

# Step-By-Step Lab2: EdgeDetect

Yuan-Teng Chang, Catapult HLS AE  
[yuan-teng.chang@siemens.com](mailto:yuan-teng.chang@siemens.com)

# Objectives

- A case study of designing HLS C++ from algorithm C++
- To be familiar with the Catapult HLS flow
- Learn how to do RTL integration and verification with FSIC project

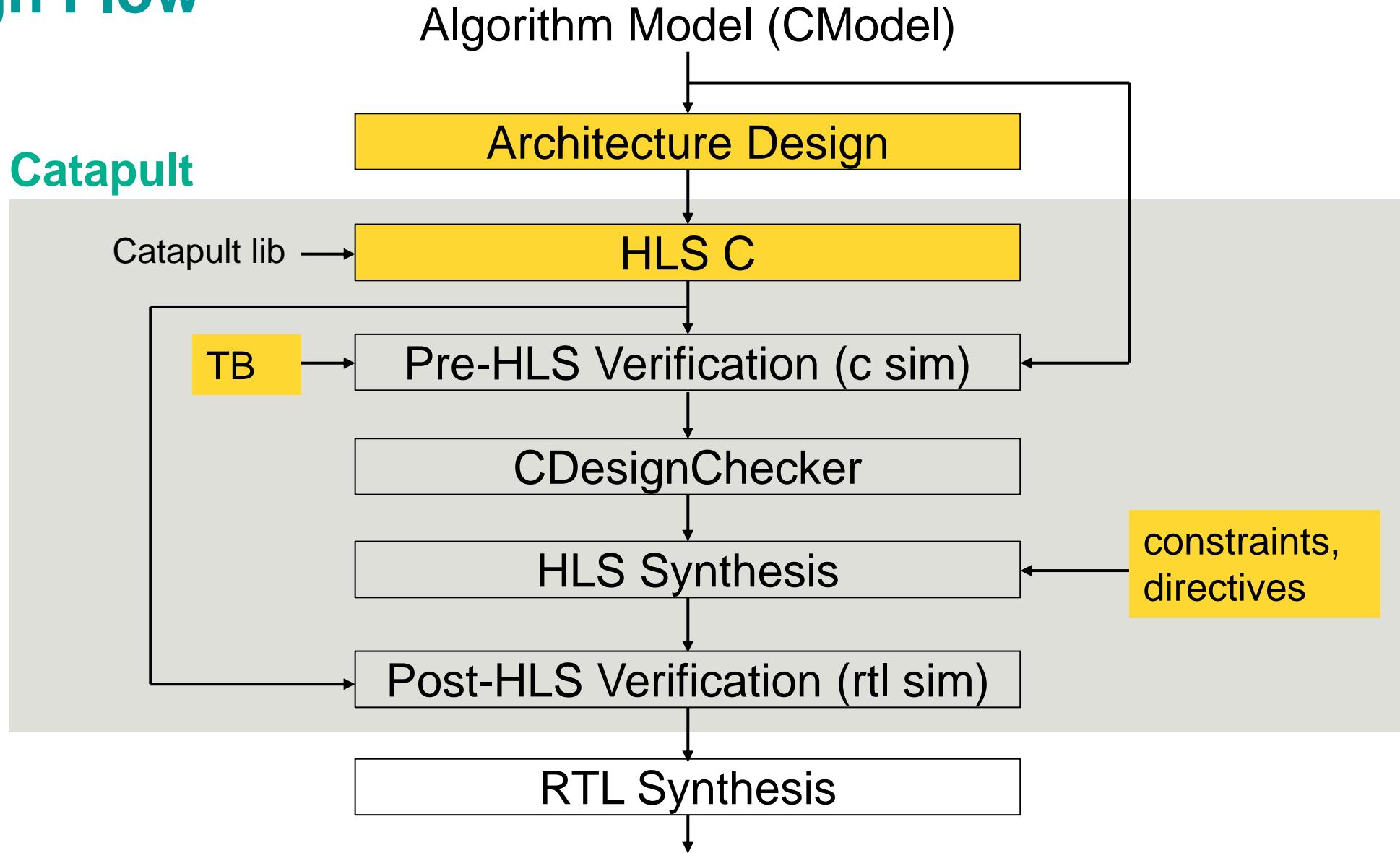
# Environment Variables

To run Catapult HLS and QuestaSim, a few environment variables are required to be defined

- Define them in a shell script like setup-catapult.csh and source it
- Modify the path depending on the tool install path in your workstation

```
2 echo "Catapult setup"
3 setenv MGC_HOME /usr/local/bin/Siemens_EDA/Catapult_Synthesis_2022.2_1-1019737/Mgc_home
4
5 setenv CXX_HOME $MGC_HOME
6 setenv SYSTEMC_INCDIR $MGC_HOME/shared/include
7 setenv SYSTEMC_LIBDIR $MGC_HOME/shared/lib
8 set path = ($path ${MGC_HOME}/bin)
9
10 setenv LD_LIBRARY_PATH $SYSTEMC_LIBDIR
11 setenv LD_LIBRARY_PATH ${MGC_HOME}/lib:$LD_LIBRARY_PATH
12
13 echo "QuestaSim setup"
14 setenv MODEL_TECH /usr/local/bin/Siemens_EDA/Questa_Sim_2022_4/questasim/bin
15 setenv QUESTA_HOME /usr/local/bin/Siemens_EDA/Questa_Sim_2022_4/questasim
16 set path = ($path ${MODEL_TECH})
```

# Design Flow



# | Design HLS C++ from algorithm C++

01\_edgedetect

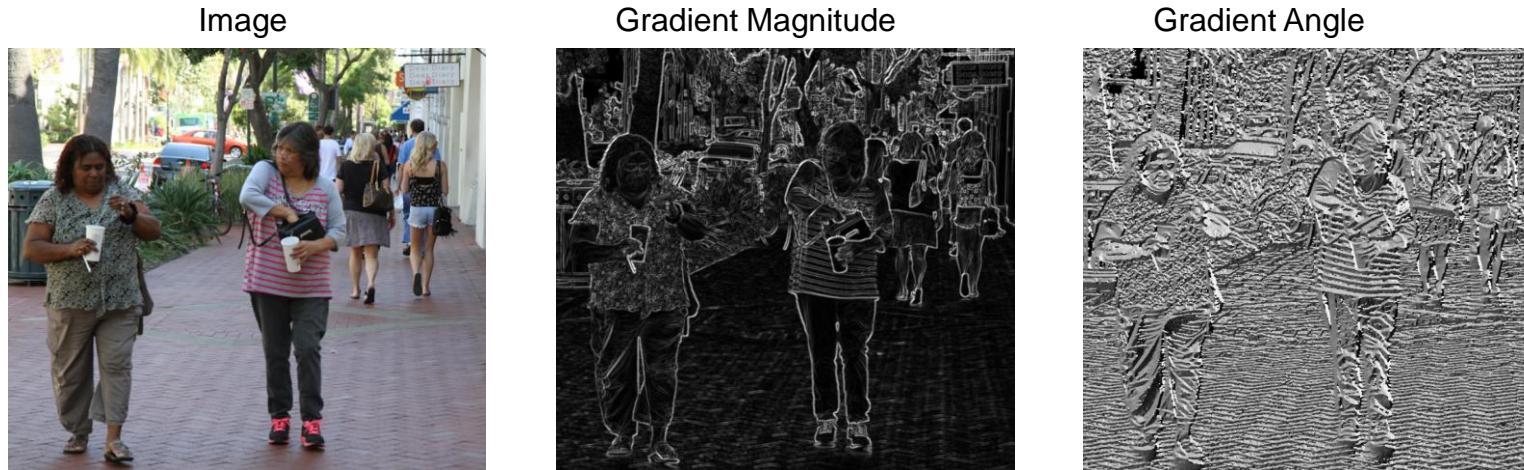
# An Edge Detection Algorithm

Used in many image and video processing algorithms

- People detection, texture matching, etc.

Computing the image gradient detects the edges in an image

- Magnitude gives the strength of the edge
- Angle gives the direction of the edge



# Gradient Magnitude

Compute derivatives of the image in the horizontal(x) and vertical(y) directions

- Computationally simple
- Requires memory architecture for efficient hardware

Take the square root of the sum of squares of the derivatives

- Computationally complex

**Gradient magnitude**

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

x derivative      y derivative      Square      Square root

The diagram illustrates the formula for Gradient magnitude. It shows the formula  $\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$ . Four blue callout boxes point to specific parts of the formula: one points to the term  $\frac{\partial f}{\partial x}$  with the label "x derivative"; another points to the term  $\frac{\partial f}{\partial y}$  with the label "y derivative"; a third points to the square operation  $(\cdot)^2$  with the label "Square"; and a fourth points to the square root operation  $\sqrt{\cdot}$  with the label "Square root".

# Computing 1-D derivative

1x3 and 3x1 filter kernel used to compute the x and y derivatives

10	20	10	200	210	250	250
----	----	----	-----	-----	-----	-----

$$f'(x) = \frac{f(x+1) - f(x-1)}{2} = \frac{210 - 10}{2} = 100$$

-1	0	1
----	---	---

ID derivative filter

1x3 kernel

Horizontal(x) derivative

Kernel slides “horizontally” across the image

0	0	0	0	0	0	0	0	0	0	0	0
-1	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0

Kernel slides “vertically” across the image

Vertical(y) derivative

0	-1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0

Image boundary  
is zero padded,  
replicated, etc

# Computing the Angle

Angle is computed by taking the arctangent of the ratio of the vertical(y) to horizontal(x) derivatives

- Computationally expensive

## Gradient direction

$$\theta = \tan^{-1} \left( \frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right)$$

# The C++ Algorithm Top-level

Implemented in C++ class

Three member functions to compute the horizontal and vertical derivatives plus magnitude/angle

Pointers used extensively for array accesses

Floating point data

Dynamic memory allocation to store derivatives

```
class EdgeDetect_Algorithm
{
...
public:
    EdgeDetect_Algorithm() {}

    void run(unsigned char *dat_in, // image data (streamed in by pixel)
             double      *magn,   // magnitude output
             double      *angle)  // angle output
    {
        // allocate buffers for image data
        double *dy = (double *)malloc(maxImageHeight*maxImageWidth*sizeof(double));
        double *dx = (double *)malloc(maxImageHeight*maxImageWidth*sizeof(double));

        verticalDerivative(dat_in, dy);
        horizontalDerivative(dat_in, dx);
        magnitudeAngle(dx, dy, magn, angle);

        free(dy);
        free(dx);
    }
    ...
};
```

# C++ Algorithm Vertical Derivative

Vertical derivative reads across three different lines of image data ( $y-1$ ,  $y$ ,  $y+1$ )

Clip function prevents the index from reading out of bounds

- Boundary pixels are replicated

Horizontal derivative is similar

0	-1	0	0	0	0	0	0	0
0	0							0
0	1							0
0								0
0								0
0								0
0								0
0								0
0								0

```
int clip(int i, int bound) {
    if (i < 0) {
        return 0;
    } else if (i >= bound) {
        return bound;
    } else {
        return i;
    }
}

void verticalDerivative(unsigned char *dat_in,
                      double *dy)
{
    for (int y = 0; y < imageHeight; y++) {
        for (int x = 0; x < imageWidth; x++) {
            // Calculate derivative
            *(dy + y * imageWidth + x) =
                dat_in[clip(y - 1, imageHeight-1) * imageWidth + x] * kernel[0] +
                dat_in[y * imageWidth + x] * kernel[1] +
                dat_in[clip(y + 1, imageHeight-1) * imageWidth + x] * kernel[2];
        }
    }
}
```

```
int clip(int i, int bound) {
    if (i < 0) {
        return 0;
    } else if (i >= bound) {
        return bound;
    } else {
        return i;
    }
}
```

