E111064503 通訊所 吳紹齊 HW3

https://github.com/cowboy35927/ESL/tree/main/Hw3

1.Base Implementation

這裡是定義通道的 header，如果沒有定義就自動定義 p2p，反之 fifo。

```
class SobelFilter: public sc_module
{
public:
  sc_in_clk i_clk;
  sc_in < bool >  i_rst;
#ifndef NATIVE_SYSTEMC
  cynw_p2p< sc_dt::sc_uint<24> >::in i_rgb;
  cynw_p2p< sc_dt::sc_uint<32> >::out o_result;
#else
  sc_fifo_in< sc_dt::sc_uint<24> > i_rgb;
  sc_fifo_out< sc_dt::sc_uint<32> > o_result;
#endif

  SC_HAS_PROCESS( SobelFilter );
  SobelFilter( sc_module_name n );
  ~SobelFilter();
private:
  //void do_filter();
  //int val[MASK_N];
  //void mergeSort(unsigned char *arr, unsigned char  *temp ,int start, int end);
  //void merge(unsigned char *arr,unsigned char  *temp, int start, int mid, int end);
  unsigned char MeanFiter(unsigned char *arr);
  void do_filter();
```

這裡是 filter.cpp，下圖是進行初始化，設定 rst 和 clk。

```
SobelFilter::SobelFilter(sc_module_name n) : sc_module(n)
{
/*
#ifndef NATIVE_SYSTEMC
  HLS_FLATTEN_ARRAY(val);
#endif
*/
  SC_THREAD(do_filter);
  sensitive << i_clk.pos();
  dont_initialize();
  reset_signal_is(i_rst, false);

#ifndef NATIVE_SYSTEMC
  i_rgb.clk_rst(i_clk, i_rst);
  o_result.clk_rst(i_clk, i_rst);
#endif
}

SobelFilter::~SobelFilter() {}
```

下圖是 do_filter 進行 reset。

```
void SobelFilter::do_filter()
{
  unsigned char buffer [3][15];
  //vector<vector<unsigned char>> buffer(3, vector<unsigned char>(15, 0));
  // int x=0;
  int i = 0;
  long long int a = 0;
  sc_time start_time;
  sc_time  end_time;
  {
#ifndef NATIVE_SYSTEMC
    HLS_DEFINE_PROTOCOL("main_reset");
    i_rgb.reset();
    o_result.reset();
#endif
    wait();
  }
```

下圖是從 testbench 去讀取 pixel，pixel 會經過 p2p 的 channel(i_rgb)來得到，然後將 pixel 拆成 r、g、b 三個部分，分別寫入 val[0]、val[1]、val[2]。

```
while (true)
{
  unsigned char mid_r = 0, mid_g = 0, mid_b = 0;
  unsigned char val [3][9];
  //vector<vector<unsigned char>> val(3, vector<unsigned char>(9, 0));
  /*
  for (unsigned int i = 0; i<MASK_N; ++i) {
    HLS_CONSTRAIN_LATENCY(0, 1, "lat00");
    val[i] = 0;
  }*/
  for (unsigned int v = 0; v < MASK_Y; ++v)
  {
    for (unsigned int u = 0; u < MASK_X; ++u)
    {
      sc_dt::sc_uint<24> rgb;
#ifndef NATIVE_SYSTEMC
      {
        HLS_DEFINE_PROTOCOL("input");
        rgb = i_rgb.get();
        wait();
      }
#else
      rgb = i_rgb.read();
#endif
      if (v==0 && u == 0){
          start_time=sc_time_stamp();
      }
      val[0][v * 3 + u] = rgb.range(7, 0);
      val[1][v * 3 + u] = rgb.range(15, 8);
      val[2][v * 3 + u] = rgb.range(23, 16);
      /*unsigned char grey = (rgb.range(7,0) + rgb.range(15,8) + rgb.range(23, 16))/3;
      for (unsigned int i = 0; i < MASK_N; ++i) {
        HLS_CONSTRAIN_LATENCY(0, 1, "lat01");
        val[i] += grey * mask[i][u][v];
      }*/
      //cout << "Now at " << sc_time_stamp() <<" read rgb,i= "<< v*3+u<< endl; //print current sc_time
    }
  }
```

然後進行 Median Filter，取出 Median pixel。

```
for(int id=0;id<3;id++)
{
  for(int i = 8; i > 0; i--)
  {
      for(int j = 0; j <= i-1; j++)
      {
          if( val[id][j] > val[id][j+1])
          {
              swap(val[id][j], val[id][j+1]);
              //temp = val[id][j];
              //val[id][j] = val[id][j+1];
              //val[id][j+1] = temp;
          }
      }
  }
}
//mergeSort(val[0],r_temp, 0, 8);
//mergeSort(val[1],g_temp, 0, 8);
//mergeSort(val[2],b_temp, 0, 8);
if (i < 5)
{
  buffer[0][i * 3] = val[0][4];
  buffer[1][i * 3] = val[1][4];
  buffer[2][i * 3] = val[2][4];
}
else if (i >= 5 && i < 10)
{
  buffer[0][(i - 5) * 3 + 1] = val[0][4];
  buffer[1][(i - 5) * 3 + 1] = val[1][4];
  buffer[2][(i - 5) * 3 + 1] = val[2][4];
}
else
{
  buffer[0][(i - 10) * 3 + 2] = val[0][4];
  buffer[1][(i - 10) * 3 + 2] = val[1][4];
  buffer[2][(i - 10) * 3 + 2] = val[2][4];
}
```

