/
/
/
/
/
/
/
/
/
/
/
/
/
/
/

/
/

/
/
/
/
/

//
/
/
/
.
/
/

//
/
/
/
/
/
/
/
/
/

組別:2
組員: 劉祐瑋、陳昇達、劉佩雯
Lab: 2_2
HackMD link: https://hackmd.io/6kZrHyqzTtKPoeHpqlZlZw?view (https://hackmd.io/6kZrHyqzTtKPoeHpqlZlZw?view) (歡迎利用此閱讀)
Github link: https://github.com/nthuyouwei/asoclab/tree/main/lab02 (https://github.com/nthuyouwei/asoclab/tree/main/lab02)

lab2_01_edgedetect

略(如workbook 介紹)

lab2_02_edgedetect_fsic

how we design our work

這部分說明建立在了解lab2_01_edgedetect下做說明

根據新的Spec我們可以畫出新的架構圖如下:

所以我們可以改原本的EdgeDetech.h 如下:

```
#pragma once

#include "EdgeDetect_defs.h"
#include "EdgeDetect_VerDer.h"
#include "EdgeDetect_HorDer.h"
#include "EdgeDetect_MagAng.h"
#include <mc_scverify.h>

namespace EdgeDetect_IP {

  #pragma hls_design top
  class EdgeDetect_Top
  {
    //instances
    EdgeDetect_VerDer VerDer_inst;
    EdgeDetect_HorDer HorDer_inst;
    EdgeDetect_MagAng MagAng_inst;

    // Static interconnect channels (FIFOs) between blocks
    ac_channel<gradType4x>       dy;
    ac_channel<gradType4x>       dx;
    ac_channel<pixelType4x>      dat_in_hor; // channel for passing input pixels to horizontalDerivative
    ac_channel<pixelType4x>      dat_in_mag;

  public:
    EdgeDetect_Top() {}

    //--------------------------------------------------------------------------
    // Function: run
    //   Top interface for data in/out of class. Combines vertical and
    //   horizontal derivative and magnitude/angle computation.
    #pragma hls_design interface
    void CCS_BLOCK(run)(maxWType             &widthIn,
                        maxHType             &heightIn,
                        bool                 &sw_in,
                        uint32t              &crc32_hw_pix_in,
                        uint32               &crc32_hw_pix_out,
                        ac_channel<Stream_t> &din_chn,
                        ac_channel<Stream_t> &dout_chn)
    {
      VerDer_inst.run(dat_in, widthIn, heightIn, dat_in_hor, dy);
      HorDer_inst.run(dat_in_hor, widthIn, heightIn, dx, dat_in_mag);
      MagAng_inst.run(dx, dy, widthIn, heightIn,dat_in_mag,sw_in,crc32_hw_pix_in ,crc32_hw_pix_out,dout_chn);
    }
  };

}
```

首先，我們需要把data width變寬4倍來達到4 pixels per cycle ，由 gradType 和 pixelType 更改為 gradType4x 和 pixelType4x。除此之外，interface 還引入了額外的控制參數sw_in(這是為了讓我們可以選取模式:select the output source from input image or the calculated magnitude)，以及兩個用於CRC校驗的輸出（ crc32_hw_pix_in 和 crc32_hw_pix_out ）。而在interconnect的部分，為了做crc以及決定是否直接輸出input image，我們需要加dat_in_mag這條通道來傳遞pixel至Mag(原先只要傳遞dx)。

故我們在EdgeDetect_defs.h中定義了新的typedef

```
#pragma once

#include <ac_int.h>
#include <ac_fixed.h>
#include <ac_channel.h>
#include <ac_math/ac_sqrt_pwl.h>
#include <ac_math/ac_atan2_cordic.h>
#include <ac_math.h>

//#define USE_CIRCULARBUF


  const int maxImageWidth = 640;
  const int maxImageHeight = 480;

  const int kernel1[3] = {1, 0, -1};

  // Define some bit-accurate types to use in this model
  typedef uint8                    pixelType;     // input pixel is 0-255
  typedef uint16                   pixelType2x;   // two pixels packed
  typedef uint32                   pixelType4x;
  typedef ac_int<64,false>         pixelType8x;
  typedef int9                     gradType;      // Derivative is max range -255 to 255
  typedef int36                    gradType4x;
  typedef uint18                   sqType;        // Result of 9-bit x 9-bit
  typedef ac_fixed<19,19,false>    sumType;       // Result of 18-bit + 18-bit fixed pt integer for squareroot
  typedef uint8                    magType;       // 8-bit unsigned magnitute result
  typedef uint32                   magType4x;
  typedef ac_fixed<8,3,true>       angType;       // 3 integer bit, 5 fractional bits for quantized angle -pi to pi

  // Compute number of bits for max image size count, used internally and in testbench
  typedef ac_int<ac::nbits<maxImageWidth+1>::val,false> maxWType;
  typedef ac_int<ac::nbits<maxImageHeight+1>::val,false> maxHType;

  struct Stream_t{
  pixelType4x pix;
  bool sof;
  bool eol;
  };
```

注意數據結構Stream_t，我是根據testbench來定義，其中sof代表起始幀標誌、eol代表行結束標誌:

```
ac_channel</*EdgeDetect_IP::*/Stream_t>  din_chn;
ac_channel</*EdgeDetect_IP::*/Stream_t>  dout_chn;

/*EdgeDetect_IP::*/Stream_t dat;
```

```
 dat.pix = pix4;
 dat.sof = (x==0 && y==0);
 dat.eol = (x==width-4);
 din_chn.write(dat);
```

接者我們進到 module Ver 來討論，02中的EdgeDetect_Verder.h跟01中EdgeDetect_Verder.h雷同，我們只需注意這時候一個for loop裡面是計算四筆pixel，並且一次傳出四筆的data和dy。改動內容如下:

- Buffer tmp 大小都變4倍:

```
pixelType8x line_buf0[maxImageWidth/8];
pixelType8x line_buf1[maxImageWidth/8];
pixelType8x rdbuf0_pix, rdbuf1_pix;
pixelType8x wrbuf0_pix, wrbuf1_pix;
pixelType4x pix0, pix1, pix2;
gradType4x pix;
maxWType x4;
Stream_t data;
```