# C++ Algorithm Magnitude Angle

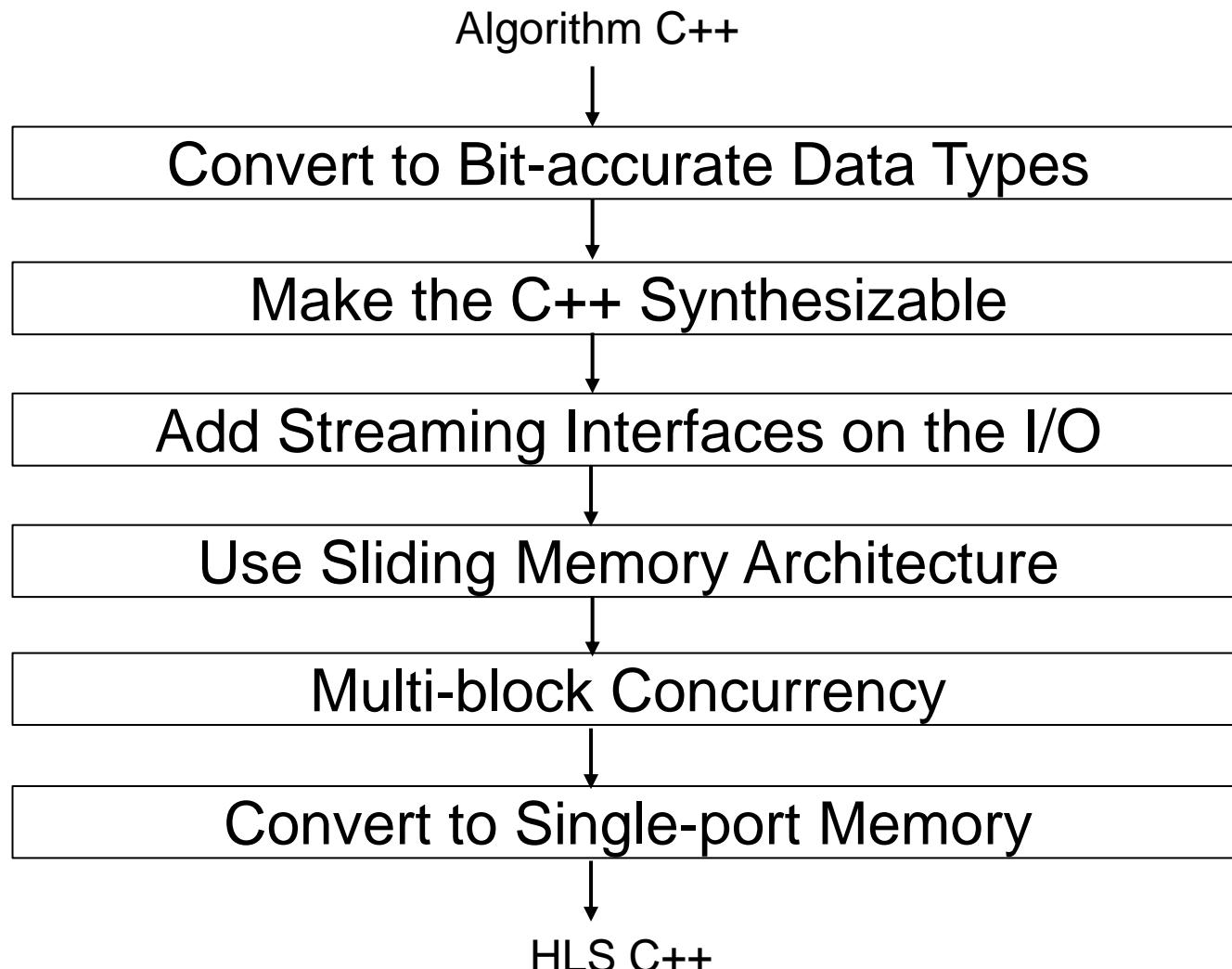
Squares the derivatives

Performs square root (math.h)

Performs atan2 (math.h)

```
void magnitudeAngle(double *dx,
                     double *dy,
                     double *magn,
                     double *angle)
{
    double dx_sq;
    double dy_sq;
    double sum;
    for (int y = 0; y < imageHeight; y++) {
        for (int x = 0; x < imageWidth; x++) {
            dx_sq = *(dx + y * imageWidth + x) * *(dx + y * imageWidth + x);
            dy_sq = *(dy + y * imageWidth + x) * *(dy + y * imageWidth + x);
            sum = dx_sq + dy_sq;
            *(magn + y * imageWidth + x) = sqrt(sum);
            *(angle + y * imageWidth + x) = atan2(dy[y * imageWidth + x], dx[y * imageWidth + x]);
        }
    }
}
```

# Converting Algorithm C++ to HLS C++



# Converting to Bit-Accurate Datatypes

## Includes

- #include<ac\_int.h>
- #include<ac\_fixed.h>

See ac data types guide for a full list of conversion methods

```
typedef uint8          pixelType;
typedef int9           gradType;
typedef uint18          sqType;
typedef ac_fixed<19,19,false> sumType;
typedef uint9           magType;
typedef ac_fixed<8,3,true> angType;
```

```
void run(pixelType *dat_in, // 8-bit unsigned for pixel data
         magType   *magn,    // 9-bit unsigned for magnitude output
         angType   *angle)   // 3-integer/5-fractional bits for quantized output
{
    // allocate buffers for image data
    gradType *dy = (gradType *)malloc(imageHeight*imageWidth*sizeof(gradType));
    gradType *dx = (gradType *)malloc(imageHeight*imageWidth*sizeof(gradType));

    verticalDerivative(dat_in, dy);
    horizontalDerivative(dat_in, dx);
    magnitudeAngle(dx, dy, magn, angle);

    free(dy);
    free(dx);
}
```

# Making C++ Synthesizable

Some input language constructs not supported

Dynamic memory allocation

- Physical hardware resources have finite amount of storage

Pointers to arrays mapped to memories on interfaces

- Arrays with constant bounds are needed

Use Catapult ac\_math.h to replace math.h

```
void run(pixelType *dat_in,  
        magType   *magn,  
        angType   *angle)
```

```
void CCS_BLOCK(run)(pixelType dat_in[imageHeight][imageWidth],  
                    magType   magn[imageHeight][imageWidth],  
                    angType   angle[imageHeight][imageWidth])
```

```
gradType *dy = (gradType *)malloc(imageHeight*imageWidth*sizeof(gradType));  
gradType *dx = (gradType *)malloc(imageHeight*imageWidth*sizeof(gradType));
```

```
gradType dx[imageHeight][imageWidth];  
gradType dy[imageHeight][imageWidth];
```

```
*(magn + y * imageWidth + x) = sqrt(sum.to_double()); // Convert ac_fixed to double to call math.h sqrt()  
*(angle + y * imageWidth + x) = atan2(dy[y * imageWidth + x].to_int(), dx[y * imageWidth + x].to_int());
```

```
ac_math::ac_sqrt_pwl(sum,sq_rt);  
magn[y][x] = sq_rt.to_uint();  
ac_math::ac_atan2_cordic((ac_fixed<9,9>) (dy[y][x]), (ac_fixed<9,9>) (dx[y][x]), at);  
angle[y][x] = at;
```

# Overall Design Goals

~1 pixel/clock throughput

- Image is  $1296 \times 864 = \sim 1119744$  clocks
- Sliding window and line buffer architectures
- Multi-block concurrency

Streaming interfaces on the IO

Minimal on-chip memory

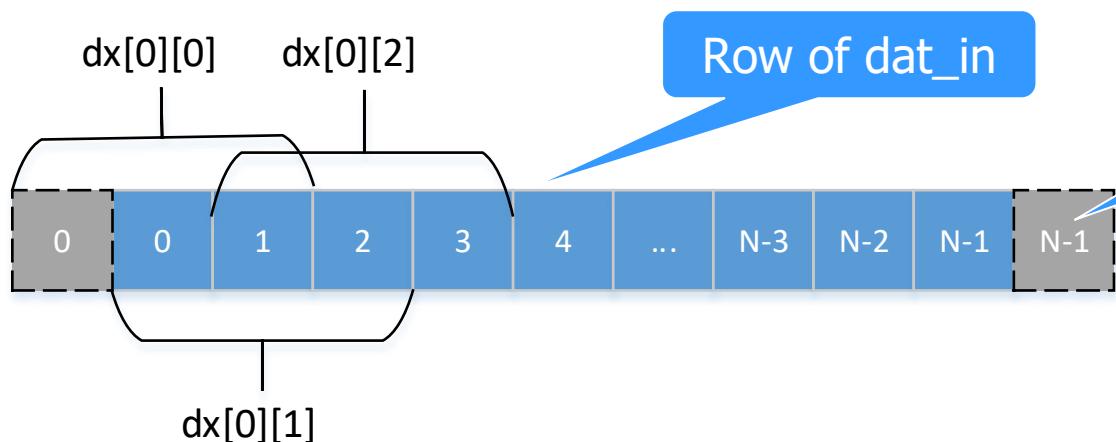
Architected for Single port memory

# Understanding the Required Memory Architecture

Pure algorithmic descriptions will have performance bottlenecks

- Due to lack of explicit memory architecture

Need to analyze array access patterns to see how to optimize the memory architecture



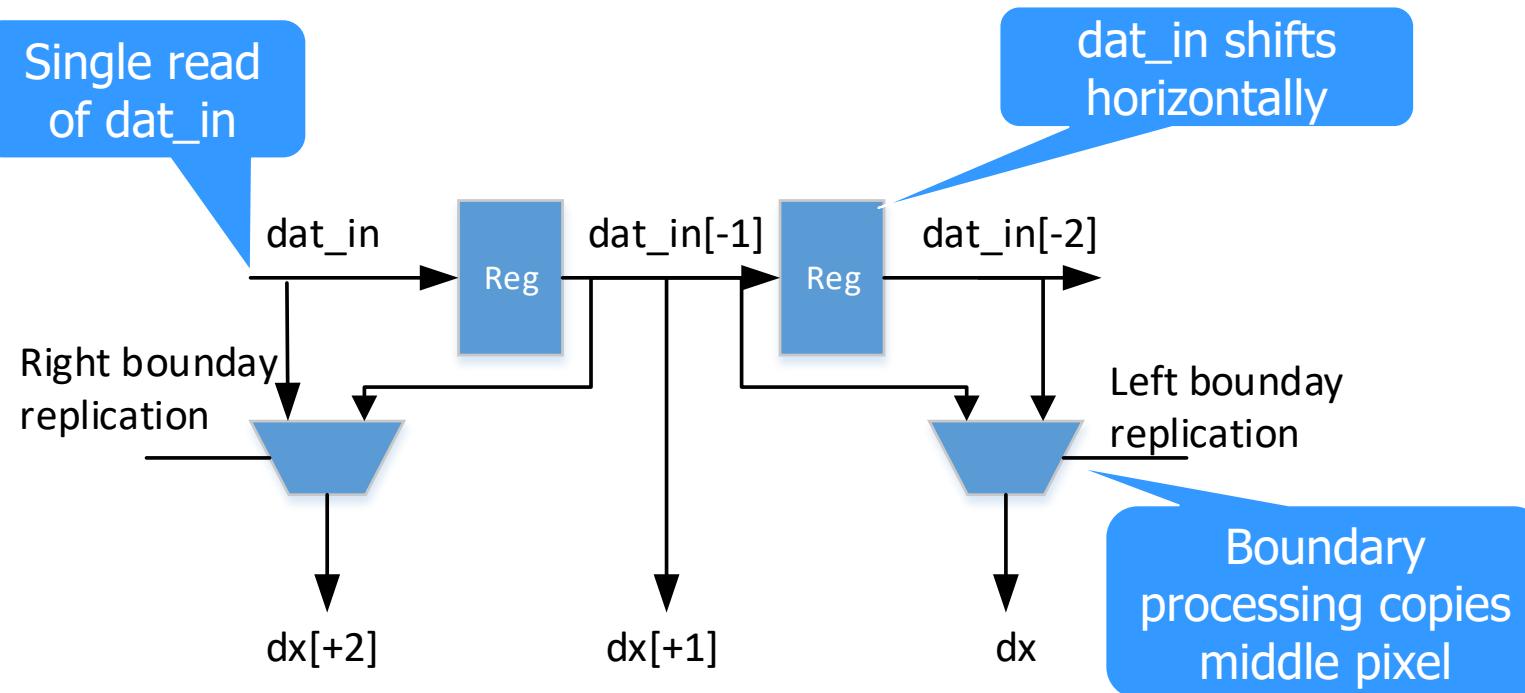
```
void horizontalDerivative(pixelType dat_in[imageHeight][imageWidth],  
                           gradType dx[imageHeight][imageWidth])  
{  
    HROW: for (int y = 0; y < imageHeight; y++) {  
        HCOL: for (int x = 0; x < imageWidth; x++) {  
            // Calculate derivative  
            dx[y][x] =  
                dat_in[y][clip(x - 1, imageWidth-1)] * kernel[0] +  
                dat_in[y][x] * kernel[1] +  
                dat_in[y][clip(x + 1, imageWidth-1)] * kernel[2];  
        }  
    }  
}
```

$dx[0][0] = dat\_in[0][0] * kernel[0] + [dat\_in[0][0] * kernel[1] + dat\_in[0][1] * kernel[2]]$   
 $dx[0][1] = [dat\_in[0][0] * kernel[0] + dat\_in[0][1] * kernel[1] + data\_in[0][2] * kernel[2]]$   
 $dx[0][2] = [dat\_in[0][1] * kernel[0] + data\_in[0][2] * kernel[1] + data\_in[0][3] * kernel[2]]$

# Shift-register Sliding Window Memory Architecture

Efficient memory architectures reduce the reading of a memory or an IO to once per clock cycle