下圖是 Mean filter，將得到的 median pixel 放入 buffer 後經過 Mean filter，得到 mean pixel，然後再分別乘 2^16、2^8、1，放入 total，最後經過 p2p channel(o_result)到 testbench。

```
      // cout << "Now at " << sc_time_stamp() <<" 5164165165,i= "<< i<< endl; //print current sc_time
      //vector<unsigned char> arr_r1(buffer[0].begin(), buffer[0].begin() + 9);
      //vector<unsigned char> arr_g1(buffer[1].begin(), buffer[1].begin() + 9);
      //vector<unsigned char> arr_b1(buffer[2].begin(), buffer[2].begin() + 9);
      unsigned char arr_r1 [9];
      unsigned char arr_g1 [9];
      unsigned char arr_b1 [9];
      for(int i=0;i<3;i++){
        for(int j=0;j<9;j++){
          if(i==0){
            arr_r1[j]=buffer[i][j];
          }else if(i==1){
            arr_g1[j]=buffer[i][j];
          }else{
            arr_b1[j]=buffer[i][j];
          }
        }
      }
      mid_r = MeanFiter(arr_r1);
      mid_g = MeanFiter(arr_g1);
      mid_b = MeanFiter(arr_b1);

      int total = 0;
      total = mid_r* 65536 + mid_g * 256 + mid_b ;
/*o_result_r.write(mid_r);
o_result_g.write(mid_g);
o_result_b.write(mid_b);
wait(1);*/
#ifndef NATIVE_SYSTEMC
      {
        HLS_DEFINE_PROTOCOL("output");
        o_result.put(total);
        wait();
      }
#else
      o_result.write(total);
#endif
      //cout << "Now at CPP85454123132 " << sc_time_stamp() << " i " << i << endl;
      //     vector<vector<unsigned char>> mean(3, vector<unsigned char>(9, 0));
```

下圖是 testbench，定義通道的 header，如果沒有定義就自動定義 p2p，反之 fifo。

```
19
20 class Testbench : public sc_module {
21 public:
22   sc_in_clk i_clk;
23   sc_out < bool >  o_rst;
24 #ifndef NATIVE_SYSTEMC
25   cynw_p2p< sc_dt::sc_uint<24> >::base_out o_rgb;
26   cynw_p2p< sc_dt::sc_uint<32> >::base_in i_result;
27 #else
28   sc_fifo_out< sc_dt::sc_uint<24> > o_rgb;
29   sc_fifo_in< sc_dt::sc_uint<32> > i_result;
30 #endif
31
32   SC_HAS_PROCESS(Testbench);
33
34   Testbench(sc_module_name n);
35   ~Testbench();
36
```

下圖是 testbench 進行初始化，設定 clk。

```
Testbench::Testbench(sc_module_name n) : sc_module(n), output_rgb_raw_data_offset(54) {
  SC_THREAD(feed_rgb);
  sensitive << i_clk.pos();
  dont_initialize();
  SC_THREAD(fetch_result);
  sensitive << i_clk.pos();
  dont_initialize();
}

Testbench::~Testbench() {
  //cout<< "Max txn time = " << max_txn_time << endl;
  //cout<< "Min txn time = " << min_txn_time << endl;
  //cout<< "Avg txn time = " << total_txn_time/n_txn << endl;
  cout << "Total run time = " << total_run_time << endl;
}
```

