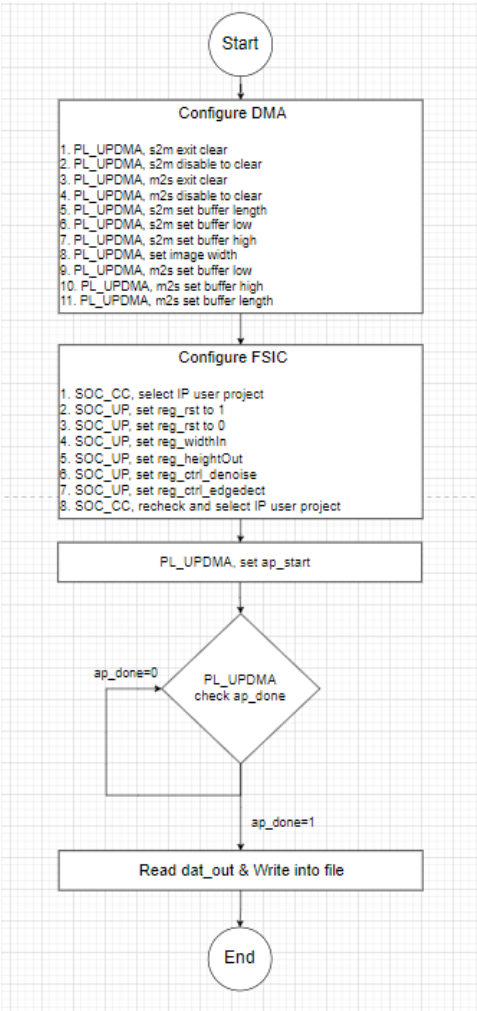
(ctrl_denoise ,ctrl_edgedect)		
(0,0)	(1,0)	(2,0)
		
(0,1)	(1,1)	(2,1)
		
(0,2)	(1,2)	(2,2)
		

三、FPGA Validation

1. Block Diagram



2. Jupyter notebook flow chart



Configure DMA

```
mmio_UPDMA.write(0x0000_0020, 0x0000_0000)   # PL_UPDMA, s2m exit clear
mmio_UPDMA.write(0x0000_0030, 0x0000_0000)   # PL_UPDMA, s2m disable to clear
mmio_UPDMA.write(0x0000_0078, 0x0000_0000)   # PL_UPDMA, m2s exit clear
mmio_UPDMA.write(0x0000_0088, 0x0000_0000)   # PL_UPDMA, m2s disable to clear

mmio_UPDMA.write(0x0000_0028, TST_TOTAL_PIXEL_NUM)   # PL_UPDMA, s2m set buffer length
mmio_UPDMA.write(0x0000_0038, S2MBUF_ADDRESS)# PL_UPDMA, s2m set buffer low
mmio_UPDMA.write(0x0000_003C, 0x0000_0000)   # PL_UPDMA, s2m set buffer high

mmio_UPDMA.write(0x0000_0054, TST_FRAME_WIDTH)   # PL_UPDMA, set image width

mmio_UPDMA.write(0x0000_005C, M2SBUF_ADDRESS)# PL_UPDMA, m2s set buffer low
mmio_UPDMA.write(0x0000_0060, 0x0000_0000)   # PL_UPDMA, m2s set buffer high
mmio_UPDMA.write(0x0000_0080, TST_TOTAL_PIXEL_NUM)   # PL_UPDMA, m2s set buffer length
```

Configure DMA

1. PL_UPDMA, s2m exit clear
2. PL_UPDMA, s2m disable to clear
3. PL_UPDMA, m2s exit clear
4. PL_UPDMA, m2s disable to clear
5. PL_UPDMA, s2m set buffer length
6. PL_UPDMA, s2m set buffer low
7. PL_UPDMA, s2m set buffer high
8. PL_UPDMA, set image width
9. PL_UPDMA, m2s set buffer low
10. PL_UPDMA, m2s set buffer high
11. PL_UPDMA, m2s set buffer length