- Allows pipelining with  $l=1$

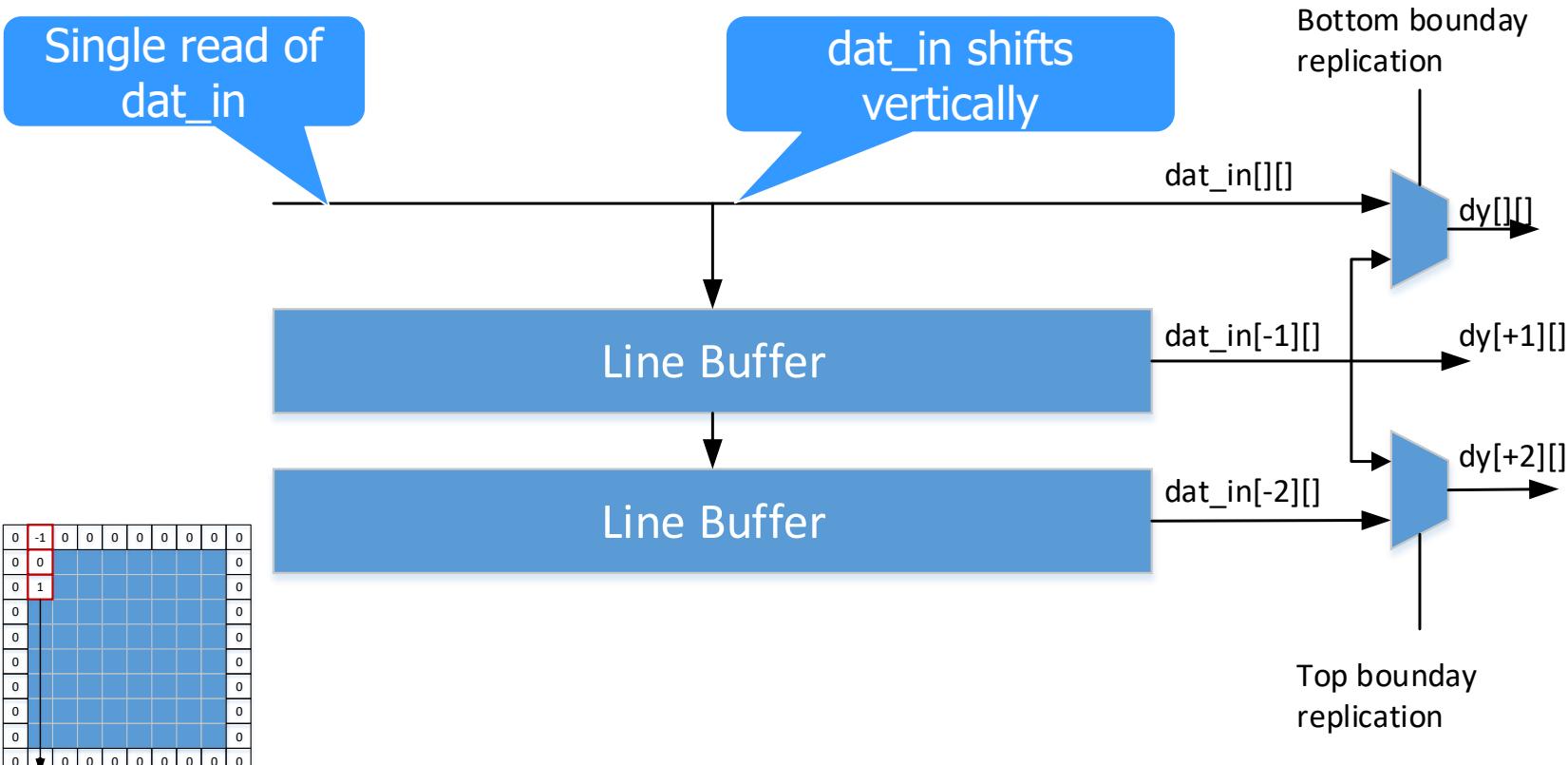


```
void horizontalDerivative(pixelType dat_in[imageHeight][imageWidth],  
                           gradType dx[imageHeight][imageWidth])  
{  
    // pixel buffers store pixel history  
    pixelType pix_buf0;  
    pixelType pix_buf1;  
  
    pixelType pix0 = 0;  
    pixelType pix1 = 0;  
    pixelType pix2 = 0;  
  
    HROW: for (int y = 0; y < imageHeight; y++) {  
        HCOL: for (int x = 0; x < imageWidth+1; x++) { // HCOL has one ext  
            pix2 = pix_buf1;  
            pix1 = pix_buf0;  
            if (x <= imageWidth-1) {  
                pix0 = dat_in[y][x]; // Read memory while in bounds  
            }  
            if (x == 1) {  
                pix2 = pix1; // left boundary (replicate pix1 left to pix2)  
            }  
            if (x == imageWidth) {  
                pix0 = pix1; // right boundary (replicate pix1 right to pix0)  
            }  
  
            pix_buf1 = pix_buf0;  
            pix_buf0 = pix0;  
  
            // Calculate derivative  
            if (x > 0) {  
                // wait till window ramp-up and adjust index j  
                dx[y][x-1] = pix2*kernel[0] + pix1*kernel[1] + pix0*kernel[2];  
            }  
        }  
    }  
}
```

# Line-buffer Sliding Window Memory Architecture

Vertical derivative moves across rows of the image

Data must shift through line buffer memories instead of registers to reuse previously read values



```
void verticalDerivative(pixelType dat_in[imageHeight][imageWidth],  
                        gradType dy[imageHeight][imageWidth])  
{  
    // Line buffers store pixel line history - Mapped to RAM  
    pixelType line_buf0[imageWidth];  
    pixelType line_buf1[imageWidth];  
    pixelType pix0, pix1, pix2;  
  
    VROW: for (int y = 0; y < imageHeight+1; y++) { // VROW has one extra  
    VCOL: for (int x = 0; x < imageWidth; x++) {  
        // vertical window of pixels  
        pix2 = line_buf1[x];  
        pix1 = line_buf0[x];  
        if (y <= imageHeight-1) {  
            pix0 = dat_in[y][x]; // Read memory while in bounds  
        }  
        line_buf1[x] = pix1; // copy previous line  
        line_buf0[x] = pix0; // store current line  
        // Boundary condition processing  
        if (y == 1) {  
            pix2 = pix1; // top boundary (replicate pix1 up to pix2)  
        }  
        if (y == imageHeight) {  
            pix0 = pix1; // bottom boundary (replicate pix1 down to pix0)  
        }  
  
        // Calculate derivative  
        if (y > 0) {  
            // wait till window ramp-up and adjust index y  
            dy[y-1][x] = pix2*kernel[0] + pix1*kernel[1] + pix0*kernel[2];  
        }  
    }  
}
```

# Understanding Memory Dependency Failures

Catapult will schedule memory reads and writes to the same memory in different cycles if simultaneous read/write to the same address is not allowed

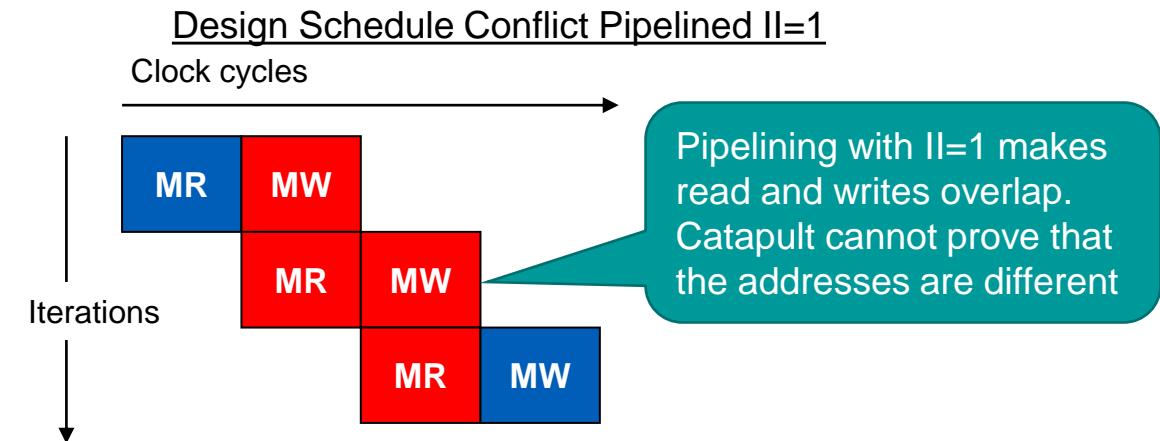
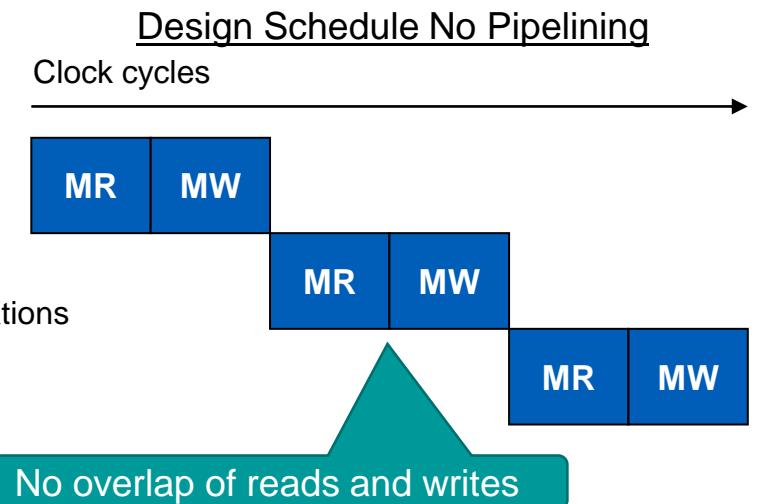
- Pipelining the design will make the reads and writes appear to overlap

```
VROW: for (int y = 0; y < imageHeight+1; y++) {  
    VCOL: for (int x = 0; x < imageWidth; x++) {  
        // vertical window of pixels  
        pix2 = line_buf1[x];  
        pix1 = line_buf0[x];  
        if (y <= imageHeight-1) {  
            pix0 = dat_in[y][x]; // Read memory while writing  
        }  
        line_buf1[x] = pix1; // copy previous line  
        line_buf0[x] = pix0; // store current line
```

Write mem

1

Read mem



# Ignoring False Memory Dependencies

The scheduler can be told to ignore the memory dependency if user knows there can't be address conflict

- Use ignore\_memory\_precedences constraint
- Avoid using wildcards "\*" for operations

May lead to many, and possibly erroneous, constraints

- Be careful, ignoring a real dependency will create “bad logic”

```
ignore_memory_precedences -from <operation> -to <operation>
```

```
✗ Feedback path is too long to schedule design with current pipeline and clock constraints.  
✗ Schedule failed, sequential delay violated. List of sequential operations: [list]  
✗ MEMORYREAD "read_mem(mem:rsc.@)" test.cpp(5,10,10)  
✗ MEMORYWRITE "if:write_mem(mem:rsc.@)" test.cpp(8,4,10)   
✗ Feedback path is too long to schedule design with current pipeline and clock constraints.  
# test.cpp(5,10,10): chained data dependency at time 11cy+8
```



```
go architect  
ignore_memory_precedences -from if:write_mem(mem:rsc.@) -to read_mem(mem:rsc.@)
```

# Sequential Loops Execute Sequentially in Hardware

Sequential loops in the same design block/partition will execute sequentially in hardware

```
#pragma hls_design interface
void CCS_BLOCK(run)(pixelType dat_in[imageHeight][imageWidth],
                     magType   magn[imageHeight][imageWidth],
                     angType   angle[imageHeight][imageWidth])
{
    // allocate buffers for image data
    gradType dx[imageHeight][imageWidth];
    gradType dy[imageHeight][imageWidth];

    verticalDerivative(dat_in, dy);
    horizontalDerivative(dat_in, dx);
    magnitudeAngle(dx, dy, magn, angle);
}
```