下圖是 testbench 的 feed_rgb，是將讀進來的圖片去解析出每個 pixel 的 rgb，將解析出來的 rgb 經過 p2p channel(o_rgb)輸出到 filter，latency 在讀第一個 pixel 後寫入當時時間 start_time，為了算 latency。

```
  total_start_time = sc_time_stamp();
  for (y = 0; y != height; ++y)
  {
    int count = 0;
    for (x = 2; x < width; x = x + 3)
    {
    adjustX = (MASK_X % 2) ? 1 : 0; // 1
    adjustY = (MASK_Y % 2) ? 1 : 0;    // 1
    xBound = MASK_X / 2;               // 1
    yBound = MASK_Y / 2;               // 1
    for (int count_y = 0; count_y < 3; count_y++)
    {
      int counting=0;
      for (int count_x = -2; count_x < 3; count_x++)
      {
        for (v = -yBound; v != yBound + adjustY; ++v)
        { //-1, 0, 1
          for (u = -xBound; u != xBound + adjustX; ++u)
          { //-1, 0, 1
            if (x + count_x + u >= 0 && x + count_x + u < width && y + count_y + v >= 0 && y + count_y + v < height)
            {
              R = *(source_bitmap +
                    bytes_per_pixel * (width * (y + count_y + v) + (x + count_x + u)) + 2);
              G = *(source_bitmap +
                    bytes_per_pixel * (width * (y + count_y + v) + (x + count_x + u)) + 1);
              B = *(source_bitmap +
                    bytes_per_pixel * (width * (y + count_y + v) + (x + count_x + u)) + 0);
            }
            else
            {
              R = 0;
              G = 0;
              B = 0;
            }
            sc_dt::sc_uint<24> rgb;
            rgb.range(7, 0) = R;
            rgb.range(15, 8) = G;
            rgb.range(23, 16) = B;
#ifndef NATIVE_SYSTEMC
            o_rgb.put(rgb);
#else
            o_rgb.write(rgb);
#endif
            if (v==-1 && u == -1){
              start_time=sc_time_stamp();
            }
            /*o_r.write(R);
            o_g.write(G);
            o_b.write(B);
```

下圖是 fetch_result，將得到的 pixel 寫成 bmp 的圖片，會從 p2p channel(i_result)得到 pixel，再去解析 rgb 後分別寫入。

```
void Testbench::fetch_result()
{
  unsigned int x, y; // for loop counter
  int adjustX, adjustY, xBound, yBound;
  int pixel_num=0;
#ifndef NATIVE_SYSTEMC
  i_result.reset();
#endif
  wait(5);
  wait(1);
  for (y = 0; y != height; ++y)
  {
    int count = 0;
    for (x = 2; x < width; x = x + 3)
    {
      adjustX = 1;  // 1
      adjustY = 1;  // 1
      xBound = 1;   // 1
      yBound = 1;   // 1
      //cout<<"height:"<<y<<",width"<<x<<endl;
      for (int count_y = 0; count_y < 3; count_y++)
      {
        for (int count_x = -2; count_x < 3; count_x++)
        {
          int total = 0;
        if (count_y == 2 && count_x == 0)
          {
#ifndef NATIVE_SYSTEMC
            total = i_result.get();
#else
            total = i_result.read();
#endif
            *(target_bitmap + bytes_per_pixel * (width * y + x - 2) + 2) = total/(256*256);
            *(target_bitmap + bytes_per_pixel * (width * y + x - 2) + 1) = (total/256)%256;
            *(target_bitmap + bytes_per_pixel * (width * y + x - 2) + 0) = (total)%(256);

            count += 1;
            pixel_num++;
            //cout << "Now at " << sc_time_stamp() << " output result: " << 13 << endl; // print current sc_time
        }
```

下圖是 system.h，用來定義通道的位數和合成模塊。

```
#ifndef SYSTEM_H_
#define SYSTEM_H_
#include <systemc>
using namespace sc_core;

#include "Testbench.h"
#ifndef NATIVE_SYSTEMC
#include "SobelFilter_wrap.h"
#else
#include "SobelFilter.h"
#endif

class System: public sc_module
{
public:
  SC_HAS_PROCESS( System );
  System( sc_module_name n, std::string input_bmp, std::string output_bmp );
  ~System();
private:
  Testbench tb;
#ifndef NATIVE_SYSTEMC
  SobelFilter_wrapper sobel_filter;
#else
  SobelFilter sobel_filter;
#endif
  sc_clock clk;
  sc_signal<bool> rst;
#ifndef NATIVE_SYSTEMC
  cynw_p2p< sc_dt::sc_uint<24> > rgb;
  cynw_p2p< sc_dt::sc_uint<32> > result;
#else
  sc_fifo< sc_dt::sc_uint<24> > rgb;
  sc_fifo< sc_dt::sc_uint<32> > result;
#endif

  std::string _output_bmp;
};
#endif
```

這裡是 systempipeline.cpp，類似於 verilog 的 top 模塊，用於連接所有模塊，從 testbench 通過 rgb 通道進行 filter，再通過 result 通道返回 testbench，最後輸出

bmp。

```cpp
#include "System.h"
System::System( sc_module_name n, string input_bmp, string output_bmp ): sc_module( n ),
  tb("tb"), sobel_filter("sobel_filter"), clk("clk", CLOCK_PERIOD, SC_NS), rst("rst"), _output_bmp(output_bmp)
{
  tb.i_clk(clk);
  tb.o_rst(rst);
  sobel_filter.i_clk(clk);
  sobel_filter.i_rst(rst);
  tb.o_rgb(rgb);
  tb.i_result(result);
  sobel_filter.i_rgb(rgb);
  sobel_filter.o_result(result);

  tb.read_bmp(input_bmp);
}

System::~System() {
  tb.write_bmp(_output_bmp);
}
```

2. Improve coding styles

使用位寬來約束運算符。

```cpp
#ifndef FILTER_DEF_H_
#define FILTER_DEF_H_

#define MASK_N 2
#define MASK_X 3
#define MASK_Y 3
typedef sc_dt::sc_uint<24> input_t;
typedef sc_dt::sc_uint<32> output_t;
#endif
```