Configure FSIC

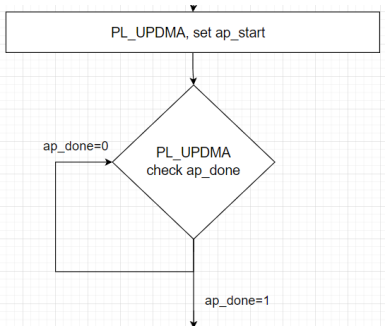
```
ADDRESS_OFFSET = SOC_CC                    # SOC_CC, select IP user project
mmio.write(ADDRESS_OFFSET, 0x0000_0000)
print("mmio.read(ADDRESS_OFFSET): ", hex(mmio.read(ADDRESS_OFFSET)))

ADDRESS_OFFSET = SOC_UP                    # SOC_UP, configure reg_rst to 1
mmio.write(ADDRESS_OFFSET + 0x00, 0x0000_0001)
print("mmio.read(ADDRESS_OFFSET + 0x00): ", hex(mmio.read(ADDRESS_OFFSET + 0x00)))

ADDRESS_OFFSET = SOC_UP                    # SOC_UP, configure reg_rst to 0
mmio.write(ADDRESS_OFFSET + 0x00, 0x0000_0000)
print("mmio.read(ADDRESS_OFFSET + 0x00): ", hex(mmio.read(ADDRESS_OFFSET + 0x00)))

ADDRESS_OFFSET = SOC_UP                    # SOC_UP, configure reg_widthIn
mmio.write(ADDRESS_OFFSET + 0x04, TST_FRAME_WIDTH)
print("mmio.read(ADDRESS_OFFSET + 0x04): ", hex(mmio.read(ADDRESS_OFFSET + 0x04)))

ADDRESS_OFFSET = SOC_UP                    # SOC_UP, configure reg_heightIn
mmio.write(ADDRESS_OFFSET + 0x08, TST_FRAME_HEIGHT)
print("mmio.read(ADDRESS_OFFSET + 0x08): ", hex(mmio.read(ADDRESS_OFFSET + 0x08)))

ADDRESS_OFFSET = SOC_UP                    # SOC_UP, configure reg_ctrl_denoise
mmio.write(ADDRESS_OFFSET + 0x0C, TST_CTRL_DENOISE)
print("mmio.read(ADDRESS_OFFSET + 0x0C): ", hex(mmio.read(ADDRESS_OFFSET + 0x0C)))

ADDRESS_OFFSET = SOC_UP                    # SOC_UP, configure reg_ctrl_edgedect
mmio.write(ADDRESS_OFFSET + 0x10, TST_CTRL_EDGEDECT)
print("mmio.read(ADDRESS_OFFSET + 0x10): ", hex(mmio.read(ADDRESS_OFFSET + 0x10)))

ADDRESS_OFFSET = SOC_CC                    # SOC_CC, recheck user project sel
mmio.write(ADDRESS_OFFSET, 0x0000_0000)
print("mmio.read(ADDRESS_OFFSET): ", hex(mmio.read(ADDRESS_OFFSET)))
```

Configure FSIC

1. SOC_CC, select IP user project
2. SOC_UP, set reg_rst to 1
3. SOC_UP, set reg_rst to 0
4. SOC_UP, set reg_widthIn
5. SOC_UP, set reg_heightOut
6. SOC_UP, set reg_ctrl_denoise
7. SOC_UP, set reg_ctrl_edgedect
8. SOC_CC, recheck and select IP user project

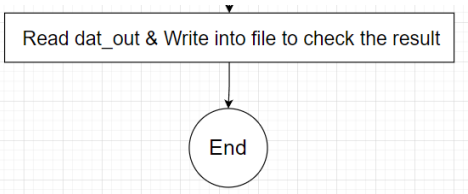
Set ap_start & polling ap_done

```
# ADDRESS_OFFSET = SOC_UP                    # SOC_UP, configure ap_start in FIR IP
# mmio.write(ADDRESS_OFFSET, 0x0000_0001)
mmio_UPDMA.write(0x0000_0000, 0x0000_0001)  # PL_UPDMA, set ap_start


while True:                                 # PL_UPDMA, polling ap_done
    if mmio_UPDMA.read(0x0000_0010)==0x0000_0001:
        break
```