- index x ，做完一次其實做了4筆所以要改成 x+=4，且break 的部分也要改成 widthIn-4:

```
VCOL: for (maxWType x = 0; ; x+=4)
```

```
if (x == maxWType(widthIn-4)) { // cast to maxWType for RTL code coverage
  break;
```

- add index x4 = x/4 ，我們一樣要利用奇偶數來判斷要從哪裡放的位置，但因為每次x會+4無法使用他來判斷，故我們利用x4=x/4，來取代原本x在for迴圈中的進行演算法的位置。

```
x4=x/4;
if (y <= heightIn-1) {
  data=  din_chn.read();
  pix0 = data.pix; // Read streaming interface
}
// Write data cache, write lower 8 on even iterations of COL loop, upper 8 on odd
if ( (x4&1) == 0 ) {
  wrbuf0_pix.set_slc(0,pix0);
} else {
  wrbuf0_pix.set_slc(32,pix0);
}
// Read line buffers into read buffer caches on even iterations of COL loop
if ( (x4&1) == 0 ) {
  // vertical window of pixels
  rdbuf1_pix = line_buf1[x4/2];
  rdbuf0_pix = line_buf0[x4/2];
} else { // Write line buffer caches on odd iterations of COL loop
  line_buf1[x4/2] = rdbuf0_pix; // copy previous line
  line_buf0[x4/2] = wrbuf0_pix; // store current line
}
// Get 8-bit data from read buffer caches, lower 8 on even iterations of COL loop
pix2 = ((x4&1)==0) ? rdbuf1_pix.slc<32>(0) : rdbuf1_pix.slc<32>(32);
pix1 = ((x4&1)==0) ? rdbuf0_pix.slc<32>(0) : rdbuf0_pix.slc<32>(32);

// Boundary condition processing
if (y == 1) {
  pix2 = pix1; // top boundary (replicate pix1 up to pix2)
}
if (y == heightIn) {
  pix0 = pix1; // bottom boundary (replicate pix1 down to pix0)
}
```

- 計算部分，一次計算四筆，並且利用set_slc來存進去對應的位置:

```
#pragma hls_unroll yes
for(int i=0;i<4;i++){
gradType tmp;
tmp=pix2.slc<8>(i*8)*kernel1[0]+pix1.slc<8>(i*8)*kernel1[1]+pix0.slc<8>(i*8)*kernel1[2];
pix.set_slc(i*9,tmp);
}
```

- 輸出部分:跟01一樣

```
if (y != 0) { // Write streaming interfaces
  dat_in_hor.write(pix1); // Pass thru original data
  dy.write(pix); // derivative output
}
```

再接者我們進到 module Hor 來討論，02中的EdgeDetect_Horder.h跟01中EdgeDetect_Horder.h雷同，我們只需注意這時候一個for loop裡面是計算四筆pixel，並且一次傳出四筆的data(原本在01中不須要傳)和dx。改動內容如下:

- Buffer tmp 大小都變4倍，除此之外原先01中是利用pix0、pix1 、pix2來暫存做運算，但因為我們有4筆資料需要運算， 故4(舊)+4(新)+1(保留一個tmp給舊的，新的要用到)總共要9個來運算，所以我們改成p\([9]\)(注意這裡只要用一個pixel的大小就好了)，以及增加一個pixel 的暫存 pix:

```
pixelType p[9];
pixelType4x pix;
gradType4x  grad;
```

- index x ，做完一次其實做了4筆所以要改成 x+=4，且再判斷read data、left and right boundary 和break時也要適當的更改:

```
HCOL: for (maxWType x = 0; ; x+=4)
```

```
if (x <= (widthIn - 4))
  pix = dat_in.read();
```

```
if (x == 4) //left boundary
```

```
else if (x == widthIn) //right boundary
```

```
if ( x == widthIn) {
  break;
}
```

- 計算部分，一次計算四筆，並且利用set_slc來存進去對應的位置:

```
// Calculate derivative
#pragma hls_unroll yes
for(int i=0; i < 4; i++)
{
  gradType grad_tmp;
  grad_tmp = p[i]*kernel1[0] + p[i+1]*kernel1[1] + p[i+2]*kernel1[2];
  grad.set_slc(i*9, grad_tmp);
}
```

- 如何利用p[9]來計算，跟01一樣我們要分為三種狀況right boundary case和left boundary case 還有中間的case，因為要等兩筆資料寫進去我們才會算第一筆的資料(因為算四筆需要4+2(前後)筆)。其中p[0]到p[8]是由舊到新，所以在中間case，我們

需要先做shift再寫入新的pixel至p[5]~p[8]。除此之外我們要把p[4]也shift至p\([0]\)(因為四筆data需要前後兩筆共六筆來計算，故我們要保留p[4])。然後會有right boundary case 和left boundary case (就是在算第一筆還有最後一筆時會用到)只是把p[0]和p[5]利用鏡像填充p[2]跟p3 ()。

```
#pragma hls_unroll yes
for (int i=0; i < 5; i++)
  p[i] = p[i+4];

#pragma hls_unroll yes
for (int i=0; i < 4; i++)
  p[i+5] = pix.slc<8>(i*8);

if (x==4){
 p[0]=p[2];
}
if (x==widthIn){
 p[5]=p[3];
}
```

- 輸出部分:我們每次計算完的p[1]至p[4]我們要再次把他輸出傳進Mag(在01中並不需要)以及我們所算的dx

```
if (x != 0) { // Write streaming interface
  pixelType4x p_tmp;
  for (int i=0;i<4;i++){
  p_tmp.set_slc(8*i,p[i+1]);
  }

  dat_out.write(p_tmp);
  dx.write(grad); // derivative out
}
```

緊接者我們進到 module Mag 來討論，這裡跟01有很大的不同，首先我們刪掉了angle的計算，並且加入了crc32計算，除此之外我們還加入sw_in來選擇輸出(the output source from input image or the calculated magnitude)，以及不同01利用square root來計算mag，在02中我們將會利用sum of absolute difference 來計算。改動內容如下:

- Buffer tmp 大小都變4倍，且增加crc32的tmp。除此之外因為輸出要Stream_t架構，所以我們到時也需要轉換。

```
gradType4x dx, dy;
pixelType4x pix;
magType4x magn;
magType4x magn_out;
Stream_t dat;
uint32 crc32_pix_in_tmp = 0XFFFFFFFF;
uint32 crc32_dat_out_tmp = 0XFFFFFFFF;
```

- index x ，做完一次其實做了4筆所以要改成 x+=4，且再判斷dat.eol和break也要適當的更改:

```
MCOL: for (maxWType x = 0; ; x+=4)
```

```
dat.eol = (x== maxWType(widthIn-4));
```

```
if (x == maxWType(widthIn-4)) { // cast to maxWType for RTL code coverage
  break;
}
```

- 計算magn，使用 ac_abs 函數計算 dx 和 dy 中每個像素梯度值的絕對值，然後利用 ac_fixed來處理數值溢出，最後利用.to_uint()來轉換type。

```
for(int i=0; i < 4; i++)
{
  ac_math::ac_abs(dx.slc<9>(i*9), abs_dx);
  ac_math::ac_abs(dy.slc<9>(i*9), abs_dy);
  uint9 abs_sum = abs_dx + abs_dy;
  ac_fixed<8,8,false,AC_TRN,AC_SAT> abs_sum_clip = abs_sum;
  magType tmp = (magType) abs_sum_clip.to_uint();
  magn.set_slc(i*8, tmp);
}
```

- sw_in 決定輸出:

```
if (!sw_in)
  magn_out = pix;
else
  magn_out = magn;
```

- 加入 crc32 (crc32 github 有提供)

```
crc32_pix_in_tmp  = calc_crc32<32>(crc32_pix_in_tmp,  pix);
crc32_dat_out_tmp = calc_crc32<32>(crc32_dat_out_tmp, magn_out);
```

- 符合最一開始講到的Stream_t架構，並輸出:

```
dat.pix = magn_out;
dat.sof = (x==0 && y==0);
dat.eol = (x== maxWType(widthIn-4));

dat_out.write(dat);
```

What's the test result of catapult design

Run design

首先我們可以利用我放在github中的directive.tcl 來run 整體design 包括設計FIFO 深度、pipeline等等，這部分如同01中的說明。

```
catapult -f directive.tcl
```

Result

C simlation (log file in github)

```
# Simulating design
# cd ../..; ./Catapult/EdgeDetect_Top.v1/scverify/orig_cxx_osci/scverify_top ./image/people640x360_rgb.bmp 1 out_algorithm.bmp out_hw.bmp
# Loading Input File
# Input file:          ./image/people640x360_rgb.bmp
# Mode:                1
# Output file (alg):   out_algorithm.bmp
# Output file (hw):    out_hw.bmp
# Image width: 640
# Image height: 360
# ############################################ FRAME NO.   0 ###############################################
# Running
# Magnitude: Manhattan norm per pixel 5.357491
# Writing algorithmic bitmap output to: out_algorithm.bmp
# Writing bit-accurate bitmap output to: out_hw.bmp
# sofErr: 0 eolErr: 0
# crc32_alg_pix_in  = ebb44e76  crc32_hw_pix_in  = ebb44e76
# crc32_alg_dat_out = 398625ad  crc32_hw_dat_out = 49e564fe
# Finished
```

C design checker

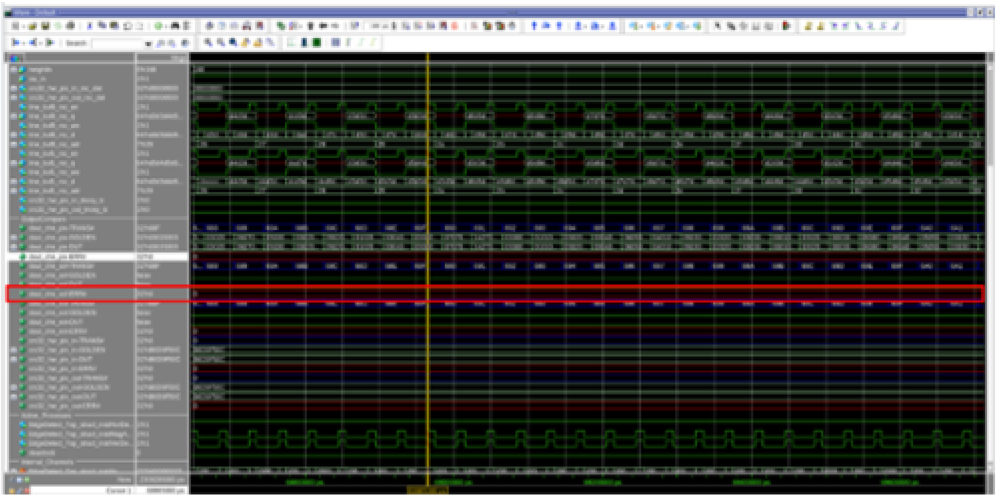
```
FATAL                                                    Violated     Waived  Undecided
------------------------------------------------------ ---------- ---------- ----------

ERROR                                                    Violated     Waived  Undecided
------------------------------------------------------ ---------- ---------- ----------
ABR - Array Bounds Read                                         2          0          0
ABW - Array Bounds Write                                        2          0          0
AOB - Arithmetic Operator with Boolean                          0          0          0
CAS - Incomplete Switch-Case                                    0          0          0
DBZ - Divide By Zero                                            0          0          0
ISE - Illegal Shift Error                                       0          0          0
OVL - Overflow/Underflow                                        4          0          3
RRT - Reset referenced in thread                                0          0          0
UMR - Uninitialized Memory Read                                 5          0          0

WARNING                                                  Violated     Waived  Undecided
------------------------------------------------------ ---------- ---------- ----------
ACC - Accumulator of native C type                              0          0          0
ACS - Accumulator of saturated type                             0          0          0
AIC - Assignment used Instead of Comparison                     0          0          0
ALS - Ac_int Left Shift check                                   0          0          0
AWE - Assignments Without Effect                                0          0          0
CBU - Conditional break in Unrolled Loop                        0          0          0
CCC - Static constant comparison                                0          0          0
CGR - Conditional Guard in Rolled Loop                          0          0          0
CIA - Comparison Instead of Assignment                          0          0          0
CNS - Constant condition of if/switch                           0          0          0
CWB - Case Without Break                                        0          0          0
DIU - Dynamic Index in Unrolled Loop                            0          0          0
FVI - For Loop with Variable Iterations                         6          0          0
FXD - Mixed fixed and non-fixed datatypes                       1          0          0
MDB - Missing Default Branch                                    0          0          0
NCO - No Contribution to Output                                 4          0          0
OSA - Optimal Size Accumulator                                  0          0          0
PDD - Platform dependent datatype (long)                        0          0          0
RIU - Rolled loop Inside Unrolled loop                          0          0          0
SAT - Sub-optimal Adder Tree                                    0          0          0
SUD - Suboptimal Use of Divide and Modulus Operator             0          0          0
```