在 filter.cpp 裡將 rgb 轉換和 mean filter 的乘法改成 shift。

```cpp
for (int i = 0; i < MASK_Y * MASK_X; i++)
{
  if (i == 4)
  {
    mean_r = mean_r + (arr_r1[i]<<1) / 10;
    mean_g = mean_g + (arr_g1[i]<<1) / 10;
    mean_b = mean_b + (arr_b1[i]<<1) / 10;
  }
  else
  {
    mean_r = mean_r + arr_r1[i] / 10;
    mean_g = mean_g + arr_g1[i] / 10;
    mean_b = mean_b + arr_b1[i] / 10;
  }
}
mid_r = mean_r;
mid_g = mean_g;
mid_b = mean_b;
//mid_r = MeanFiter(arr_r1);
//mid_g = MeanFiter(arr_g1);
//mid_b = MeanFiter(arr_b1);

int total = 0;
total = (mid_r<<16) + (mid_g<<8) + mid_b ;
```

將 testbench 的除法改成 shift。

```
  *(target_bitmap + bytes_per_pixel * (width * y + x) + 2) = (total>>16);
  *(target_bitmap + bytes_per_pixel * (width * y + x) + 1) = (total>>8)%256;
  *(target_bitmap + bytes_per_pixel * (width * y + x) + 0) = (total)%(256);
```

3. Optimized Implementation

在 do_filter 裡的 while 前加上 while_1。

```
void SobelFilter::do_filter()
{
  unsigned char buffer [3][15];
  //vector<vector<unsigned char>> buffer(3, vector<unsigned char>(15, 0));
  // int x=0;
  int i = 0;
  //long long int a = 0;
  sc_time start_time;
  sc_time  end_time;
  {
#ifndef NATIVE_SYSTEMC
    HLS_DEFINE_PROTOCOL("main_reset");
    i_rgb.reset();
    o_result.reset();
#endif
    wait();
  }
while_1:
  while (true)
  {
    unsigned char val [3][9];
    unsigned char mean_r = 0, mean_g = 0, mean_b = 0;
    int mid_r = 0, mid_g = 0, mid_b = 0;
    //vector<vector<unsigned char>> val(3, vector<unsigned char>(9, 0));
    /*
    for (unsigned int i = 0; i<MASK_N; ++i) {
      HLS_CONSTRAIN_LATENCY(0, 1, "lat00");
      val[i] = 0;
    }*/
    for (unsigned int v = 0; v < MASK_Y; ++v)
    {
      for (unsigned int u = 0; u < MASK_X; ++u)
      {
        sc_dt::sc_uint<24> rgb;
#ifndef NATIVE_SYSTEMC
        {
          HLS_DEFINE_PROTOCOL("input");
          rgb = i_rgb.get();
          wait();
        }
#else
```

下圖是 HLS configuration，多加了 unroll_loops 和 constrain_latency 的指令

```
### 4. Define your HLS configuration (arbitrary names, BASIC and DPA in this example).
define_hls_config SobelFilter BASIC
define_hls_config SobelFilter DPA       --dpopt_auto=op,expr
define_hls_config SobelFilter UNROLL_ALL --flatten_arrays=all -post_elab_tcl {
    unroll_loops [find -loop "*_loop" ]
    constrain_latency -max_lat $HLS::ACHIEVABLE [find -loop "while_1"]
}
```

下圖是 simulation 對 V_UNROLL_ALL。

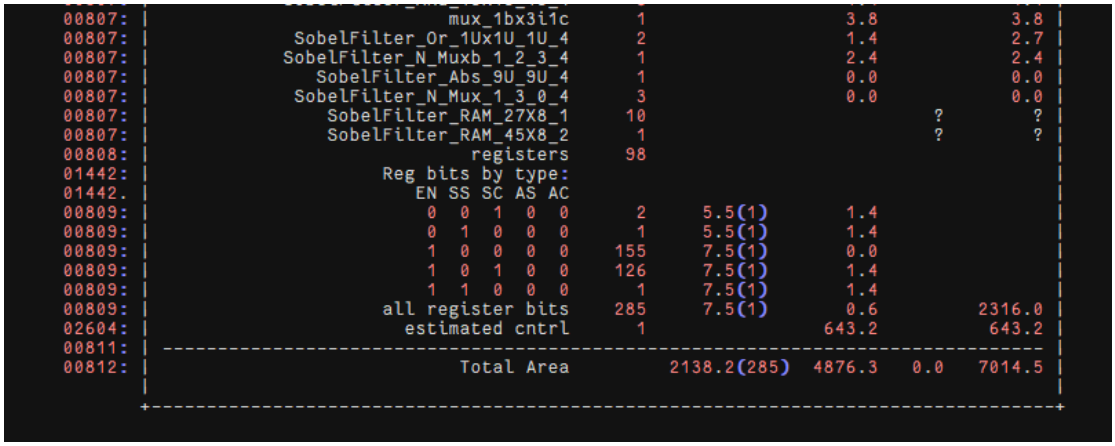
```
### 5. Define simulation configuration for each HLS configuration
### 5.1 The behavioral simulation (C++ only).
define_sim_config B -argv "$IN_FILE_NAME $OUT_FILE_NAME"
### 5.2 The Verilog simulation for HLS config "BASIC".
define_sim_config V_BASIC "SobelFilter RTL_V BASIC" -argv "$IN_FILE_NAME $OUT_FILE_NAME"
### 5.3 The Verilog simulation for HLS config "DPA".
define_sim_config V_DPA "SobelFilter RTL_V DPA" -argv "$IN_FILE_NAME $OUT_FILE_NAME"
###
define_sim_config V_UNROLL_ALL "SobelFilter RTL_V UNROLL_ALL" -argv "$IN_FILE_NAME $OUT_FILE_NAME"
```