Read dat_out & write into file to check the result

```
def write_buffer_to_hex_file(buffer, file_path):
    with open(file_path, 'w') as file:
        for value in buffer:
            file.write(f'{value:02x}\n')

output_hex_file_path = 'output.hex'
write_buffer_to_hex_file(UPDMA_BUF0, output_hex_file_path)

with open(output_hex_file_path, 'r') as file:
    for _ in range(10):
        print(file.readline().strip())
```



3. Result

Gaussian noise

Gaussian Filter

SaltPepper noise

Medium Filter

Reduced edge variation

Laplacian Filter

Enhanced edge variation

Sobel Filter

四、Synopsys flow

1. Synthesis

從 Catapult HLS 合成出來的 Verilog 檔，經過 tcl 檔合成出 netlist 檔。

```
  Optimization Complete
  ---------------------
1
## reporting and output
report_timing > ../../reports/timing_${DESIGN_NAME}_timing_reports.log
report_qor > ../../reports/timing_${DESIGN_NAME}_qor_reports.log
report_area -hierarchy  > ../../reports/timing_${DESIGN_NAME}_area_reports.log
report_power -hierarchy > ../../reports/timing_${DESIGN_NAME}_power_reports.log
change_names -rules verilog
Warning: The specified replacement character (_) is conflicting with the specified allow
ed or restricted character.   (UCN-4)
1
write_file -format verilog -hierarchy -pg -output ../../input/${DESIGN_NAME}.v
Warning: No PG information is available for design. (UPF-663)
Writing verilog file '/home/m111/m111061631/asoc_lab3/lab_formal_release/input/EdgeDetec
t_IP_EdgeDetect_Top.v'.
Warning: Verilog 'assign' or 'tran' statements are written out. (VO-4)
Warning: Verilog writer has added 32 nets to module EdgeDetect_IP_Denoise_IP_run using S
YNOPSYS_UNCONNECTED_ as prefix.  Please use the change_names command to make the correct
 changes before invoking the verilog writer.  (VO-11)
Warning: Verilog writer has added 2 nets to module EdgeDetect_IP_EdgeDetect_Filter_run_D
W02_mult_1_DW02_mult_2 using SYNOPSYS_UNCONNECTED_ as prefix.  Please use the change_nam
es command to make the correct changes before invoking the verilog writer.  (VO-11)
Warning: Verilog writer has added 25 nets to module EdgeDetect_IP_EdgeDetect_Filter_run
using SYNOPSYS_UNCONNECTED_ as prefix.  Please use the change_names command to make the
correct changes before invoking the verilog writer.  (VO-11)
1
quit

Memory usage for this session 364 Mbytes.
Memory usage for this session including child processes 364 Mbytes.
CPU usage for this session 44 seconds ( 0.01 hours ).
Elapsed time for this session 565 seconds ( 0.16 hours ).

Thank you ...
date >  compile_for_timing
date >  all
[m111061631@ws41 work]$
```

流程通過圖

```
****************************************
Report : area
Design : EdgeDetect_IP_EdgeDetect_Top
Version: R-2020.09-SP5
Date   : Wed Jun 19 22:23:43 2024
****************************************

Library(s) Used:

    saed14rvt_tt0p8v25c (File: /home/course/ee5252/lab_snps_flow/SAED14_EDK_LAB/SAED14_EDK_LAB/lib/stdcell_rvt
/db_nldm/saed14rvt_tt0p8v25c.db)

Number of ports:                          1923
Number of nets:                           6871
Number of cells:                          5068
Number of combinational cells:            4369
Number of sequential cells:                643
Number of macros/black boxes:                0
Number of buf/inv:                        1312
Number of references:                        4

Combinational area:               1499.165987
Buf/Inv area:                      261.027599
Noncombinational area:             649.794011
Macro/Black Box area:                0.000000
Net Interconnect area:            3105.299193

Total cell area:                  2148.959998
Total area:                       5254.259191

Hierarchical area distribution
------------------------------
```