Questasim result

report_txt:

```
# Checking results
# 'crc32_hw_pix_in'
#    capture count        = 1
#    comparison count     = 1
#    ignore count         = 0
#    error count          = 0
#    stuck in dut fifo    = 0
#    stuck in golden fifo = 0
# 'crc32_hw_pix_out'
#    capture count        = 1
#    comparison count     = 1
#    ignore count         = 0
#    error count          = 0
#    stuck in dut fifo    = 0
#    stuck in golden fifo = 0
# 'dout_chn_pix'
#    capture count        = 57600
#    comparison count     = 57600
#    ignore count         = 0
#    error count          = 0
#    stuck in dut fifo    = 0
#    stuck in golden fifo = 0
# 'dout_chn_sof'
#    capture count        = 57600
#    comparison count     = 57600
#    ignore count         = 0
#    error count          = 0
#    stuck in dut fifo    = 0
#    stuck in golden fifo = 0
# 'dout_chn_eol'
#    capture count        = 57600
#    comparison count     = 57600
#    ignore count         = 0
#    error count          = 0
#    stuck in dut fifo    = 0
#    stuck in golden fifo = 0
#
# Info: scverify_top/user_tb: Simulation PASSED @ 2309286 ns
# ** Note: (vsim-6574) SystemC simulation stopped by user.
# 1
```

waveform(紅框為error=0):



其他rtl.rpt resource usage

都放在github了

lab2_03_fsic_prj

how we integrate our design in FSIC

在完成 lab2_02_edgedetect_fsic 生成 RTL code 之後，我們會得到一個 concat_EdgeDetect_Top.v 的檔案，我們要將它放到 fsic 的環境去跑simulation，而 fsic 的環境所需要的檔案可以從 filelist 中得知


```
../fpga.v
../../rtl/axi_ctrl_logic.sv
../../rtl/axil_axis.sv
../../rtl/axilite_master.sv
../../rtl/axilite_slave.sv
../../rtl/axis_master.sv
../../rtl/axis_slave.sv
// ../../rtl/axis_switch.v
../../rtl/sw_caravel.v
../../rtl/config_ctrl.v
../../rtl/fsic_clkrst.v
../../rtl/fsic_clock.v
../../rtl/fsic_coreclk_phase_cnt.v
../../rtl/fsic_io_serdes_rx.v
../../rtl/fsic.v
../../rtl/io_serdes.v
../../rtl/logic_anlz.dummy_io.v
../../rtl/logic_anlz.dummy_io.vd
// ../../rtl/mprj_io.dummy_io.v
// ../../rtl/mprj_io.dummy_io.vd
../../rtl/mprj_io.sv
../../rtl/user_subsys.all.v
../../rtl/user_prj0.v
../../rtl/user_prj1.v
../../rtl/user_prj2.v
../../rtl/user_prj3.v
../../rtl/spram.v
../../rtl/concat_EdgeDetect_Top_fsic.v
```

而這些檔案可以從 Lab1 fsic-sim 的 資料夾中找到，並上傳到 rtl 的資料夾下。

同時我們要將 concat_EdgeDetect_Top.v 改名為 concat_EdgeDetect_Top_fsic.v 也複製到 rtl 的資料夾下。

緊接著我們要把EdgeDetect top module 放入user_prj0(這部分design是從github clone下來的，大致一致，接下來我主要會說明設計)。

```
EdgeDetect_Top U_EdgeDetect (
.clk                        (axi_clk           ), //user_clock2 ?
.rst                        (reg_rst           ),
.arst_n                     (axi_reset_n       ), //~uck2_rst_n ?
.widthIn                    (reg_widthIn       ), //I
.heightIn                   (reg_heightIn      ), //I
.sw_in                      (reg_sw_in         ), //I
.crc32_hw_pix_in_rsc_dat       (crc32_stream_in    ), //O
.crc32_hw_pix_in_triosy_lz     (),                    //O, not useful
.crc32_hw_pix_out_rsc_dat      (crc32_stream_out   ), //O
.crc32_hw_pix_out_triosy_lz    (edgedetect_done    ), //O
.din_chn_rsc_dat             (dat_in_rsc_dat    ), //I
.din_chn_rsc_vld             (ss_tvalid         ), //I
.din_chn_rsc_rdy             (dat_in_rsc_rdy    ), //O
.dout_chn_rsc_dat            (dat_out_rsc_dat   ), //O
.dout_chn_rsc_vld            (sm_tvalid         ), //O
.dout_chn_rsc_rdy            (sm_tready         ), //I
.line_buf0_rsc_en           (ram0_en           ), //O
.line_buf0_rsc_q            (ram0_q            ), //I
.line_buf0_rsc_we           (ram0_we           ), //O
.line_buf0_rsc_d            (ram0_d            ), //O
.line_buf0_rsc_adr          (ram0_adr          ), //O
.line_buf1_rsc_en           (ram1_en           ), //O
.line_buf1_rsc_q            (ram1_q            ), //I
.line_buf1_rsc_we           (ram1_we           ), //O
.line_buf1_rsc_d            (ram1_d            ), //O
.line_buf1_rsc_adr          (ram1_adr          )  //O
);
```