比較(BASIC):

1.256*256 大小的圖片:

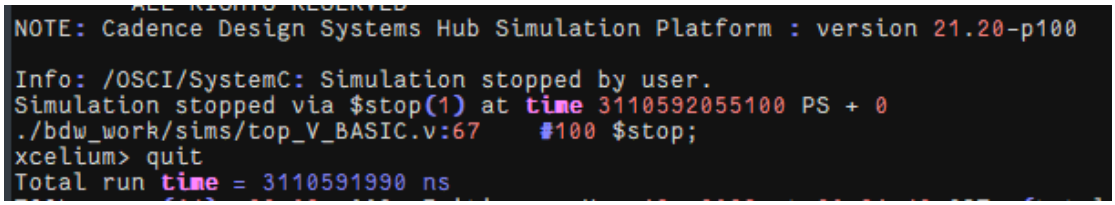
(1)Base Implementation:

Area:



```
00807: |                          mux_1bx3i1c      1                3.8           3.8 |
00807: |             SobelFilter_Or_1Ux1U_1U_4     2                1.4           2.7 |
00807: |           SobelFilter_N_Muxb_1_2_3_4      1                2.4           2.4 |
00807: |              SobelFilter_Abs_9U_9U_4      1                0.0           0.0 |
00807: |            SobelFilter_N_Mux_1_3_0_4      3                0.0           0.0 |
00807: |               SobelFilter_RAM_27X8_1     10                              ?   ? |
00807: |               SobelFilter_RAM_45X8_2      1                              ?   ? |
00808: |                            registers     98                                   |
01442: |                      Reg bits by type:                                        |
01442. |                         EN SS SC AS AC                                        |
00809: |                          0  0  1  0  0      2     5.5(1)       1.4            |
00809: |                          0  1  0  0  0      1     5.5(1)       1.4            |
00809: |                          1  0  0  0  0    155     7.5(1)       0.0            |
00809: |                          1  0  1  0  0    126     7.5(1)       1.4            |
00809: |                          1  1  0  0  0      1     7.5(1)       1.4            |
00809: |                      all register bits    285     7.5(1)       0.6      2316.0 |
02604: |                        estimated cntrl      1                643.2       643.2 |
00811: | ------------------------------------------------------------------------------ |
00812: |                               Total Area         2138.2(285)  4876.3   0.0  7014.5 |
       |                                                                               |
       +-------------------------------------------------------------------------------+
```

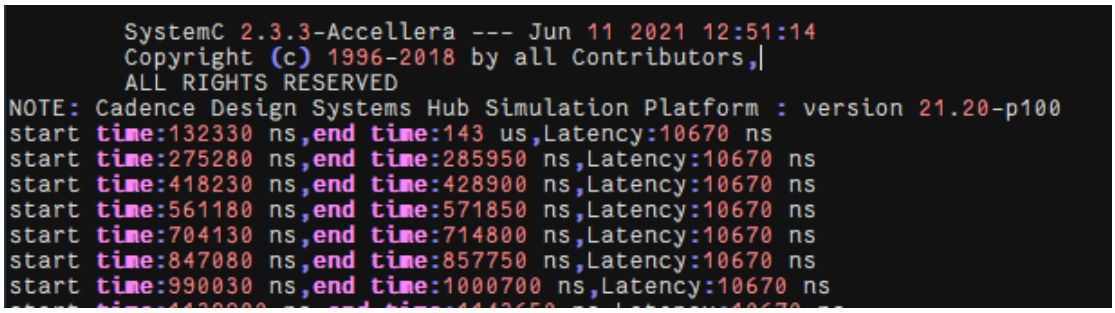
Run time:



```
NOTE: Cadence Design Systems Hub Simulation Platform : version 21.20-p100

Info: /OSCI/SystemC: Simulation stopped by user.
Simulation stopped via $stop(1) at time 3110592055100 PS + 0
./bdw_work/sims/top_V_BASIC.v:67    #100 $stop;
xcelium> quit
Total run time = 3110591990 ns
```