Area Report

```
****************************************
Report : power
        -hier
        -analysis_effort low
Design : EdgeDetect_IP_EdgeDetect_Top
Version: R-2020.09-SP5
Date   : Wed Jun 19 22:23:43 2024
****************************************

Library(s) Used:

    saed14rvt_tt0p8v25c (File: /home/course/ee5252/lab_snps_flow/SAED14_EDK_LAB/SAED14_EDK_LAB/lib/stdcell_rvt
/db_nldm/saed14rvt_tt0p8v25c.db)


Operating Conditions: tt0p8v25c   Library: saed14rvt_tt0p8v25c
Wire Load Model Mode: top

Design        Wire Load Model            Library
------------------------------------------------
EdgeDetect_IP_EdgeDetect_Top
                       8000              saed14rvt_tt0p8v25c


Global Operating Voltage = 0.8
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000ff
    Time Units = 1ns
    Dynamic Power Units = 1uW    (derived from V,C,T units)
    Leakage Power Units = 1pW
```

Power Report

```
data arrival time                                                    1.74

clock clk (rise edge)                                   2.00         2.00
clock network delay (ideal)                             0.00         2.00
clock uncertainty                                      -0.30         1.70
EdgeDetect_Filter_inst/EdgeDetect_IP_EdgeDetect_Filter_run_inst/dat_out_rsci
                                                        0.00         1.70 r
library setup time                                      0.04         1.74
data required time                                                   1.74
---------------------------------------------------------------------------
data required time                                                   1.74
data arrival time                                                   -1.74
---------------------------------------------------------------------------
slack (MET)                                                          0.00
```

Timing Report

2. Floorplan

```
##########################################################################
######Save_Block
save_block -as ${DESIGN_NAME}_1_data_setup
Information: Saving 'EdgeDetect_IP_EdgeDetect_Top:EdgeDetect_IP_EdgeDetect
_Top.design' to 'EdgeDetect_IP_EdgeDetect_Top:EdgeDetect_IP_EdgeDetect_Top
_1_data_setup.design'. (DES-028)
1
save_lib
Saving library 'EdgeDetect_IP_EdgeDetect_Top'
1
close_block
Closing block 'EdgeDetect_IP_EdgeDetect_Top:EdgeDetect_IP_EdgeDetect_Top.d
esign'
1
close_lib
Closing library 'EdgeDetect_IP_EdgeDetect_Top'
1
exit
Maximum memory usage for this session: 411.73 MB
Maximum memory usage for this session including child processes: 755.96 MB
CPU usage for this session:     13 seconds (  0.00 hours)
Elapsed time for this session:    484 seconds (  0.13 hours)
Thank you for using IC Compiler II.
date > step1_data_setup
```

Step1.data_setup

data_setup 的指令主要是呼叫資料庫，並設定目前的 top design 並執行 Constraints file。

```
##############################################################################Save_B
lock
save_block -as temp_floorplane_placed
Information: Saving 'EdgeDetect_IP_EdgeDetect_Top:temp_data_setup.design' to 'EdgeDete
ct_IP_EdgeDetect_Top:temp_floorplane_placed.design'. (DES-028)
1
##############################################################################Save_B
lock
save_block -as ${DESIGN_NAME}_2_floorplan_ends
Information: Saving 'EdgeDetect_IP_EdgeDetect_Top:temp_data_setup.design' to 'EdgeDete
ct_IP_EdgeDetect_Top:EdgeDetect_IP_EdgeDetect_Top_2_floorplan_ends.design'. (DES-028)
1
save_lib
Saving library 'EdgeDetect_IP_EdgeDetect_Top'
1
##########################################################################
close_block
Information: Decrementing open_count of block 'EdgeDetect_IP_EdgeDetect_Top:temp_data_
setup.design' to 1. (DES-022)
1
close_lib
Closing library 'EdgeDetect_IP_EdgeDetect_Top'
1
exit
Maximum memory usage for this session: 718.62 MB
Maximum memory usage for this session including child processes: 718.62 MB
CPU usage for this session:     17 seconds (  0.00 hours)
Elapsed time for this session:    489 seconds (  0.14 hours)
Thank you for using IC Compiler II.
date > step2_floorplan
```