在接線上時，因為在Verder中有兩個line buffer所以我們需要接上兩個 SRAM (spram.v)

```
//SRAM
SPRAM #(.data_width(64),.addr_width(7),.depth(80)) U_SPRAM_0(
.adr (ram0_adr ),
.d   (ram0_d   ),
.en  (ram0_en  ),
.we  (ram0_we  ),
.clk (axi_clk  ), //user_clock2 ?
.q   (ram0_q   )
);

SPRAM #(.data_width(64),.addr_width(7),.depth(80)) U_SPRAM_1(
.adr (ram1_adr ),
.d   (ram1_d   ),
.en  (ram1_en  ),
.we  (ram1_we  ),
.clk (axi_clk  ), //user_clock2 ?
.q   (ram1_q   )
);
//~
```

接者我們要連接我們的axi-stream的接口:

```
.dout_chn_rsc_vld            (sm_tvalid         ), //O
.dout_chn_rsc_rdy            (sm_tready         ), //I
```

```
.din_chn_rsc_vld             (ss_tvalid         ), //I
```

除了 axi-stream的interface外，我們還需要利用設定register來輸出或輸入其他data像是widthin、hightin、sw_in ...等，我們到時會利用axi-lite來讀取或寫入(可以從soc端也可以從fpga端)。

- Control Write Register

```
//write register
always @(posedge axi_clk or negedge axi_reset_n)  begin
  if ( !axi_reset_n ) begin
    reg_widthIn         <= 640;
    reg_heightIn        <= 480;
    reg_sw_in           <= 1;
    reg_rst             <= 0;
  end else begin
    if ( awvalid_in && wvalid_in ) begin    //when awvalid_in=1 and wvalid_in=1 means awready_out=1 and wready_out=1
      if          (awaddr[11:2] == 10'h000 ) begin //offset 0
        if ( wstrb[0] == 1) reg_rst           <= wdata[0];
      end else if (awaddr[11:2] == 10'h001 ) begin //offset 1
        if ( wstrb[0] == 1) reg_widthIn[7:0]  <= wdata[7:0];
        if ( wstrb[1] == 1) reg_widthIn[9:8]  <= wdata[9:8];
      end else if (awaddr[11:2] == 10'h002 ) begin //offset 2
        if ( wstrb[0] == 1) reg_heightIn[7:0] <= wdata[7:0];
        if ( wstrb[1] == 1) reg_heightIn[8]   <= wdata[8];
      end else if (awaddr[11:2] == 10'h003 ) begin //offset 3
        if ( wstrb[0] == 1) reg_sw_in         <= wdata[0];
      end
    end
  end
end

always @(posedge axi_clk or negedge axi_reset_n)  begin
  if ( !axi_reset_n ) begin
      reg_edgedetect_done <= 0;
  end else begin
    if (edgedetect_done)
      reg_edgedetect_done <= 1;
    else if (awaddr[11:2] == 10'h006 ) begin //offset 6
      if ( wstrb[0] == 1) reg_edgedetect_done <= 0;
    end
  end
end
```

- Control Read Register

```
//read register
reg [(pDATA_WIDTH-1) : 0] rdata_tmp;
assign arready = 1; // ?
assign rvalid  = 1; // ?
assign rdata =  rdata_tmp;

always @* begin
  if      (araddr[11:2] == 10'h000) rdata_tmp = reg_rst;
  else if (araddr[11:2] == 10'h001) rdata_tmp = reg_widthIn;
  else if (araddr[11:2] == 10'h002) rdata_tmp = reg_heightIn;
  else if (araddr[11:2] == 10'h003) rdata_tmp = reg_sw_in;
  else if (araddr[11:2] == 10'h004) rdata_tmp = reg_crc32_stream_in;
  else if (araddr[11:2] == 10'h005) rdata_tmp = reg_crc32_stream_out;
  else if (araddr[11:2] == 10'h006) rdata_tmp = reg_edgedetect_done;
  else                              rdata_tmp = 0;
end
```

how we test our design in FSIC

testbench design

這部分design是從github clone下來的，大致一致，接下來我主要會說明他的設計。

首先如同lab1_sim所提到我們需要初始化，然而因為user project selction control defalut 就是user_prj0，所以這部分不用特別設定。(如果不在user_prj0，就需要設定，如同lab1)

再者，上述有提到我們需要利用 axi-lite寫入data，如下:

```
soc_up_cfg_write('h4, 4'b0111, cfg_read_data_expect_value);  //widthIn
soc_up_cfg_write('h8, 4'b0111, cfg_read_data_expect_value);  //heightIn
soc_up_cfg_write('hc, 4'b0001, cfg_read_data_expect_value);  //sw
```