Cycles

verticalDerivative Loops  
~1 million cycles

horizontalDerivative Loops  
~1 million cycles

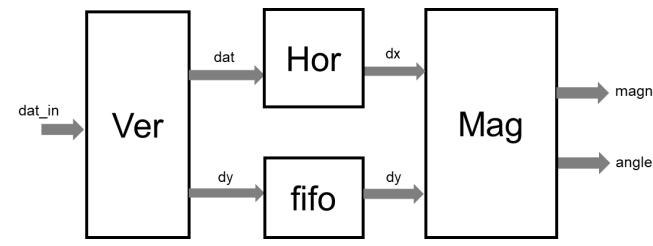
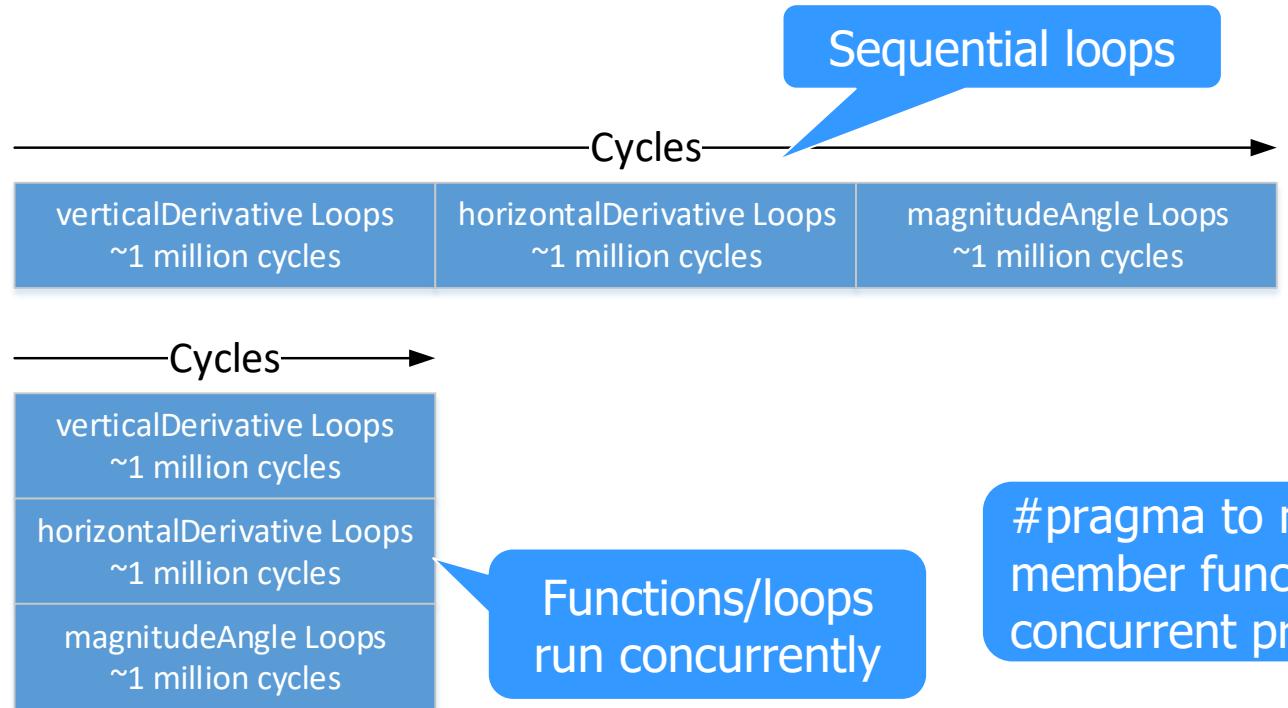
magnitudeAngle Loops  
~1 million cycles

# Adding Hierarchy in Catapult

Catapult can synthesize classes/functions as concurrent processes using design constraints

- ac\_channel interconnect required
- Allows sequential loops to run concurrently

ac\_channel  
interconnect



```
// Static interconnect channels (FIFOs) between blocks
ac_channel<gradType>      dy;
ac_channel<gradType>      dx;
ac_channel<pixelType>      dat; // channel for passing image data

public:
    EdgeDetect_Hierarchy() {}

//-----
// Function: run
//   Top interface for data in/out of class. Combines vertical derivative,
//   horizontal derivative and magnitude/angle computation
#pragma hls_design interface
void CCS_BLOCK(run)(ac_channel<pixelType> &dat_in,
                    ac_channel<magType> &magn,
                    ac_channel<angType> &angle)

{
    verticalDerivative(dat_in, dat, dy);
    horizontalDerivative(dat, dx);
    magnitudeAngle(dx, dy, magn, angle);
}

private:
//-----
// Function: verticalDerivative
//   Compute the vertical derivative on the input data
#pragma hls_design
void verticalDerivative(ac_channel<pixelType> &dat_in,
                       ac_channel<pixelType> &dat_out,
                       ac_channel<gradType> &dy)
```

# Dual Port Memory is Expensive for ASIC

FPGA memory hard macros are dual port

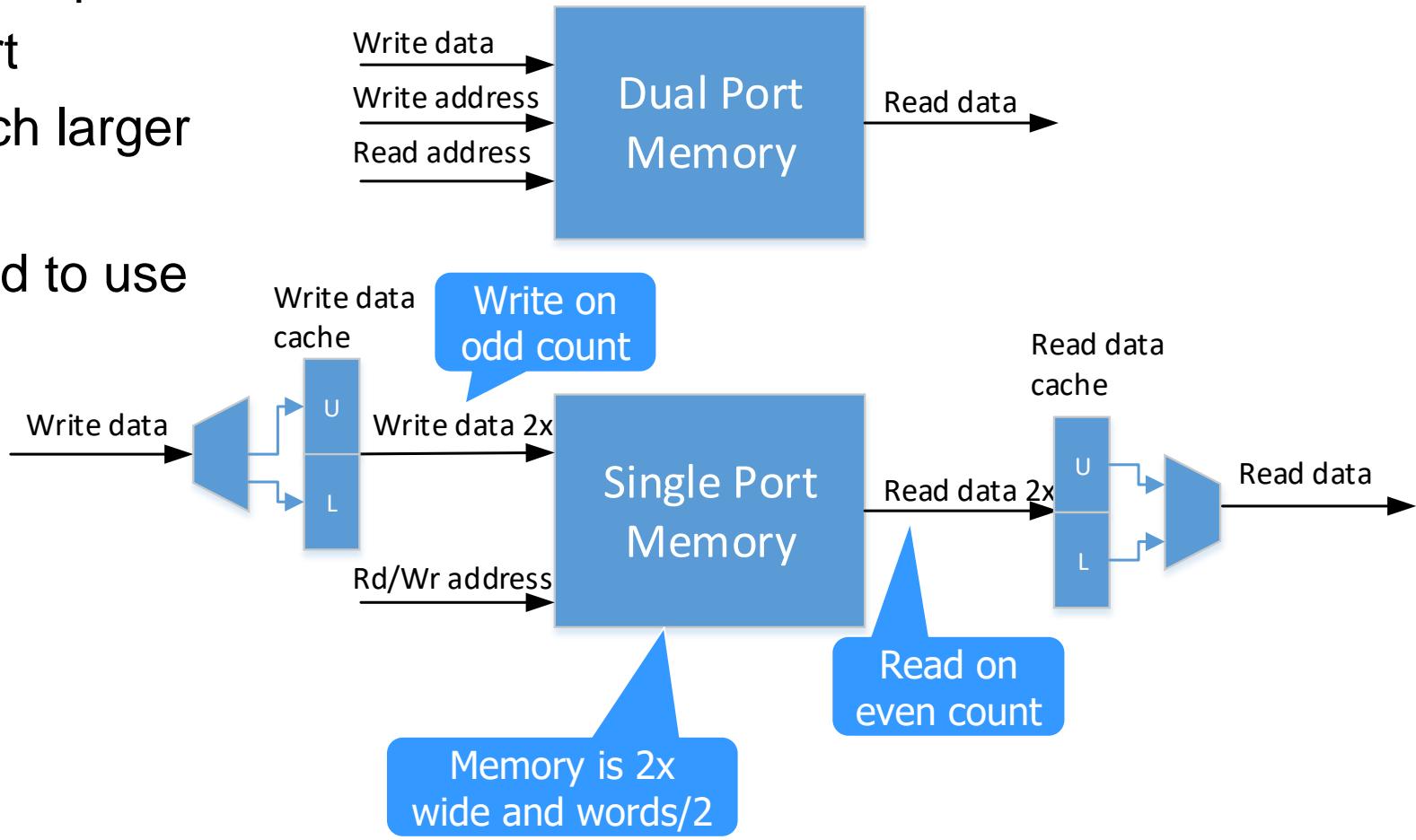
- Cost is the same as a single port

Dual port ASIC memory has a much larger area than single port memory

C++ code can often be restructured to use single port memory

```
pixelType2x line_buf0[imageWidth/2];  
pixelType2x line_buf1[imageWidth/2];
```

```
// Read line buffers into read  
if ( (x&1) == 0 ) {  
    // vertical window of pixels  
    rdbuf1_pix = line_buf1[x/2];  
    rdbuf0_pix = line_buf0[x/2];  
} else { // Write line buffer  
    line_buf1[x/2] = rdbuf0_pix;  
    line_buf0[x/2] = wrbuf0_pix;  
}
```



# | Catapult Flow – HLS & HLV

01\_edgedetect/cmodel – Algorithm C++

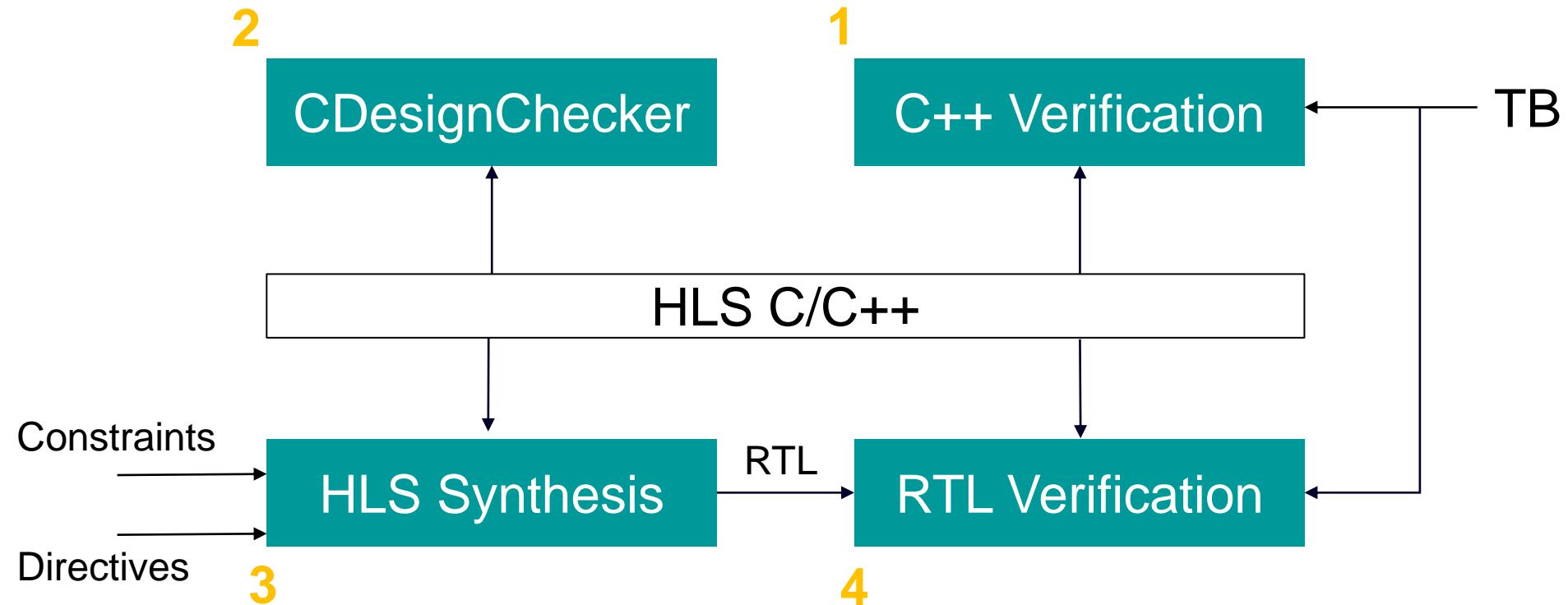
01\_edgedetect/hls\_c – HLS C++ design and TB

01\_edgedetect/bin – c sim directory

01\_edgedetect/catapult\_work – Catapult working directory

01\_edgedetect/Makefile – c sim Makefile

# Catapult Flow – HLS & HLV



# HLS C++ (EdgeDetect\_defs.h)

```
#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>

#ifndef USE_CIRCULARBUF

namespace EdgeDetect_IP
{
    const int maxImageWidth = 1926;
    const int maxImageHeight = 864;

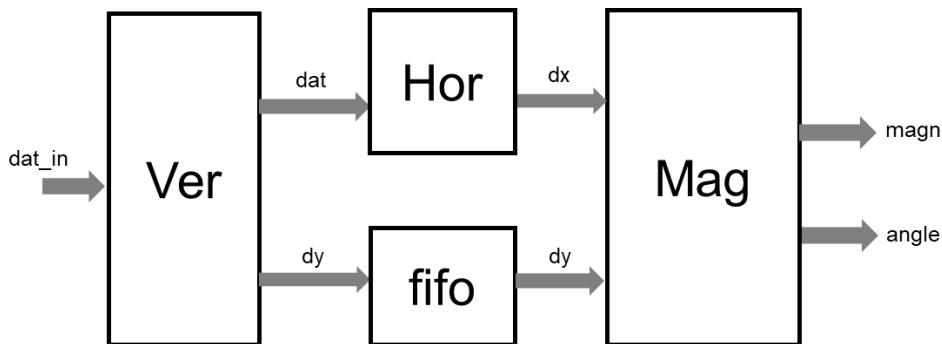
    const int kernel[3] = {1, 0, -1};

    // 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 int9           gradType;       // Derivative is max range -255 to 255
    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 uint9           magType;       // 9-bit unsigned magnitude result
    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;
}
```