Latency:

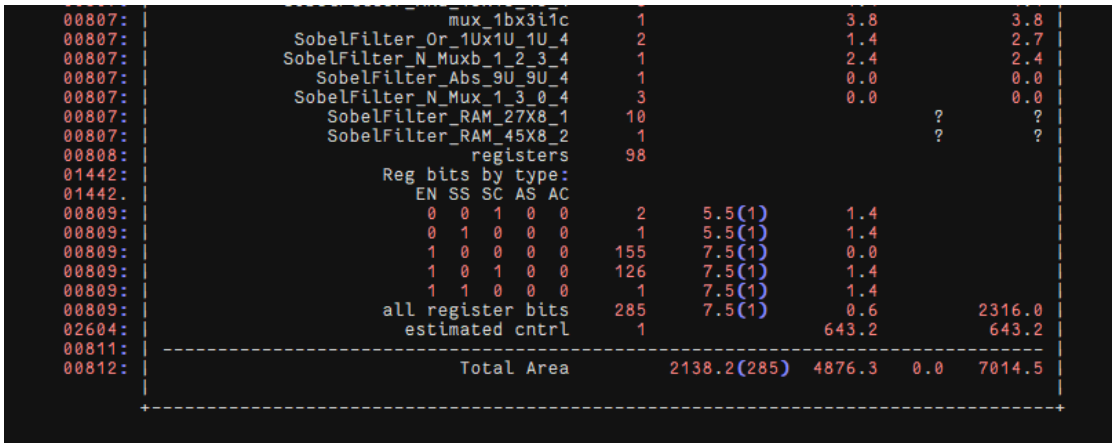


```
        SystemC 2.3.3-Accellera --- Jun 11 2021 12:51:14
        Copyright (c) 1996-2018 by all Contributors,
        ALL RIGHTS RESERVED
NOTE: Cadence Design Systems Hub Simulation Platform : version 21.20-p100
start time:132330 ns,end time:143 us,Latency:10670 ns
start time:275280 ns,end time:285950 ns,Latency:10670 ns
start time:418230 ns,end time:428900 ns,Latency:10670 ns
start time:561180 ns,end time:571850 ns,Latency:10670 ns
start time:704130 ns,end time:714800 ns,Latency:10670 ns
start time:847080 ns,end time:857750 ns,Latency:10670 ns
start time:990030 ns,end time:1000700 ns,Latency:10670 ns
```

Throughtput=281,162 pxiel/s。

(2) **Improve coding styles**

Area:



Run time:

```
Info: /OSCI/SystemC: Simulation stopped by user.
Simulation stopped via $stop(1) at time 3110592055100 PS + 0
/home/m111/m111064503/EE6470/Hw3/sobel_stratus_improve/stratus/bdw_work/sims/top_V_BASIC.v:67
xcelium> quit
Total run time = 3110591990 ns
```

Latency:

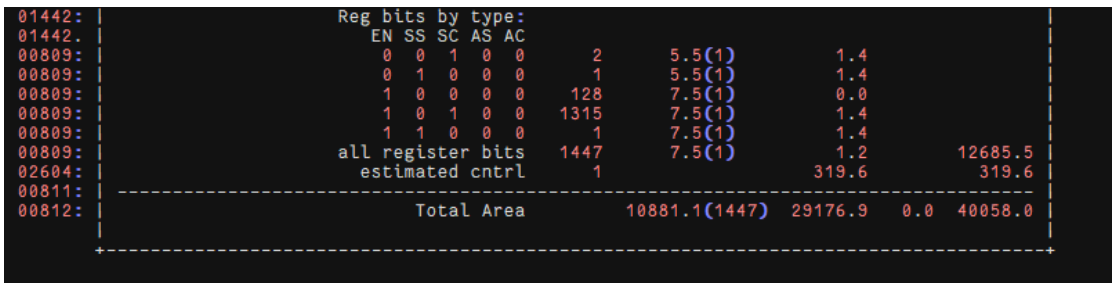
```
        SystemC 2.3.3-Accellera --- Jun 11 2021 12:51:14
        Copyright (c) 1996-2018 by all Contributors,
        ALL RIGHTS RESERVED
NOTE: Cadence Design Systems Hub Simulation Platform : version 21.20-p100
start time:132330 ns,end time:143 us,Latency:10670 ns
start time:275280 ns,end time:285950 ns,Latency:10670 ns
start time:418230 ns,end time:428900 ns,Latency:10670 ns
start time:561180 ns,end time:571850 ns,Latency:10670 ns
start time:704130 ns,end time:714800 ns,Latency:10670 ns
start time:847080 ns,end time:857750 ns,Latency:10670 ns
start time:990030 ns,end time:1000700 ns,Latency:10670 ns
```

Throughtput=281,162 pxiel/s。

將乘法改成 shift 後，優化過的 code 合出來的 area、latency、run time、throughput 都一樣，我認為應該是 tool 已經能自己優化乘除法器了，所以原本是用*/的部分和 shift 合出來的結果會一樣。