Step2.floorplan

floorplan 的指令主要是設定電源 VDD 和接地 VSS，並生成成 clocks、exceptions、disable 等報告。

```
##############################################################################Save_
Block
save_block -as ${DESIGN_NAME}_3_powerplan_ends
Information: Saving 'EdgeDetect_IP_EdgeDetect_Top:temp_floorplan_ends.design' to 'Edg
eDetect_IP_EdgeDetect_Top:EdgeDetect_IP_EdgeDetect_Top_3_powerplan_ends.design'. (DES
-028)
1
save_lib
Saving library 'EdgeDetect_IP_EdgeDetect_Top'
1
close_block
Information: Decrementing open_count of block 'EdgeDetect_IP_EdgeDetect_Top:temp_floo
rplan_ends.design' to 1. (DES-022)
1
close_lib
Closing library 'EdgeDetect_IP_EdgeDetect_Top'
1
exit
Maximum memory usage for this session: 348.37 MB
Maximum memory usage for this session including child processes: 348.37 MB
CPU usage for this session:     27 seconds (  0.01 hours)
Elapsed time for this session:    614 seconds (  0.17 hours)
Thank you for using IC Compiler II.
date >  step3_powerplan
date > all
```

Step3.powerplan

powerplan 的指令主要是連接 VDD 和 VSS，建立電源規劃策略、編譯並應用設置的電源策略、檢查設計中的電源和地線連接性，確保所有部分正確連接等。

3. Place and Route

```
##############################################################################Save_
Block
save_block -as ${DESIGN_NAME}_4_place_ends
Information: Saving 'EdgeDetect_IP_EdgeDetect_Top:temp_powerplan_ends.design' to 'Edg
eDetect_IP_EdgeDetect_Top:EdgeDetect_IP_EdgeDetect_Top_4_place_ends.design'. (DES-028
)
1
save_lib
Saving library 'EdgeDetect_IP_EdgeDetect_Top'
1
##############################################################################
close_block
Information: Decrementing open_count of block 'EdgeDetect_IP_EdgeDetect_Top:temp_powe
rplan_ends.design' to 1. (DES-022)
1
close_lib
Closing library 'EdgeDetect_IP_EdgeDetect_Top'
Information: The net parasitics of block EdgeDetect_IP_EdgeDetect_Top are cleared. (T
IM-123)
1
exit
Maximum memory usage for this session: 715.53 MB
Maximum memory usage for this session including child processes: 715.53 MB
CPU usage for this session:    141 seconds (  0.04 hours)
Elapsed time for this session:    890 seconds (  0.25 hours)
Thank you for using IC Compiler II.
date > step4_place
```