# HLS C++ (EdgeDetect\_VerDer.h EdgeDetect\_HorDer.h EdgeDetect\_MagAng.h )

```
namespace EdgeDetect_IP
{
    class EdgeDetect_MagAng
    {
    public:
        EdgeDetect_MagAng() {}
        #pragma hls_design interface
        void CCS_BLOCK(run)(ac_channel<gradType> &dx_in,
                            ac_channel<gradType> &dy_in,
                            maxWType           &widthIn,
                            maxHType            &heightIn,
                            ac_channel<magType> &magn,
                            ac_channel<angType> &angle)
        {...}
    };
}
```



```
namespace EdgeDetect_IP
{
    class EdgeDetect_VerDer
    {
    public:
        EdgeDetect_VerDer() {}
        #pragma hls_design interface
        void CCS_BLOCK(run)(ac_channel<pixelType> &dat_in,
                            maxWType           &widthIn,
                            maxHType            &heightIn,
                            ac_channel<pixelType> &dat_out,
                            ac_channel<gradType> &dy)
        {...}
    };
}
```

```
namespace EdgeDetect_IP
{
    class EdgeDetect_HorDer
    {
    public:
        EdgeDetect_HorDer() {}
        #pragma hls_design interface
        void CCS_BLOCK(run)(ac_channel<pixelType> &dat_in,
                            maxWType           &widthIn,
                            maxHType            &heightIn,
                            ac_channel<gradType> &dx)
        {...}
    };
}
```

# HLS C++ (EdgeDetect.h)

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

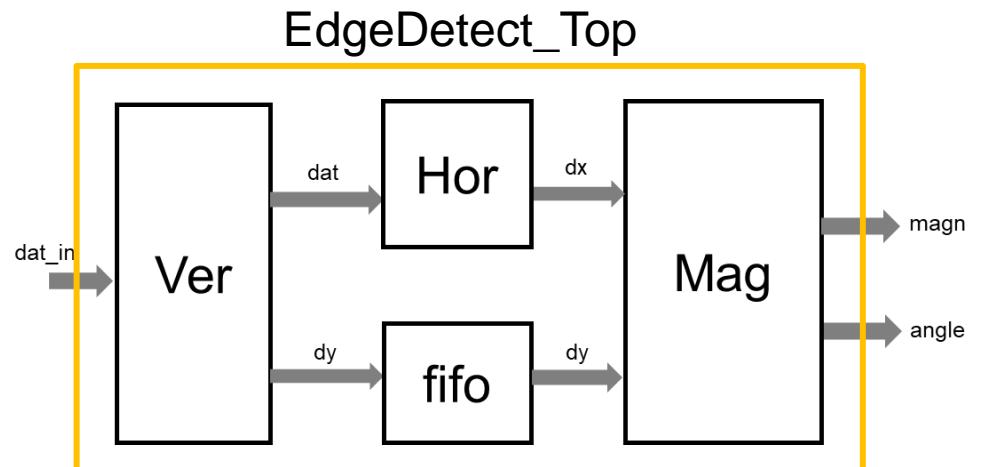
        // Static interconnect channels (FIFOs) between blocks
        ac_channel<gradType>      dy;
        ac_channel<gradType>      dx;
        ac_channel<pixelType>      dat; // channel for passing input

    public:
        EdgeDetect_Top() {}

        #pragma hls_design interface
        void CCS_BLOCK(run)(ac_channel<pixelType> &dat_in,
                            maxWType           &widthIn,
                            maxHType           &heightIn,
                            ac_channel<magType> &magn,
                            ac_channel<angType> &angle)

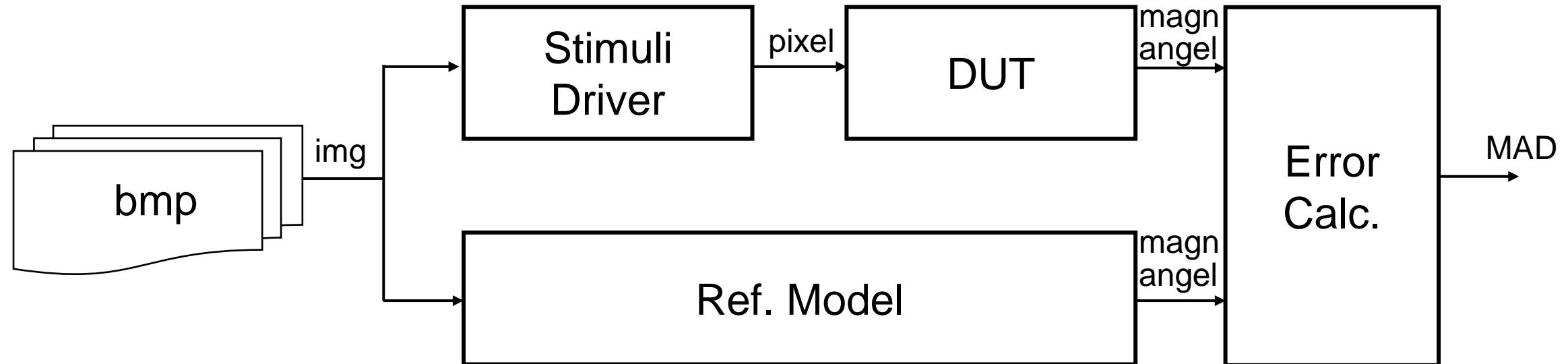
        {
            VerDer_inst.run(dat_in, widthIn, heightIn, dat, dy);
            HorDer_inst.run(dat, widthIn, heightIn, dx);
            MagAng_inst.run(dx, dy, widthIn, heightIn, magn, angle);
        }
    };
}
```

- Design blocks are connected with ac\_channel
- Glue logic is not allowed, only connected in the Top module



# Testbench (EdgeDetect\_tb.cpp)

## Testbench Architecture



# EdgeDetect\_tb.cpp

Alg & DUT instances

Stimuli

Alg & DUT run

Calculate SAD

Print MAD

Write out magn map

```
CCS_MAIN(int argc, char *argv[])
{
    EdgeDetect_Algorithm<iW,iH> ref_inst;
    EdgeDetect_IP::EdgeDetect_Top dut;
    bmp_read((char*)bmpIn.c_str(), &width, &height, &rarray, &garray, &barray);
    unsigned cnt = 0;
    for (int y = 0; y < iH; y++) {
        for (int x = 0; x < iW; x++) {
            dat_in.write(rarray[cnt]); // just using red component (pseudo monochrome)
            dat_in_orig[cnt] = rarray[cnt];
            cnt++;
        }
    }
    ref_inst.run(dat_in_orig,magn_orig,angle_orig);
    dut.run(dat_in,magn,angle);
    for (int y = 0; y < iH; y++) {
        for (int x = 0; x < iW; x++) {
            int hw = magn.read();
            int alg = (int)*(magn_orig+cnt);
            int diff = alg-hw;
            int adiff = abs(diff);
            sumErr += adiff;
            ...
            cnt++;
        }
    }
    printf("Magnitude: Manhattan norm per pixel %f\n",sumErr/(iH*iW));
    printf("Angle: Manhattan norm per pixel %f\n",sumAngErr/(iH*iW));
    bmp_24_write((char*)bmpAlg.c_str(), iW, iH, garray, garray, garray);
    bmp_24_write((char*)bmpBA.c_str(), iW, iH, rarray, rarray, rarray);
    CCS_RETURN(0);
}
```

# C++ Verification by gcc

## Makefile

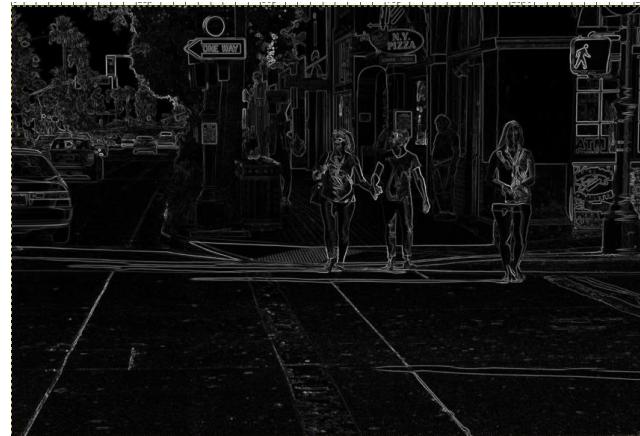
- g++ -g -O3 -std=c++11 -I \$MGC\_HOME/shared/include -I ./bmpUtil/inc -I ./cmodel/inc -I ./hls\_c/inc ./bmpUtil/src/bmp\_io.cpp .../hls\_c/src/EdgeDetect\_tb.cpp -o SCVerify\_EdgeDetect.exe

## Execute

- ./ SCVerify\_EdgeDetect.exe <inputbmp> <outputbmp\_alg> <outputbmp\_ba>  
Ex. ./ SCVerify\_EdgeDetect.exe ./image/people\_gray.bmp out\_algorithm.bmp out\_hw.bmp



people\_gray.bmp



out\_algorithm.bmp



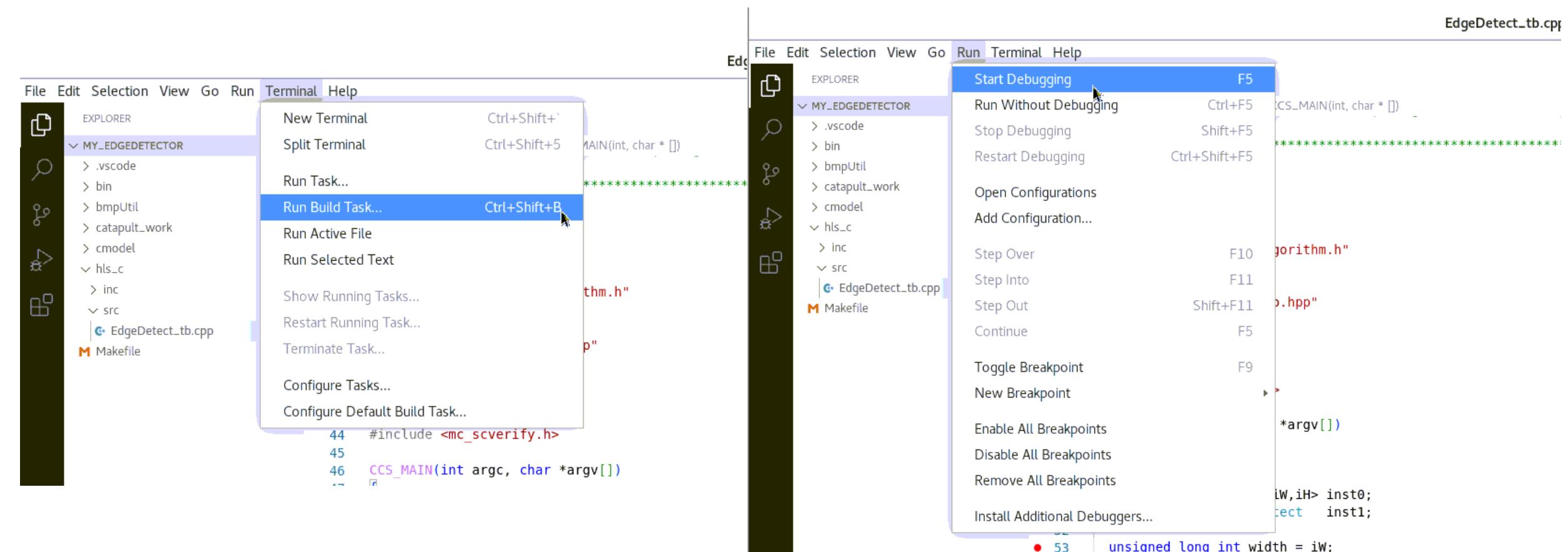
out\_hw.bmp

# VSCode Environment

A friendly IDE for source code editor, debugger, ...

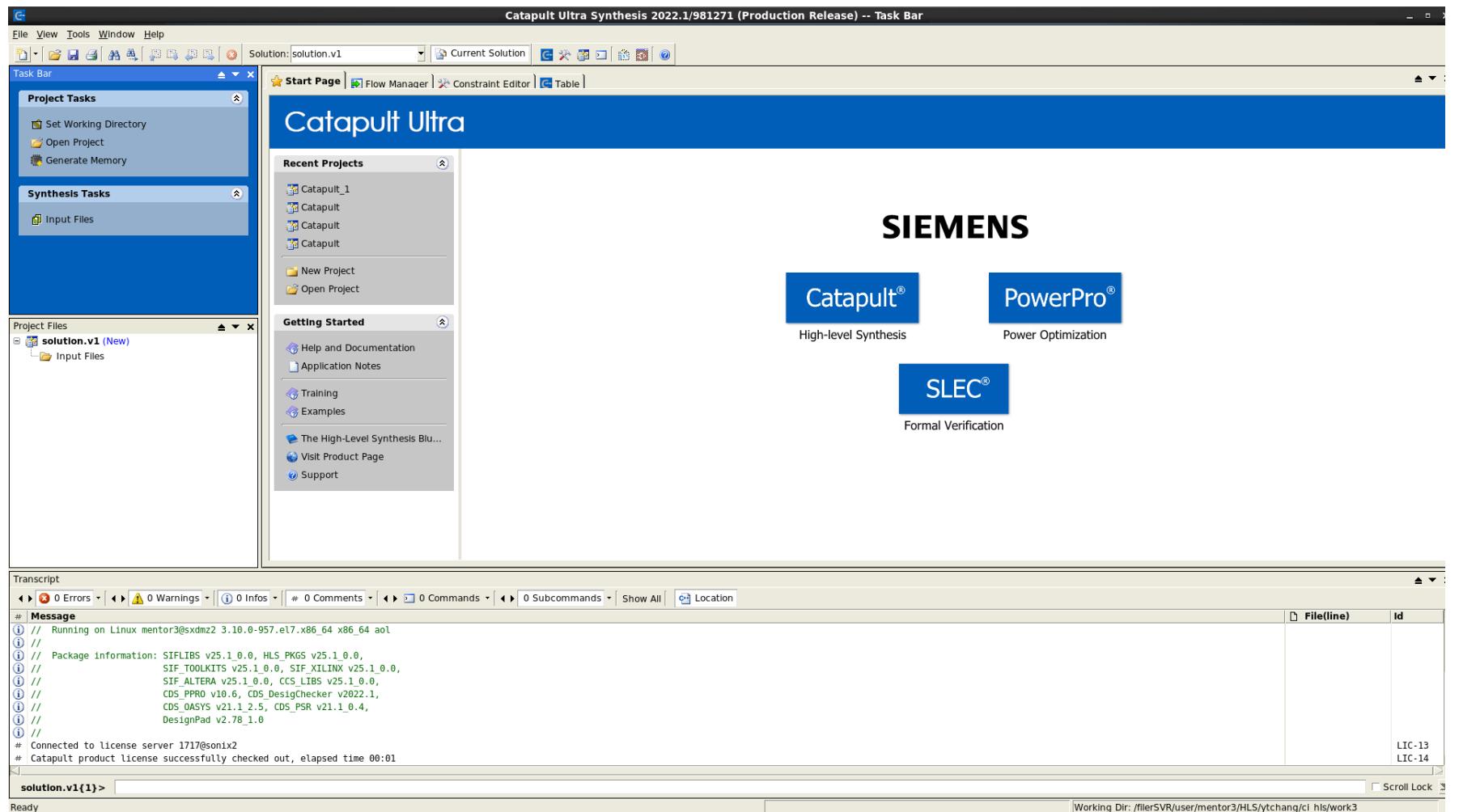
VSCode project has been built, call it by ‘code . &’