(3) **Optimized Implementation**

Area:



Run time:

```
Info: /OSCI/SystemC: Simulation stopped by user.
Simulation stopped via $stop(1) at time 434112055100 PS + 0
/home/m111/m111064503/EE6470/Hw3/sobel_stratus_optimal/stratus/bdw_work/sims/top_V_UNROLL_ALL.v:67
xcelium> quit
Total run time = 434111990 ns
```

Latency:

```
        SystemC 2.3.3-Accellera --- Jun 11 2021 12:51:14
        Copyright (c) 1996-2018 by all Contributors,
        ALL RIGHTS RESERVED
NOTE: Cadence Design Systems Hub Simulation Platform : version 21.20-p100
start time:18410 ns,end time:20 us,Latency:1590 ns
start time:38360 ns,end time:39950 ns,Latency:1590 ns
start time:58310 ns,end time:59900 ns,Latency:1590 ns
start time:78260 ns,end time:79850 ns,Latency:1590 ns
start time:98210 ns,end time:99800 ns,Latency:1590 ns
start time:118160 ns,end time:119750 ns,Latency:1590 ns
start time:138110 ns,end time:139700 ns,Latency:1590 ns
start time:158060 ns,end time:159650 ns,Latency:1590 ns
start time:178010 ns,end time:179600 ns,Latency:1590 ns
start time:197960 ns,end time:199550 ns,Latency:1590 ns
```

Throughtput=1,886,792 pxiel/s。

使用 unroll 後，面積暴增到了 4 萬，因為將 loop 迴圈展開，所以面積會比原本多了 6 倍多，但 latency 也少了 6 倍多，run time 則是快了 7 倍多，throughtput 則多了 6 倍多，所以這邊是以面積換取速度。

2.512*512 大小的圖片：

(2)Base Implementation:

Area:

```
00807: |                          mux_1bx3i1c      1                3.8            3.8 |
00807: |              SobelFilter_Or_1Ux1U_1U_4      2                1.4            2.7 |
00807: |            SobelFilter_N_Muxb_1_2_3_4      1                2.4            2.4 |
00807: |               SobelFilter_Abs_9U_9U_4      1                0.0            0.0 |
00807: |             SobelFilter_N_Mux_1_3_0_4      3                0.0            0.0 |
00807: |                SobelFilter_RAM_27X8_1     10                                ?   ? |
00807: |                SobelFilter_RAM_45X8_2      1                                ?   ? |
00808: |                             registers     98                                    |
01442: |                     Reg bits by type:                                          |
01442. |                        EN SS SC AS AC                                          |
00809: |                         0  0  1  0  0      2     5.5(1)       1.4              |
00809: |                         0  1  0  0  0      1     5.5(1)       1.4              |
00809: |                         1  0  0  0  0    155     7.5(1)       0.0              |
00809: |                         1  0  1  0  0    126     7.5(1)       1.4              |
00809: |                         1  1  0  0  0      1     7.5(1)       1.4              |
00809: |                    all register bits    285     7.5(1)       0.6         2316.0 |
02604: |                      estimated cntrl      1                643.2          643.2 |
00811: | ------------------------------------------------------------------------------- |
00812: |                            Total Area         2138.2(285)  4876.3   0.0   7014.5 |
       |                                                                                |
       +--------------------------------------------------------------------------------+
```

Run time:

```
Info: /OSCI/SystemC: Simulation stopped by user.
Simulation stopped via $stop(1) at time 12442368055100 PS + 0
./bdw_work/sims/top_V_BASIC.v:67    #100 $stop;
xcelium> quit
Total run time = 12442367990 ns
```

Latency:

```
        SystemC 2.3.3-Accellera --- Jun 11 2021 12:51:14
        Copyright (c) 1996-2018 by all Contributors,
        ALL RIGHTS RESERVED
NOTE: Cadence Design Systems Hub Simulation Platform : version 21.20-p100
start time:132330 ns,end time:143 us,Latency:10670 ns
start time:275280 ns,end time:285950 ns,Latency:10670 ns
start time:418230 ns,end time:428900 ns,Latency:10670 ns
start time:561180 ns,end time:571850 ns,Latency:10670 ns
start time:704130 ns,end time:714800 ns,Latency:10670 ns
start time:847080 ns,end time:857750 ns,Latency:10670 ns
start time:990030 ns,end time:1000700 ns,Latency:10670 ns
```