Step4.place

place 的指令主要設置一系列優化和布局相關的選項，place_opt 和 legalize_placement 分別執行了佈局和合法化操作。

```
##############################################################################Save_
Block
save_block -as ${DESIGN_NAME}_5_clock_ends
Information: The command 'save_block' cleared the undo history. (UNDO-016)
Information: Saving 'EdgeDetect_IP_EdgeDetect_Top:temp_place_ends.design' to 'EdgeDet
ect_IP_EdgeDetect_Top:EdgeDetect_IP_EdgeDetect_Top_5_clock_ends.design'. (DES-028)
1
save_lib
Saving library 'EdgeDetect_IP_EdgeDetect_Top'
1
close_block
Information: Decrementing open_count of block 'EdgeDetect_IP_EdgeDetect_Top:temp_plac
e_ends.design' to 1. (DES-022)
1
close_lib
Closing library 'EdgeDetect_IP_EdgeDetect_Top'
Information: The net parasitics of block EdgeDetect_IP_EdgeDetect_Top are cleared. (T
IM-123)
1
exit
Maximum memory usage for this session: 1241.62 MB
Maximum memory usage for this session including child processes: 1241.62 MB
CPU usage for this session:    340 seconds (  0.09 hours)
Elapsed time for this session:   1159 seconds (  0.32 hours)
Thank you for using IC Compiler II.
date > step5_clock_tree_syntesis
```

Step5.clock tree synthesis

clock tree synthesis 的指令命令了時脈樹的選項，包含了 clock 和 skew，並創建時脈的 route 規則用於後面的 flow。

```
close_block
Information: Decrementing open_count of block 'EdgeDetect_IP_EdgeDetect_Top:temp_clock
_ends.design' to 1. (DES-022)
1
close_lib
Closing library 'EdgeDetect_IP_EdgeDetect_Top'
Information: The net parasitics of block EdgeDetect_IP_EdgeDetect_Top are cleared. (TI
M-123)
1
exit
Maximum memory usage for this session: 788.69 MB
Maximum memory usage for this session including child processes: 788.69 MB
CPU usage for this session:    184 seconds (  0.05 hours)
Elapsed time for this session:    658 seconds (  0.18 hours)
Thank you for using IC Compiler II.
```

Step6.Route

route 的指令主要是進行自動化佈線以及優化，同時自動連接電源，並進行 LVS 檢查。

4. StarRC

StarRC 用於執行靜態分析和從設計中提取 RC 參數，其分別針對 slow 和 fast 兩種 corner 進行分析。

```
Setup                   Elp=00:00:25 Cpu=00:00:01 Usr=1.6    Sys=0.0    Mem=539.8
Layers                  Elp=00:00:00 Cpu=00:00:00 Usr=0.0    Sys=0.0    Mem=539.8
HN                      Elp=00:00:01 Cpu=00:00:00 Usr=0.2    Sys=0.0    Mem=546.0
Cells                   Elp=00:00:01 Cpu=00:00:00 Usr=0.2    Sys=0.1    Mem=560.2
Translate               Elp=00:00:00 Cpu=00:00:00 Usr=0.1    Sys=0.0    Mem=551.8
NetlistSetup            Elp=00:00:00 Cpu=00:00:00 Usr=0.0    Sys=0.0    Mem=480.5
GPD_XtractSetup         Elp=00:00:00 Cpu=00:00:00 Usr=0.0    Sys=0.0    Mem=540.1
GPD_NameMap             Elp=00:00:00 Cpu=00:00:00 Usr=0.0    Sys=0.0    Mem=481.0
xTract                  Elp=00:00:08 Cpu=00:00:08 Usr=7.5    Sys=0.7    Mem=961.3
xTractPP                Elp=00:00:00 Cpu=00:00:00 Usr=0.0    Sys=0.0    Mem=338.8
ReportViolations        Elp=00:00:00 Cpu=00:00:00 Usr=0.0    Sys=0.0    Mem=480.5
ReportOpens             Elp=00:00:00 Cpu=00:00:00 Usr=0.0    Sys=0.0    Mem=480.2
GPD_PostProcess         Elp=00:00:00 Cpu=00:00:00 Usr=0.0    Sys=0.0    Mem=480.6
GPD_Converter1          Elp=00:00:01 Cpu=00:00:00 Usr=0.5    Sys=0.0    Mem=336.5
GPD_Converter2          Elp=00:00:00 Cpu=00:00:00 Usr=0.0    Sys=0.0    Mem=332.6
GPD_Converter_merge_c1 Elp=00:00:00 Cpu=00:00:00 Usr=0.0    Sys=0.0    Mem=332.5
GPD_Converter_merge_c2 Elp=00:00:00 Cpu=00:00:00 Usr=0.0    Sys=0.0    Mem=332.5

Done          Elp=00:00:36 Cpu=00:00:10 Usr=10.1   Sys=0.8    Mem=961.3
date >  run_StarRC_cmd
date >  all
```