```
hang@a4sa46 My EdgeDetector]$ code . &
```



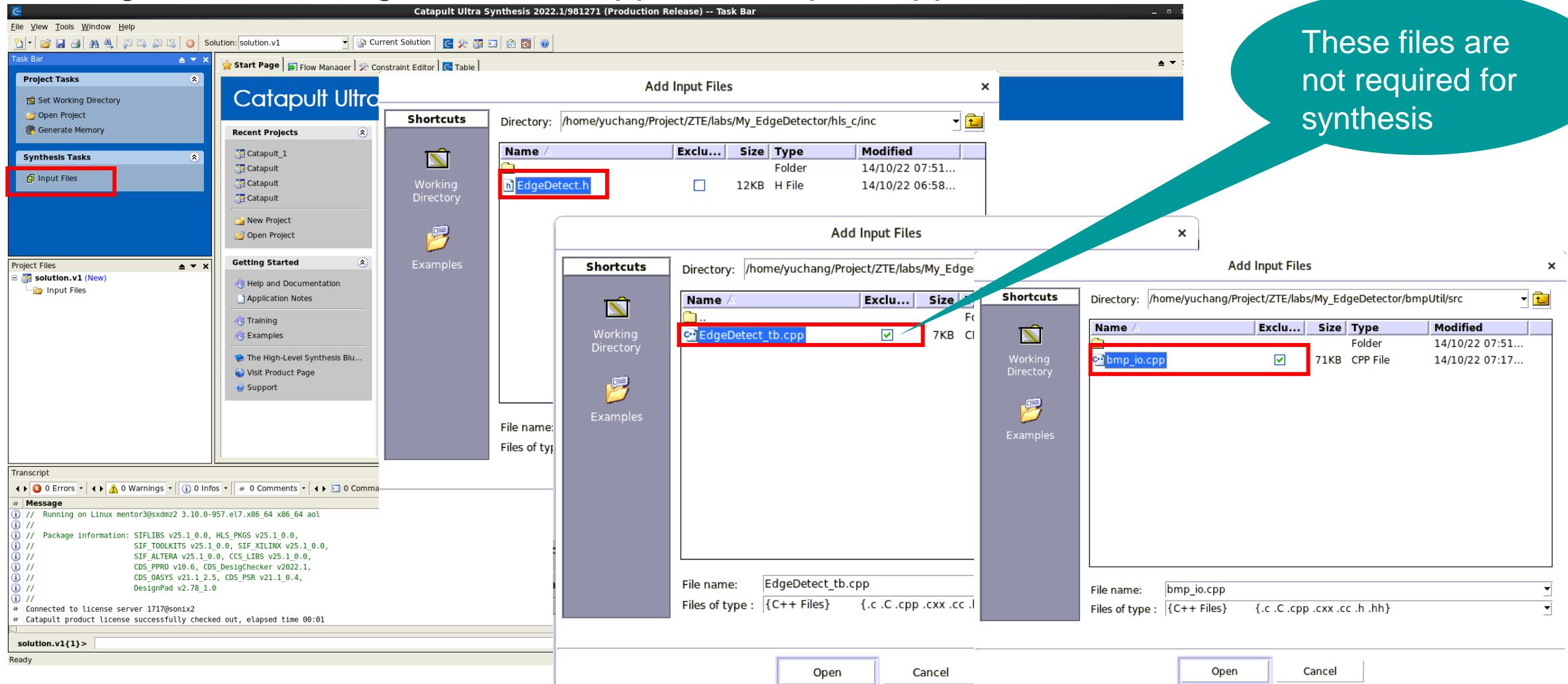
# Launch Catapult

catapult &

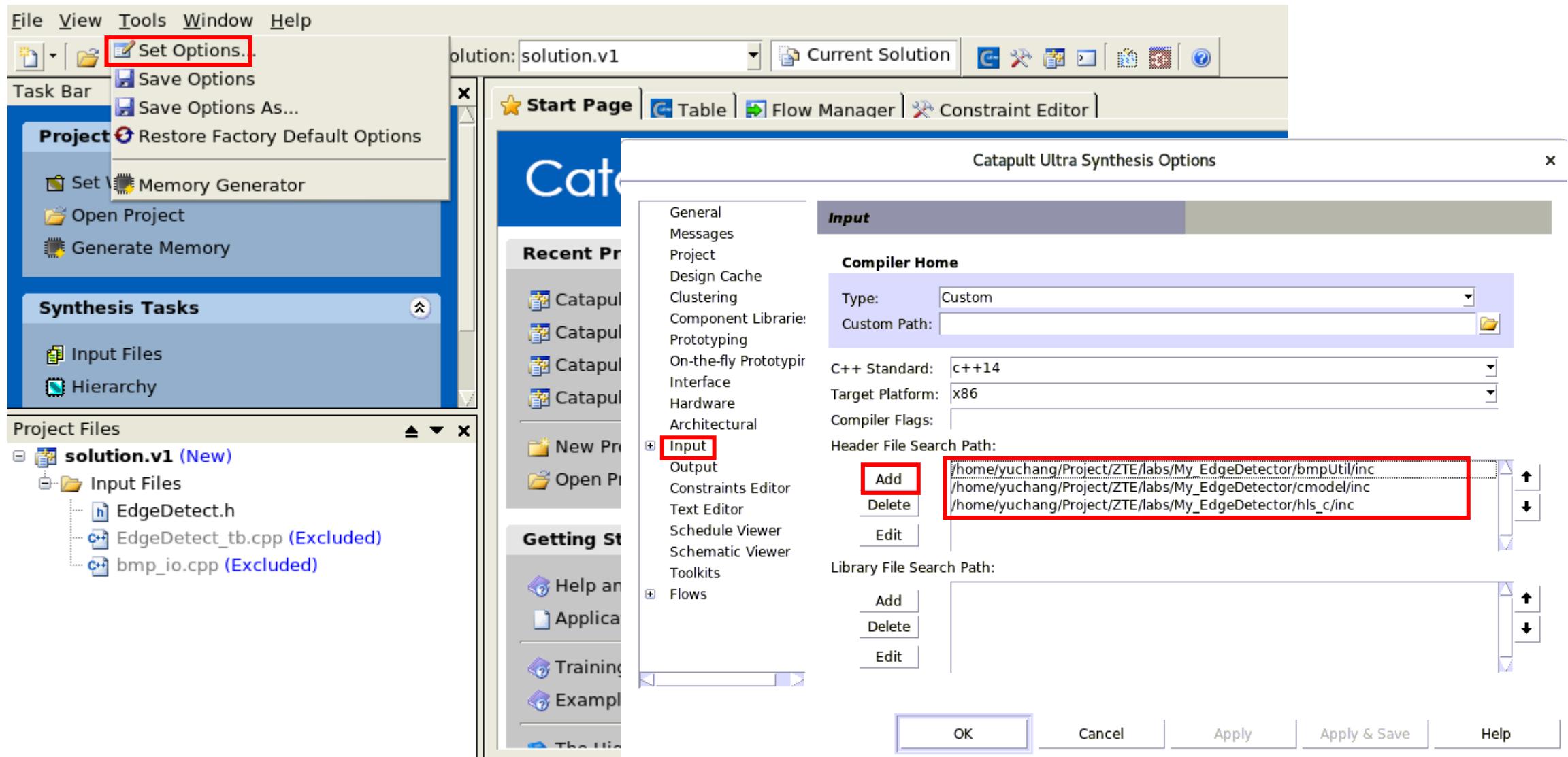


# Add input file

Add EdgeDetect.h, EdgeDetect\_tb.cpp, and bmp\_io.cpp



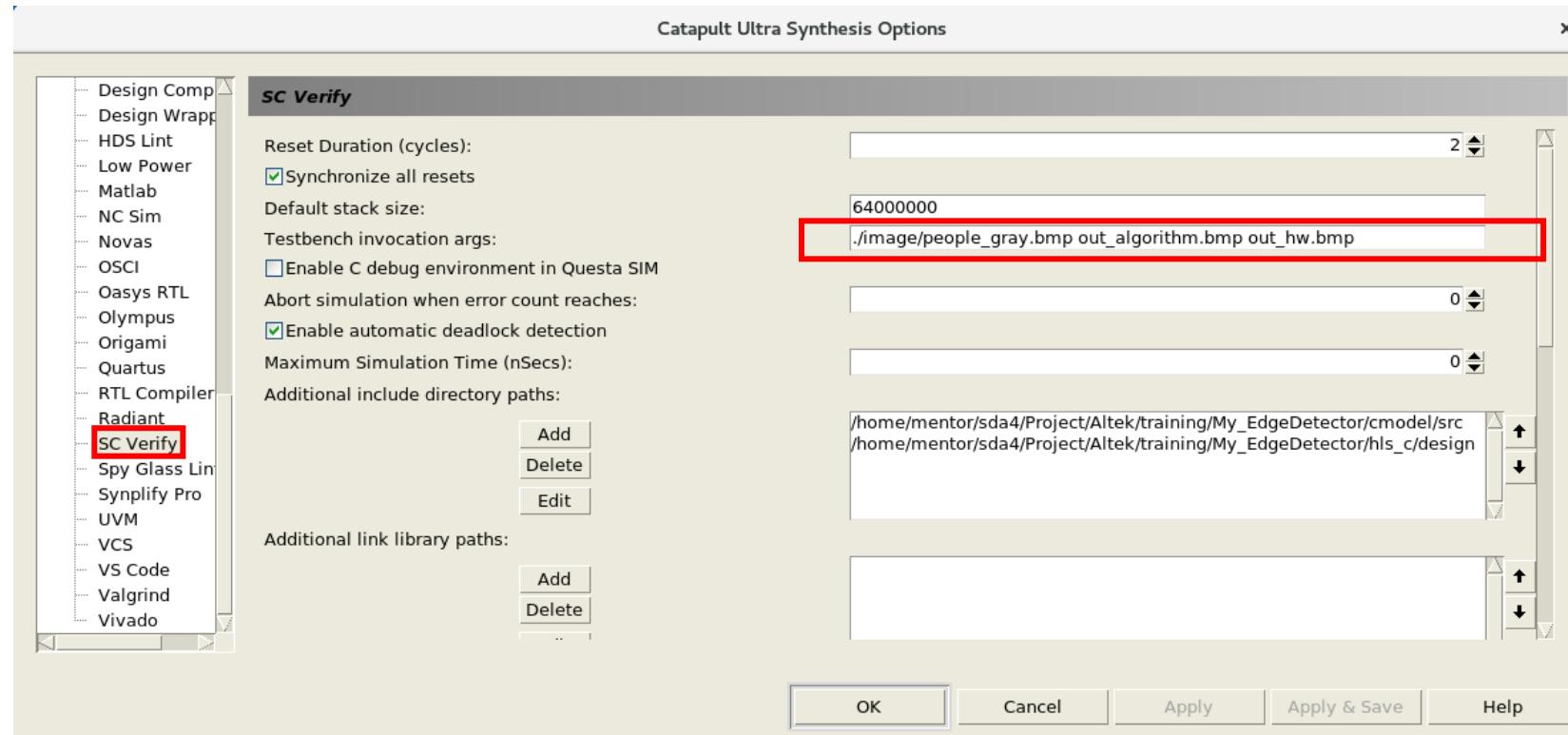
# Add include path



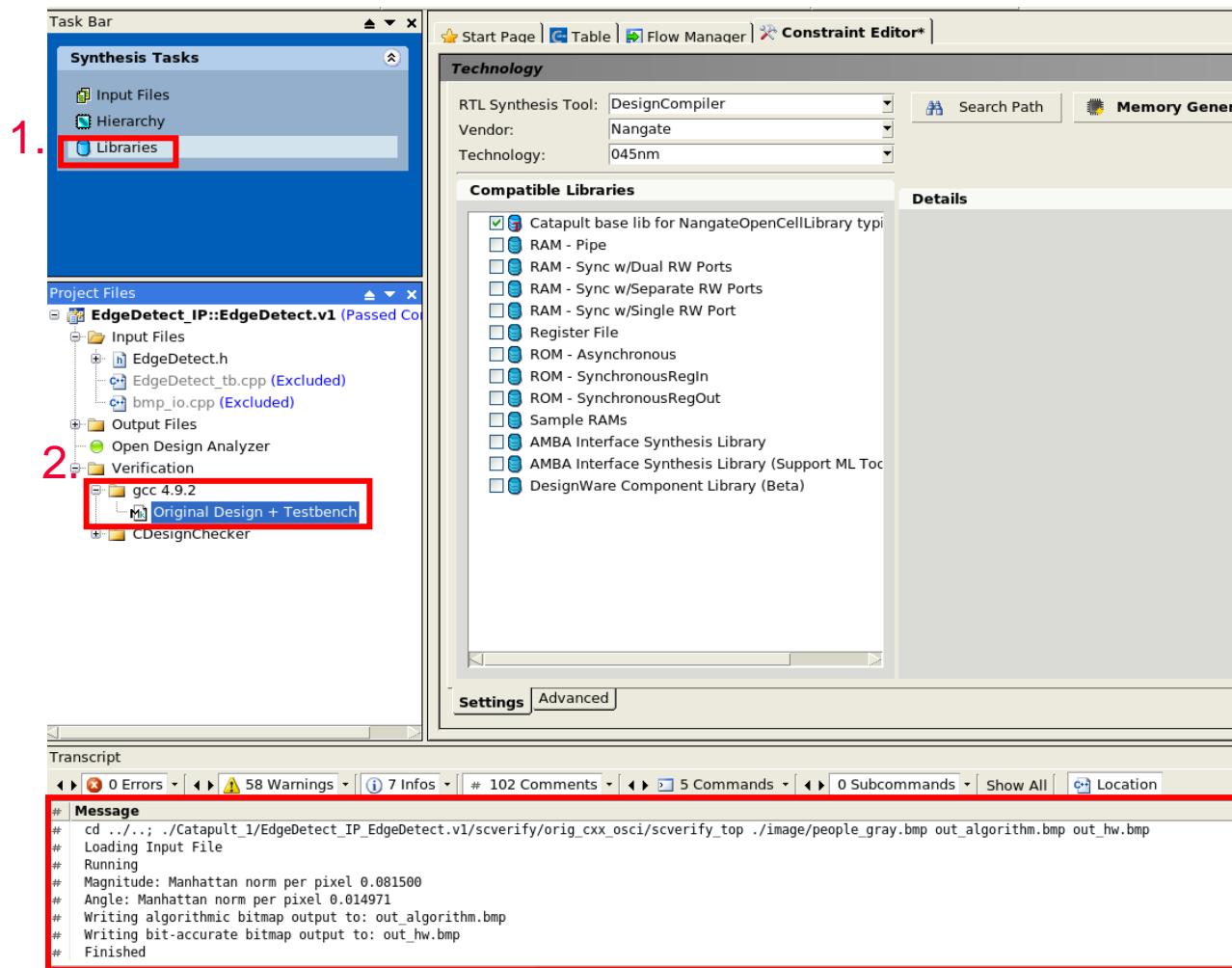