Throughtput=281,162 pxiel/s。

(2) **Improve coding styles**

Area:

```
00807: |                          mux_1bx3i1c      1                3.8           3.8 |
00807: |              SobelFilter_Or_1Ux1U_1U_4      2                1.4           2.7 |
00807: |             SobelFilter_N_Muxb_1_2_3_4      1                2.4           2.4 |
00807: |                SobelFilter_Abs_9U_9U_4      1                0.0           0.0 |
00807: |              SobelFilter_N_Mux_1_3_0_4      3                0.0           0.0 |
00807: |                 SobelFilter_RAM_27X8_1     10                              ?   ? |
00807: |                 SobelFilter_RAM_45X8_2      1                              ?   ? |
00808: |                              registers     98                                  |
01442: |                     Reg bits by type:                                          |
01442. |                        EN SS SC AS AC                                          |
00809: |                         0  0  1  0  0      2     5.5(1)       1.4              |
00809: |                         0  1  0  0  0      1     5.5(1)       1.4              |
00809: |                         1  0  0  0  0    155     7.5(1)       0.0              |
00809: |                         1  0  1  0  0    126     7.5(1)       1.4              |
00809: |                         1  1  0  0  0      1     7.5(1)       1.4              |
00809: |                   all register bits    285     7.5(1)       0.6        2316.0 |
02604: |                     estimated cntrl      1                643.2         643.2 |
00811: | ------------------------------------------------------------------------------ |
00812: |                           Total Area         2138.2(285)  4876.3   0.0  7014.5 |
       |                                                                              |
       +------------------------------------------------------------------------------+
```

Run time:

```
Info: /OSCI/SystemC: Simulation stopped by user.
Simulation stopped via $stop(1) at time 12442368055100 PS + 0
./bdw_work/sims/top_V_BASIC.v:67     #100 $stop;
xcelium> quit
Total run time = 12442367990 ns
```

Latency:

```
        SystemC 2.3.3-Accellera --- Jun 11 2021 12:51:14
        Copyright (c) 1996-2018 by all Contributors,
        ALL RIGHTS RESERVED
NOTE: Cadence Design Systems Hub Simulation Platform : version 21.20-p100
start time:132330 ns,end time:143 us,Latency:10670 ns
start time:275280 ns,end time:285950 ns,Latency:10670 ns
start time:418230 ns,end time:428900 ns,Latency:10670 ns
start time:561180 ns,end time:571850 ns,Latency:10670 ns
start time:704130 ns,end time:714800 ns,Latency:10670 ns
start time:847080 ns,end time:857750 ns,Latency:10670 ns
start time:990030 ns,end time:1000700 ns,Latency:10670 ns
```

Throughtput=281,162 pxiel/s。

將乘法改成 shift 後，優化過的 code 合出來的 area、latency、run time、throughput 都一樣，我認為應該是 tool 已經能自己優化乘除法器了，所以原本是用*/的部分和 shift 合出來的結果會一樣。

(3) **Optimized Implementation**

Area:

```
01442: |                    Reg bits by type:                                          |
01442. |                       EN SS SC AS AC                                          |
00809: |                        0  0  1  0  0       2       5.5(1)         1.4         |
00809: |                        0  1  0  0  0       1       5.5(1)         1.4         |
00809: |                        1  0  0  0  0     128       7.5(1)         0.0         |
00809: |                        1  0  1  0  0    1315       7.5(1)         1.4         |
00809: |                        1  1  0  0  0       1       7.5(1)         1.4         |
00809: |                    all register bits    1447       7.5(1)         1.2   12685.5 |
02604: |                      estimated cntrl       1                     319.6     319.6 |
00811: | ------------------------------------------------------------------------------ |
00812: |                             Total Area          10881.1(1447)  29176.9   0.0  40058.0 |
       |                                                                              |
       +------------------------------------------------------------------------------+
```

Run time:

```
Info: /OSCI/SystemC: Simulation stopped by user.
Simulation stopped via $stop(1) at time 1736448055100 PS + 0
./bdw_work/sims/top_V_UNROLL_ALL.v:67      #100 $stop;
xcelium> quit
Total run time = 1736447990 ns
```

Latency:

```
        SystemC 2.3.3-Accellera --- Jun 11 2021 12:51:14
        Copyright (c) 1996-2018 by all Contributors,
        ALL RIGHTS RESERVED
NOTE: Cadence Design Systems Hub Simulation Platform : version 21.20-p100
start time:18410 ns,end time:20 us,Latency:1590 ns
start time:38360 ns,end time:39950 ns,Latency:1590 ns
start time:58310 ns,end time:59900 ns,Latency:1590 ns
start time:78260 ns,end time:79850 ns,Latency:1590 ns
start time:98210 ns,end time:99800 ns,Latency:1590 ns
start time:118160 ns,end time:119750 ns,Latency:1590 ns
start time:138110 ns,end time:139700 ns,Latency:1590 ns
start time:158060 ns,end time:159650 ns,Latency:1590 ns
start time:178010 ns,end time:179600 ns,Latency:1590 ns
start time:197960 ns,end time:199550 ns,Latency:1590 ns
```

Throughtput=1,886,792 pxiel/s。

使用 unroll 後，面積暴增到了 4 萬，因為將 loop 迴圈展開，所以面積會比原本多了 6 倍多，但 latency 也少了 6 倍多，run time 則是快了 7 倍多，throughtput 則多了 6 倍多，所以這邊是以面積換取速度。