通過流程圖

5. PrimeTime

透過 PrimeTime 分析，每一次迭代會造成 Area 些微減少，也並未產生任何 violation，由此可知，PrimeTime 迭代後的效果是確保設計在滿足所有約束的 同時進行最大程度優化性能。

```
Initial cell usage:
Cell Group            Count                    Area
-----------------------------------------------------------
Combinational           560 ( 79%)           166.99 ( 50%)
Sequential              153 ( 21%)           164.64 ( 50%)
Clock                     0 (  0%)             0.00 (  0%)
Others                    0 (  0%)             0.00 (  0%)
-----------------------------------------------------------
Total                   713 (100%)           331.62 (100%)
```

```
Cell usage after iteration 1:
Cell Group            Count                    Area
-----------------------------------------------------------
Combinational           560 ( 79%)           166.99 ( 50%)
Sequential              153 ( 21%)           164.64 ( 50%)
Clock                     0 (  0%)             0.00 (  0%)
Others                    0 (  0%)             0.00 (  0%)
-----------------------------------------------------------
Total                   713 (100%)           331.62 (100%)
```

```
Cell usage after iteration 2:
Cell Group            Count                    Area
-----------------------------------------------------------
Combinational           560 ( 79%)           166.99 ( 50%)
Sequential              153 ( 21%)           164.55 ( 50%)
Clock                     0 (  0%)             0.00 (  0%)
Others                    0 (  0%)             0.00 (  0%)
-----------------------------------------------------------
Total                   713 (100%)           331.53 (100%)
```

```
Cell usage after iteration 3:
Cell Group            Count                    Area
-----------------------------------------------------------
Combinational           560 ( 79%)          166.19 ( 50%)
Sequential              153 ( 21%)          164.55 ( 50%)
Clock                     0 (  0%)            0.00 (  0%)
Others                    0 (  0%)            0.00 (  0%)
-----------------------------------------------------------
Total                   713 (100%)          330.74 (100%)
```

```
Cell usage after iteration 4:
Cell Group            Count                    Area
-----------------------------------------------------------
Combinational           560 ( 79%)          164.24 ( 50%)
Sequential              153 ( 21%)          164.55 ( 50%)
Clock                     0 (  0%)            0.00 (  0%)
Others                    0 (  0%)            0.00 (  0%)
-----------------------------------------------------------
Total                   713 (100%)          328.78 (100%)
```

```
Final cell usage:
Cell Group            Count                    Area
-----------------------------------------------------------
Combinational           560 ( 79%)          164.24 ( 50%)
Sequential              153 ( 21%)          164.55 ( 50%)
Clock                     0 (  0%)            0.00 (  0%)
Others                    0 (  0%)            0.00 (  0%)
-----------------------------------------------------------
Total                   713 (100%)          328.78 (100%)
```

6. DRC

檢查 layout 檔有沒有違反晶圓代工廠的 DRC Rule

```
IC Validator Machine Memory Report
ws41          : Average = 0.332 GB, Peak = 0.701 GB

Overall Disk Usage Disk=0.030 GB
Network Disk Usage Peak=0.007 GB (no group)
Group File Disk Usage Peak=0.023 GB
Overall engine Time=0:01:14 Highest command Mem=0.139 GB

Overall Master Mem=0.758 GB

Time Ended: 2024-06-19 14:50:34
```