# Add testbench invocation args

Assign arguments for running the testbench

- SCVerify\_EdgeDetect.exe ./image/people\_gray.bmp out\_algorithm.bmp out\_hw.bmp



# Run C++ Verification (Pre-HLS Verification)



# Run C++ Verification (Cont.)

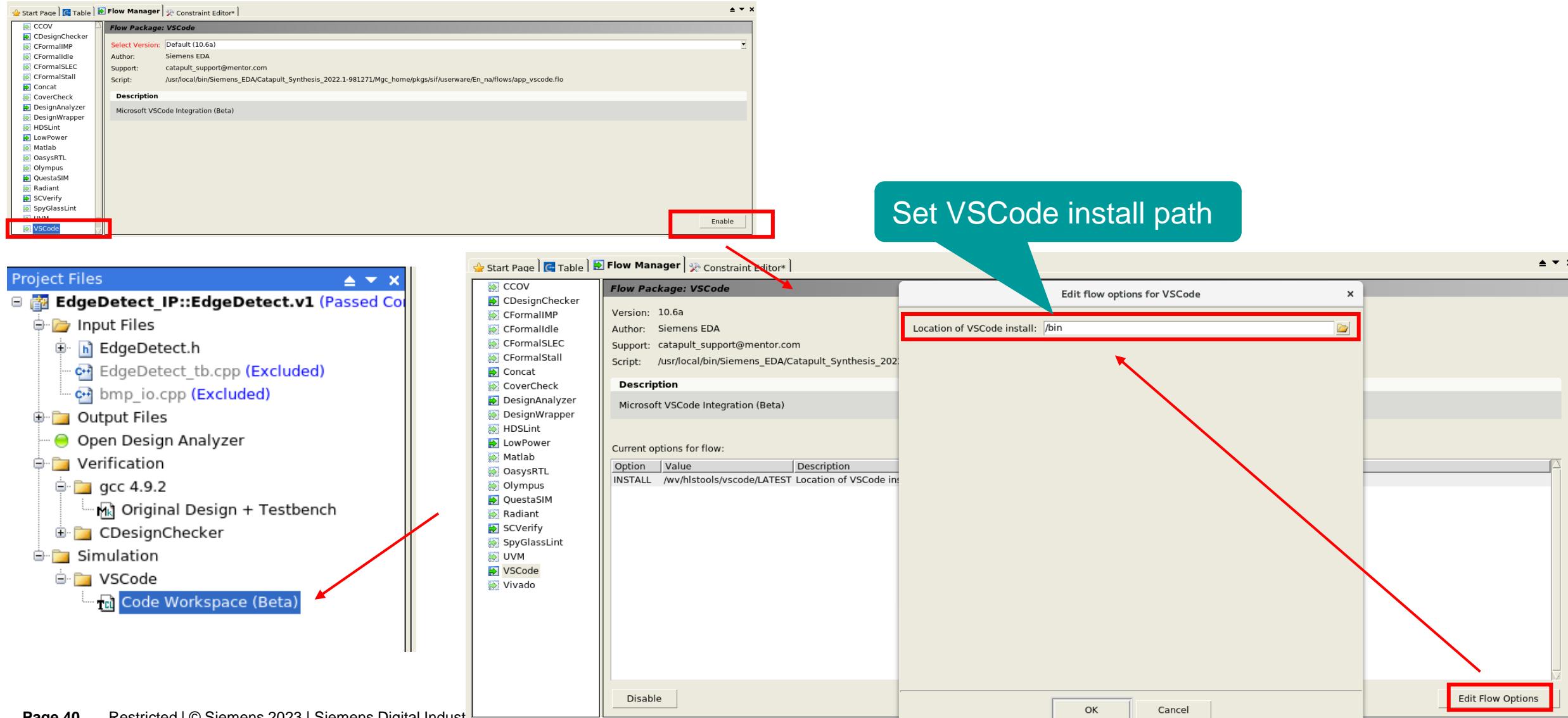
The execution file is generated and can be found in this path