最後我們利用axi-stream傳data，並且也可以讀出data驗證。(這部分design方法如同lab1做fir一樣的方式，可以參考此詳細說明:
https://github.com/nthuyouwei/asoclab/blob/main/lab01/fsic-sim/asoclab01_fsic-sim_report.pdf (https://github.com/nthuyouwei/asoclab/blob/main/lab01/fsic-sim/asoclab01_fsic-sim_report.pdf))

```
test002_fpga_axis_req();        //target to Axis Switch

$display($time, "=> wait for soc_to_fpga_axis_event");
@(soc_to_fpga_axis_event);
```

```
task test002_fpga_axis_req;
    //input [7:0] compare_data;

    //FPGA to SOC Axilite test
    begin

        @ (posedge fpga_coreclk);
        fpga_as_is_tready <= 1;

        for(idx3=0; idx3<fpga_axis_test_length; idx3=idx3+1)begin       //
            fpga_axis_req(32'h11111111 * (idx3 & 32'h0000_000F), TID_DN_UP, 1);     //target to User Project
        end

        $display($time, "=> test002_fpga_axis_req done");
    end
endtask
```

```
task fpga_axis_req;
    input [31:0] data;
    input [1:0] tid;
    input mode; //o ffor noram, 1 for random data
    reg [31:0] tdata;
    `ifdef USER_PROJECT_SIDEBAND_SUPPORT
        reg [pUSER_PROJECT_SIDEBAND_WIDTH-1:0]tupsb;
    `endif
    reg [3:0] tstrb;
    reg [3:0] tkeep;
    reg tlast;

    begin
        if (mode) begin     //for random data
            tdata = $random;
            `ifdef USER_PROJECT_SIDEBAND_SUPPORT
                tupsb = $random;
            `endif
            tstrb = $random;
            tkeep = $random;
            tlast = $random;
        end
        else begin
            tdata = data;
            `ifdef USER_PROJECT_SIDEBAND_SUPPORT
                //tupsb = 5'b00000;
                tupsb = tdata[4:0];
            `endif
            tstrb = 4'b0000;
            tkeep = 4'b0000;
            tlast = 1'b0;
        end
        `ifdef USER_PROJECT_SIDEBAND_SUPPORT
            fpga_as_is_tupsb <= tupsb;
        `endif
        fpga_as_is_tstrb <=  tstrb;
        fpga_as_is_tkeep <=  tkeep;
        fpga_as_is_tlast <=  tlast;
        fpga_as_is_tdata <= tdata;  //for axis write data
        `ifdef USER_PROJECT_SIDEBAND_SUPPORT
            $strobe($time, "=> fpga_axis_req send data, fpga_as_is_tupsb = %b, fpga_as_is_tstrb = %b, fpga_as_is_tkeep = %b,
        `else
            $strobe($time, "=> fpga_axis_req send data, fpga_as_is_tstrb = %b, fpga_as_is_tkeep = %b, fpga_as_is_tlast = %b,
        `endif

        fpga_as_is_tid <=  tid;     //set target
        fpga_as_is_tuser <=  TUSER_AXIS;        //for axis req
        fpga_as_is_tvalid <= 1;
        `ifdef USER_PROJECT_SIDEBAND_SUPPORT
            soc_to_fpga_axis_expect_value[soc_to_fpga_axis_expect_count] <= {tupsb, tstrb, tkeep, tlast, tdata};
        `else
            soc_to_fpga_axis_expect_value[soc_to_fpga_axis_expect_count] <= {tstrb, tkeep, tlast, tdata};
        `endif
        soc_to_fpga_axis_expect_count <= soc_to_fpga_axis_expect_count+1;

        @ (posedge fpga_coreclk);
        while (fpga_is_as_tready == 0) begin        // wait util fpga_is_as_tready == 1 then change data
                @ (posedge fpga_coreclk);
        end
        fpga_as_is_tvalid <= 0;

    end
endtask
```

Use questasim simulation -vsim

the result:

```
# ==========================================================================================
# ==========================================================================================
# ==========================================================================================
#             18555165=> Final result [PASS], check_cnt = 115301, error_cnt =     0
# ==========================================================================================
# ==========================================================================================
# ==========================================================================================
# ** Note: $finish    : ../tb_fsic.v(437)
#    Time: 18555165 ns  Iteration: 0  Instance: /tb_fsic
# qwavedb_dumpvars : Simulation ending at [4 1375295816] 0
# End time: 22:22:50 on Apr 04,2024, Elapsed time: 0:02:25
# Errors: 0, Warnings: 3
[u110011141@ws41 vsim]$
```

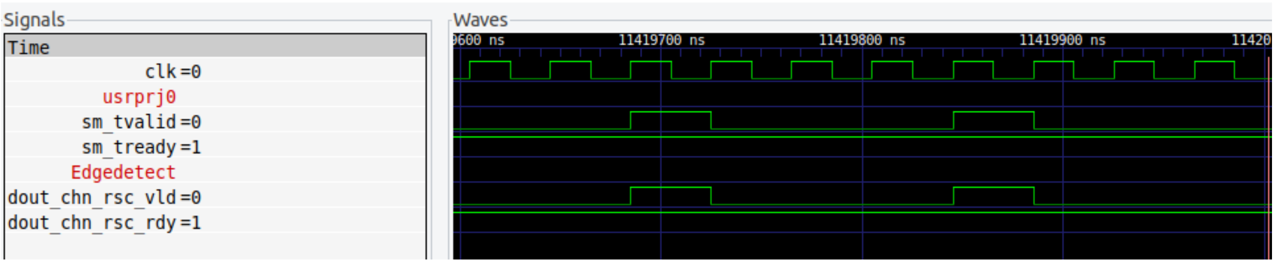
Use vivado simulation - xsim

我們可以跟lab1一樣利用vivado xsim來跑模擬，會得到一樣的結果。

```
===============================================================================
===============================================================================
===============================================================================
            18552405=> Final result [PASS], check_cnt = 115301, error_cnt = 0000
===============================================================================
===============================================================================
===============================================================================
$finish called at time : 18552405 ns : File "/home/ubuntu/Desktop/git_asoc/lab02/03_fsic_prj_xsim/rtl/user/testbench/tb_fsic.v" Line 437
run: Time (s): cpu = 00:22:27 ; elapsed = 00:33:15 . Memory (MB): peak = 2859.367 ; gain = 516.828 ; free physical = 121 ; free virtual = 6207
## quit
INFO: xsimkernel Simulation Memory Usage: 127144 KB (Peak: 170952 KB), Simulation CPU Usage: 1864410 ms
```

我們還可以利用gtkwave來分析waveform:

首先最重要的throughput:

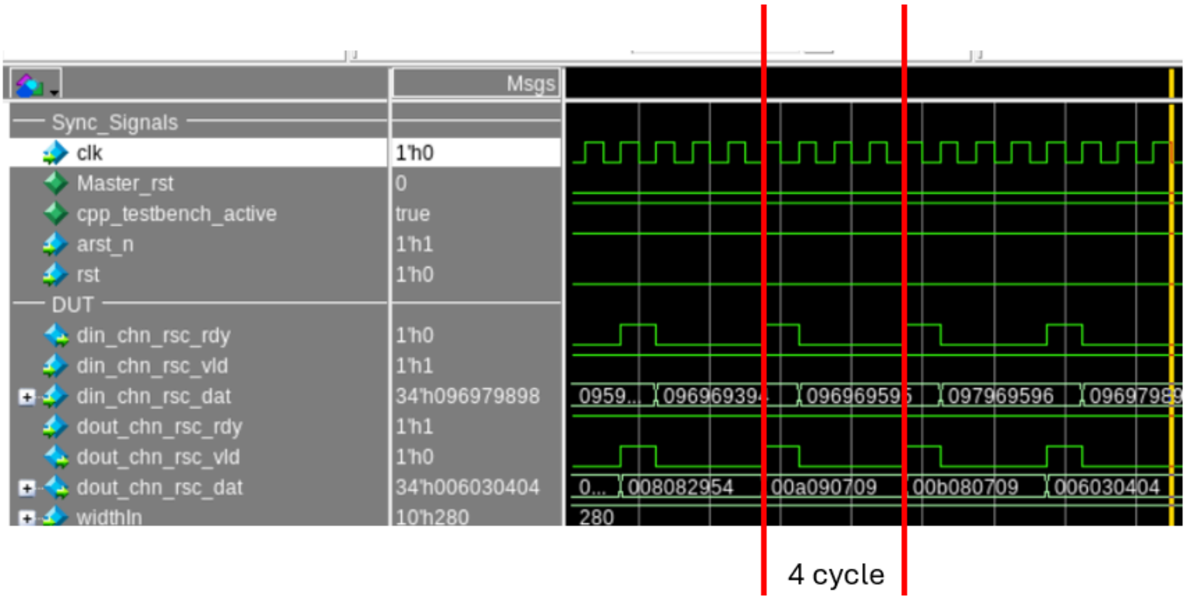


可以知道這邊thoughput 為4 ，這我們可以跟catapult report上的report來比對，我一開始以為應該是一樣的但我猜想這裡應該是跟Mag比對因為Mag是最後一層輸出。且後面也去看了cosim的waveform throughput也是4。

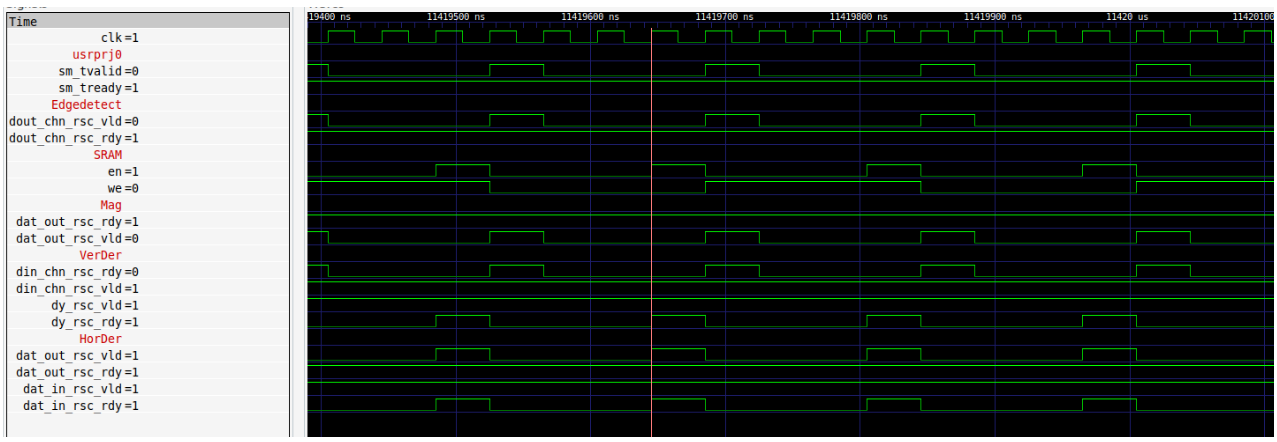
- catapult report:

olution	Latency...	Latency...	Throug...	Throug...
solution.v1 (new)				
EdgeDetect_Top.v1 (extract)	7	70.00	6	60.00
EdgeDetect_Top				
EdgeDetect_MagAng				
run	2	20.00	4	40.00
run:rlp	2	20.00	4	40.00
main	2	20.00	4	40.00
MROW			1	10.00
EdgeDetect_HorDer				
run	2	20.00	5	50.00
run:rlp	2	20.00	5	50.00
main	2	20.00	5	50.00
HROW			1	10.00
EdgeDetect_VerDer				
run	3	30.00	6	60.00
run:rlp	3	30.00	6	60.00
main	3	30.00	6	60.00
VROW			1	10.00

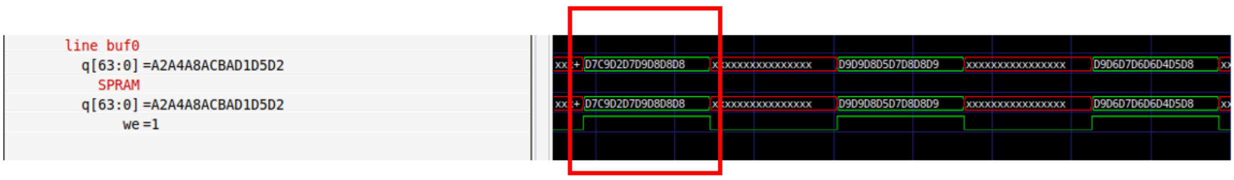
- cosim waveform:



最後，我們當然也可以把各其他module的waveform拿出來觀察研究:

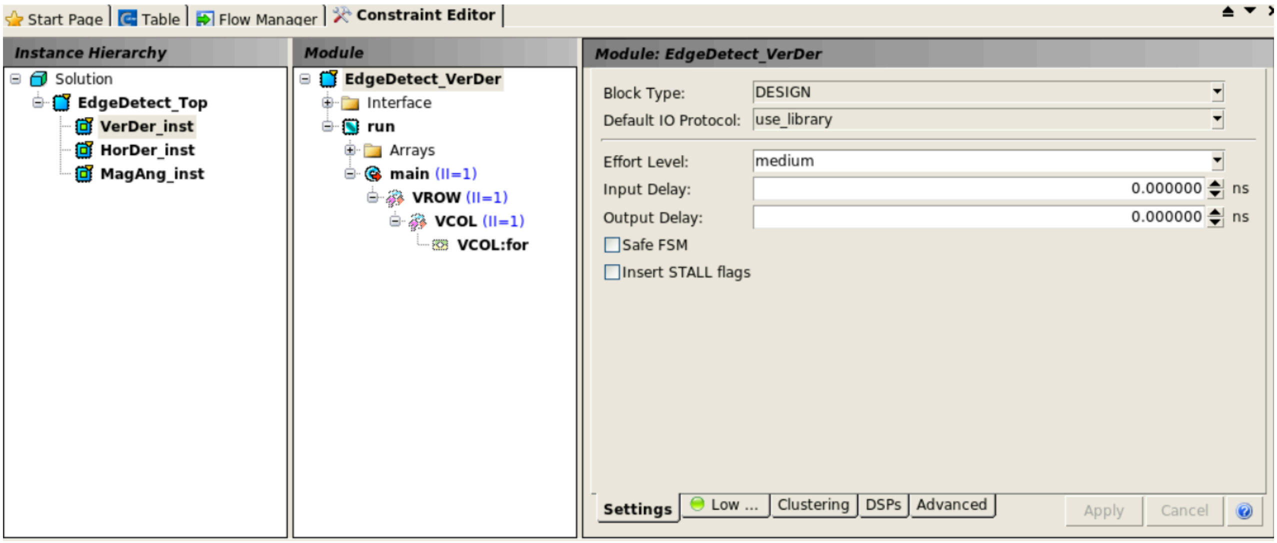


例如我們可以確定在ver module中的line buffer data有寫入SPRAM:



futher optimization

我們可以更改architecture讓每個main function都設定成II=1



最後可以發現throughput=1:

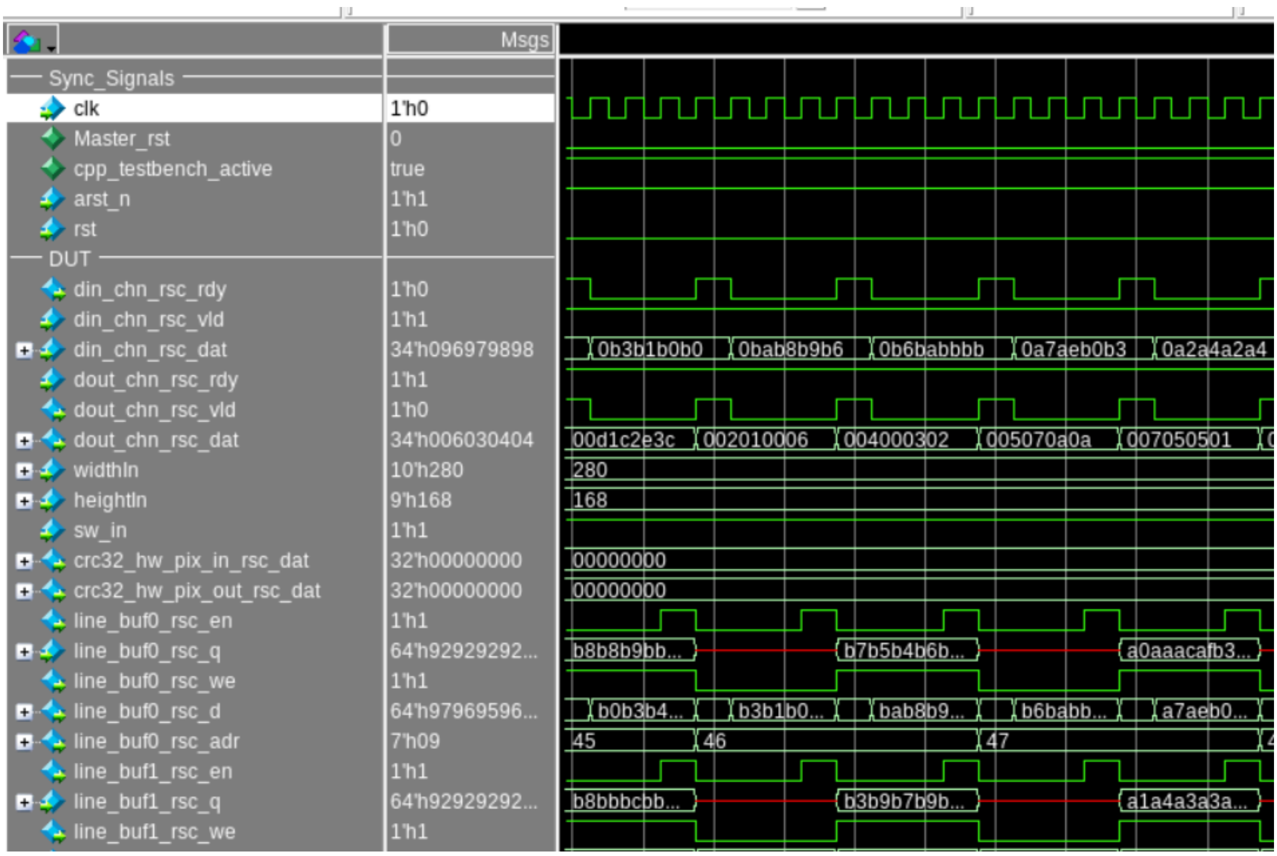
Start Page | Table | Flow Manager | Constraint Editor

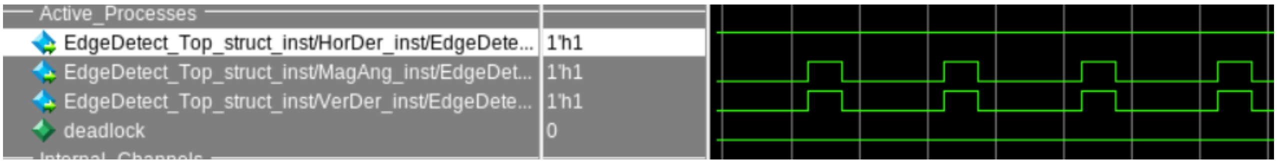
Report: General

Solution	Latency...	Latency...	Throug...	Throug...	Slack	Total Area
solution.v1 (new)						
EdgeDetect_Top.v1 (extract)	7	70.00	6	60.00	7.30	12198.08
EdgeDetect_Top.v2 (extract)	6	60.00	1	10.00	6.85	11453.91

不過在驗證上，不論是cosim或者fsic可能需要重新設計tb，因為以目前cosim 結果可以發現throughput還是等於4且block Mag and Ver有idle的狀態，除此之外，目前fsic 中的tb會卡住，未來如果時間允許的話也可以再去修改。

- waveform of questasim





無法連至 reCAPTCHA 服務。請檢查你的網際網路連線，並重新載入頁面以取得 reCAPTCHA 驗證問題。