通過流程圖

7. LVS

在 LVS 驗證中，需要輸入 GDSII 和 SPICE 檔案，這是為了要能夠去比較合成出 來的電路和 SPICE 中原始的電路之間的動作是否一致。

```
-------------
IC Validator Run: Time=0:02:00

IC Validator Machine Memory Report
ws41          : Average = 1.079 GB, Peak = 2.120 GB

Overall Disk Usage Disk=0.029 GB
Network Disk Usage Peak=0.007 GB (no group)
Group File Disk Usage Peak=0.022 GB
Overall engine Time=0:02:00 Highest command Mem=0.782 GB

Overall Master Mem=2.750 GB

Time Ended: 2024-06-19 14:57:19

IC Validator is done.
```

通過流程圖

8. Formality

Formality 是整個 Synopsys flow 的最後一關，用來驗證 layout 後的電路的邏輯是否與一開始 rtl 電路的一致。

```
*****************************************************************************
**

Status:  Verifying...

********************************** Matching Results **********************************
**
 767 Compare points matched by name
 0 Compare points matched by signature analysis
 0 Compare points matched by topology
 102 Matched primary inputs, black-box outputs
 0(0) Unmatched reference(implementation) compare points
 0(0) Unmatched reference(implementation) primary inputs, black-box outputs
*****************************************************************************
**


********************************* Verification Results *********************************
**
Verification SUCCEEDED
---------------------
 Reference design: r:/WORK/EdgeDetect_IP_EdgeDetect_Top
 Implementation design: i:/WORK/EdgeDetect_IP_EdgeDetect_Top
 767 Passing compare points
-----------------------------------------------------------------------------
--
Matched Compare Points     BBPin    Loop   BBNet     Cut    Port     DFF     LAT   TOT
AL
-----------------------------------------------------------------------------
--
Passing (equivalent)           0       0       0       0     124     643       0     7
67
Failing (not equivalent)       0       0       0       0       0       0       0
 0
*****************************************************************************
**
1
quit

Maximum memory usage for this session: 631 MB
CPU usage for this session: 5.38 seconds ( 0.00 hours )
Current time: Wed Jun 19 21:52:05 2024
Elapsed time: 121 seconds ( 0.03 hours )

Thank you for using Formality (R)!
```

驗證通過

9. Difficulties we met

這次 Synopsys flow 遇到最大的困難是設置環境變數的地方，因為發現若是在清大工作站使用 TA 提供的 ICC2 的版本，也就是 2020.09 的版本，在跑 route 的時候會遇到 Error 如下圖，導致後面的 flow 也無法順利執行。

```
A detailed stack trace has been captured in /home/m111/m111061631/asoc_lab3/lab_formal_release/lab2_pnr/work/Synopsys_stack_trace_18692.txt.

The tool has just encountered a fatal error:

If you encountered this fatal error when using the most recent
Synopsys release, submit the above stack trace and a test case that
reproduces the problem to the Synopsys Support Center by using
Enter A Call at http://solvnet.synopsys.com/EnterACall.

* For information about the latest software releases, go to the Synopsys
  SolvNet Release Library at http://solvnet.synopsys.com/ReleaseLibrary.

* For information about required Operating System patches, go to
  http://www.synopsys.com/support


Fatal: Internal system error, cannot recover.
Error code=6

Release = 'R-2020.09-SP3'  Architecture = 'linux64'  Program = 'IC Compiler II'
Exec = '/usr/cadtool/cad/synopsys/icc2/2020.09-sp3/linux64/nwtn/bin/dgcom_exec'

'533868330 48010806371328 48010806371207 48010806377080 48010392550381 48010392216982 48010392212969 48010392215349 48010804022481 48010804024558 48010434873940 48010434809740
 455750428 455886883 582897498 582663053 626480197 626499846 582786398 582577399 582612714 582663053 626480197 626499846 582647833 582659639 565102098 565119299 6611856 6623769
 48010806289749 27689489'

A snapshot of runtime data has been captured in /home/m111/m111061631/asoc_lab3/lab_formal_release/lab2_pnr/work/crte_000018692.txt
```

因此，在跑 route 的時候需要將 ICC2 的環境設置在 2021.06 的版本，改完環境變數之後，就能夠順利運行。