```
[yuchang@a4sa46 catapult_work]$ ll ./Catapult/EdgeDetect/IP/EdgeDetect.v1/scverify.orig.cxxosci/scverify_top
```

You can copy this execution file to your environment and do verification

```
[yuchang@a4sa46 bin]$ ./SCVerify_EdgeDetect.exe ./image/people_gray.bmp out_algorithm.bmp out_hw.bmp
Loading Input File
Running
Magnitude: Manhattan norm per pixel 0.081500
Angle: Manhattan norm per pixel 0.014971
Writing algorithmic bitmap output to: out_algorithm.bmp
Writing bit-accurate bitmap output to: out_hw.bmp
Finished
[yuchang@a4sa46 bin]$
```

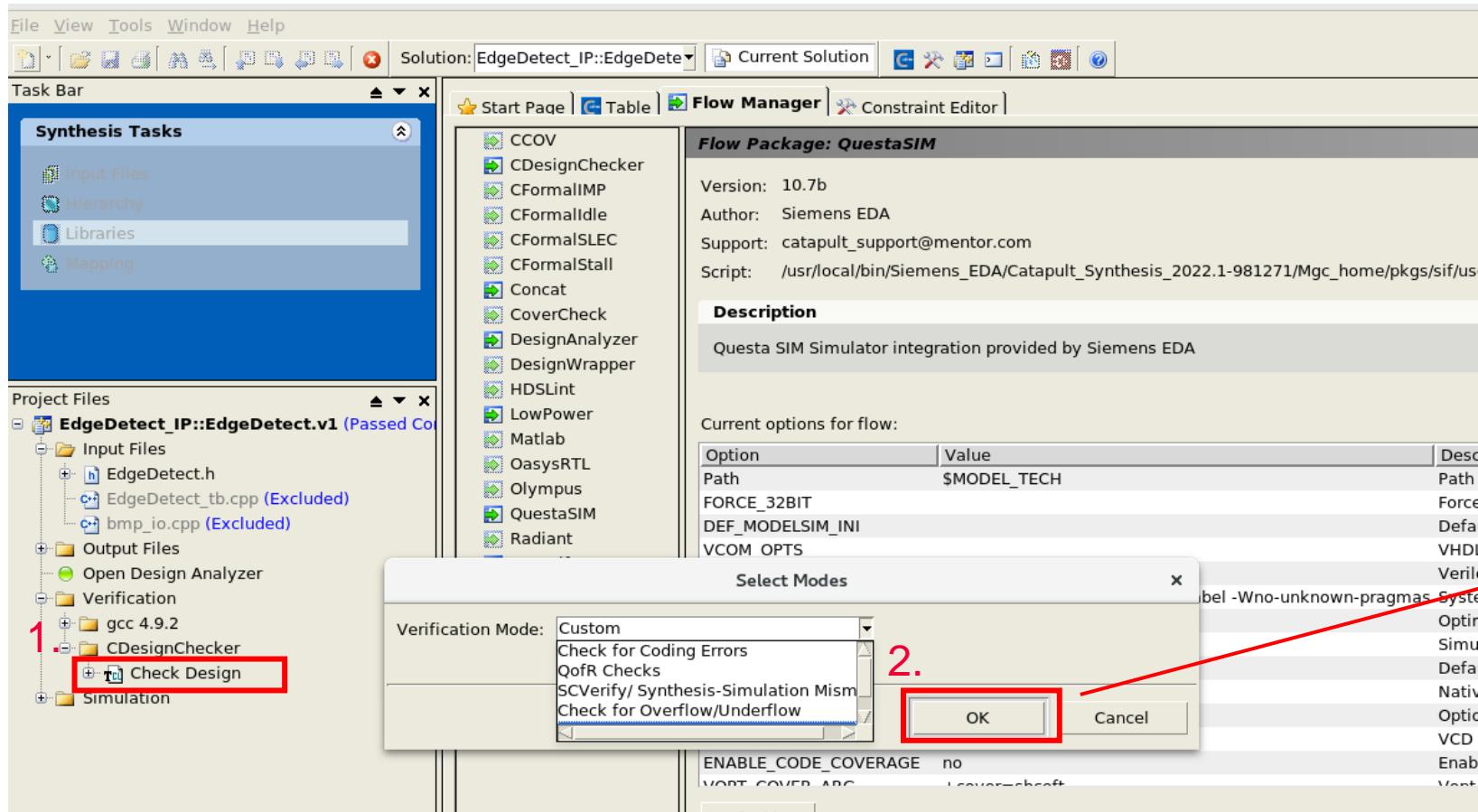
# Catapult can help build VSCode environment



# Run CDesignChecker

Static check for your hls c++ design (Lint)

Very important to find the design issue before synthesis



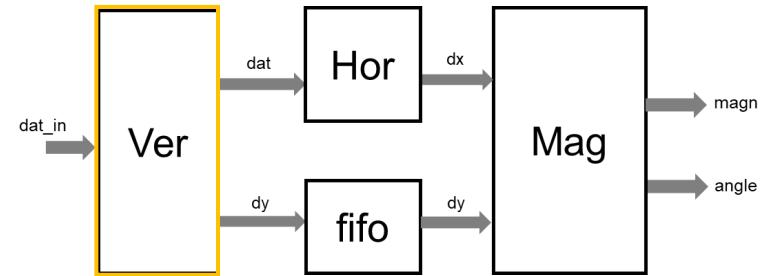
# CDesignChecker Report

Design_Check.rpt		
	Violated	Waived
31 FXD	FXD - Mixed fixed and non-fixed datatypes	DISABLED
32 ISE	ISE - Illegal Shift Error	CHECKED
33 LRC	LRC - Long Reset Cycle	DISABLED
34 MDB	MDB - Missing Default Branch	DISABLED
35 MXS	MXS - Mixed signed and unsigned datatypes	DISABLED
36 NCO	NCO - No Contribution to Output	DISABLED
37 OSA	OSA - Optimal Size Accumulator	DISABLED
38 OVL	OVL - Overflow/Underflow	DISABLED
39 PDD	PDD - Platform dependent datatype (long)	DISABLED
40 RIU	RIU - Rolled loop Inside Unrolled loop	DISABLED
41 RRT	RRT - Reset referenced in thread	DISABLED
42 SAT	SAT - Sub-optimal Adder Tree	DISABLED
43 STF	STF - Funcs with statics called multiple times	DISABLED
44 SUD	SUD - Suboptimal Use of Divide and Modulus Operator	DISABLED
45 UMR	UMR - Uninitialized Memory Read	CHECKED
46 FATAL		
47	Violated	Waived
48		
49		
50 ERROR		
51	Violated	Waived
52 ABR - Array Bounds Read	0	0
53 ABW - Array Bounds Write	0	0
54 CAS - Incomplete Switch-Case	0	0
55 DBZ - Divide By Zero	0	0
56 ISE - Illegal Shift Error	0	0
57 UMR - Uninitialized Memory Read	2	2
58 WARNING		
59	Violated	Waived
60		
61 INFO		
62	Violated	Waived
63		
64		
65 DISABLED		
66		
67 ACC - Accumulator of native C type		

# Synthesize EdgeDetect\_VerDer

Bot-up synthesis

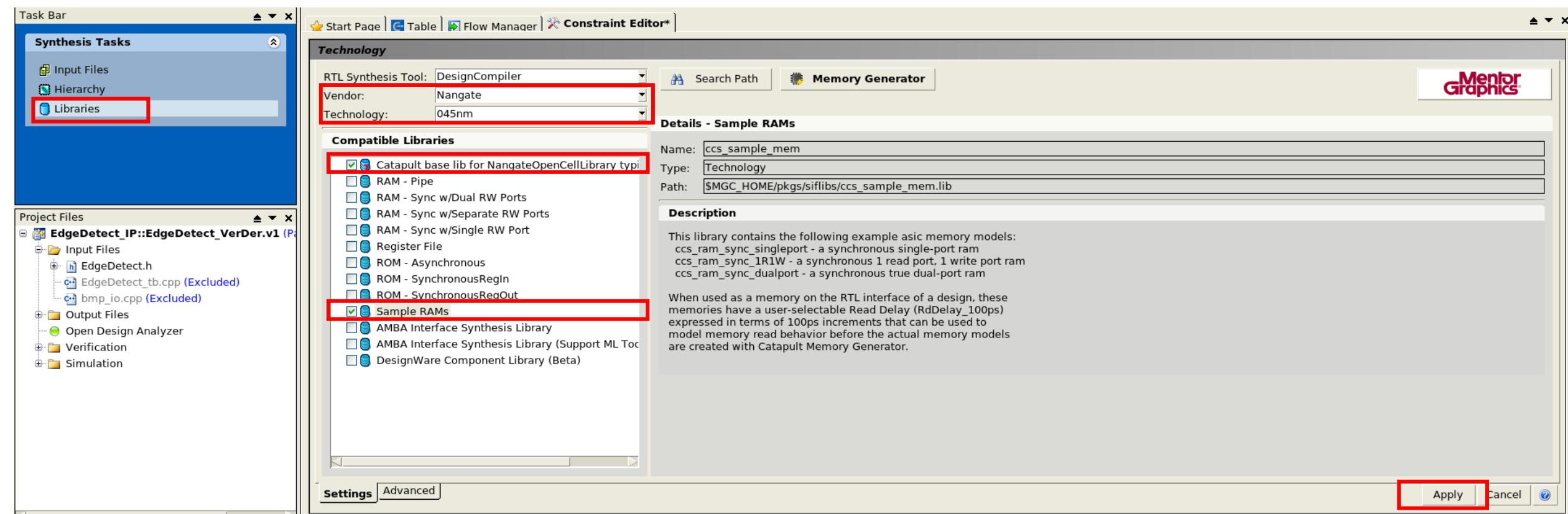
- \*\_Ver -> \*\_Hor -> \*\_Mag -> \*\_Top



The screenshot shows the Synthesis Tasks interface in a software tool. The left panel displays the 'Synthesis Tasks' tree, where 'Hierarchy' is selected and highlighted with a red box. The central panel shows the 'Function: EdgeDetect\_IP::EdgeDetect\_VerDer' details. Under 'Hierarchy Setting', the 'Top' radio button is selected and highlighted with a red box. The right panel contains sections for 'Design block attributes' (with 'Module' checked) and 'Design block CCORE attributes' (with 'CCORE' checked). At the bottom right of the dialog, the 'Apply' button is highlighted with a red box.

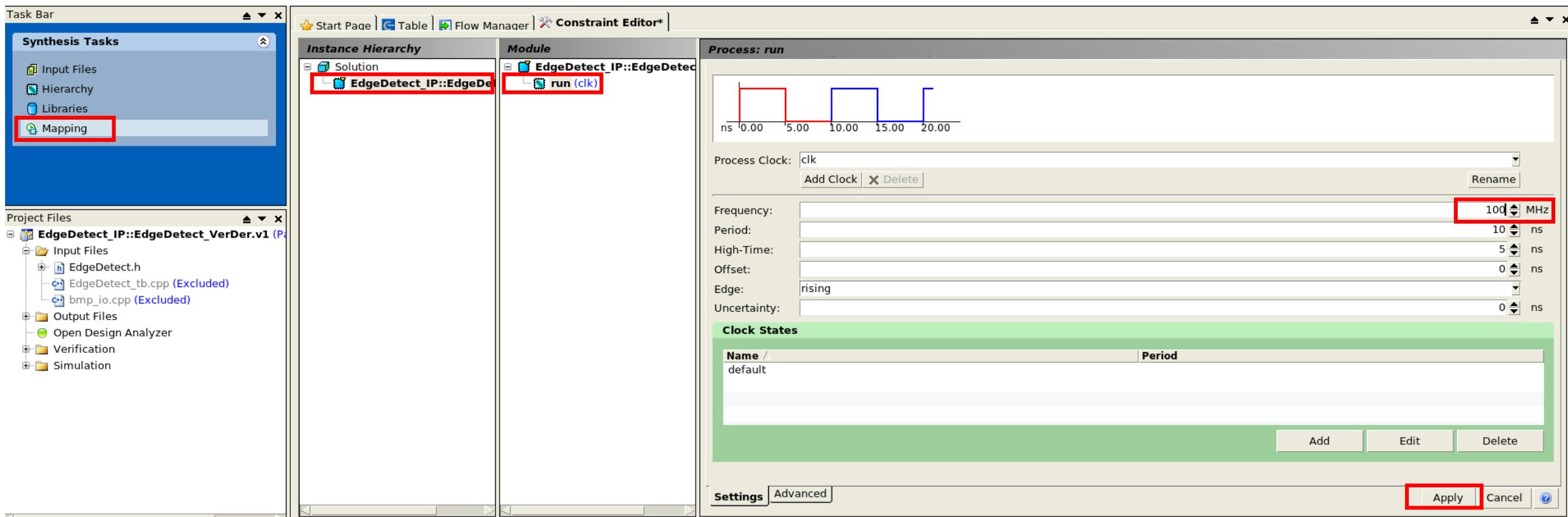
# Library Setup

## Use Nangate ASIC library and Sample RAMs



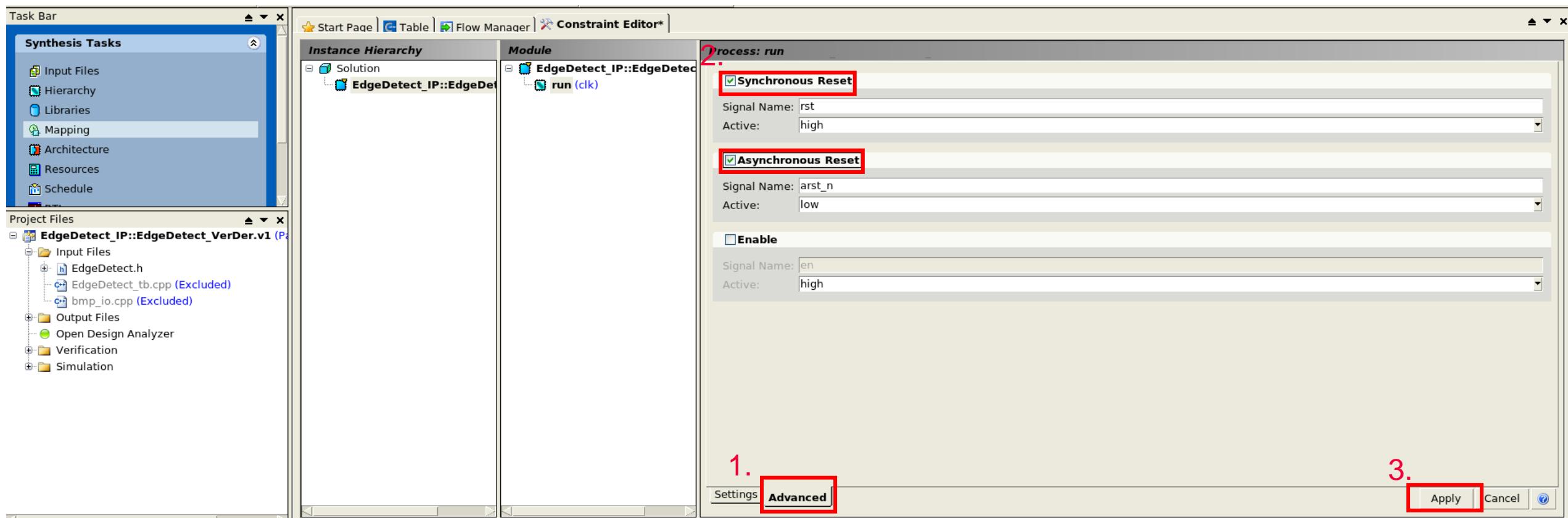
# Mapping

Set frequency as 100MHz



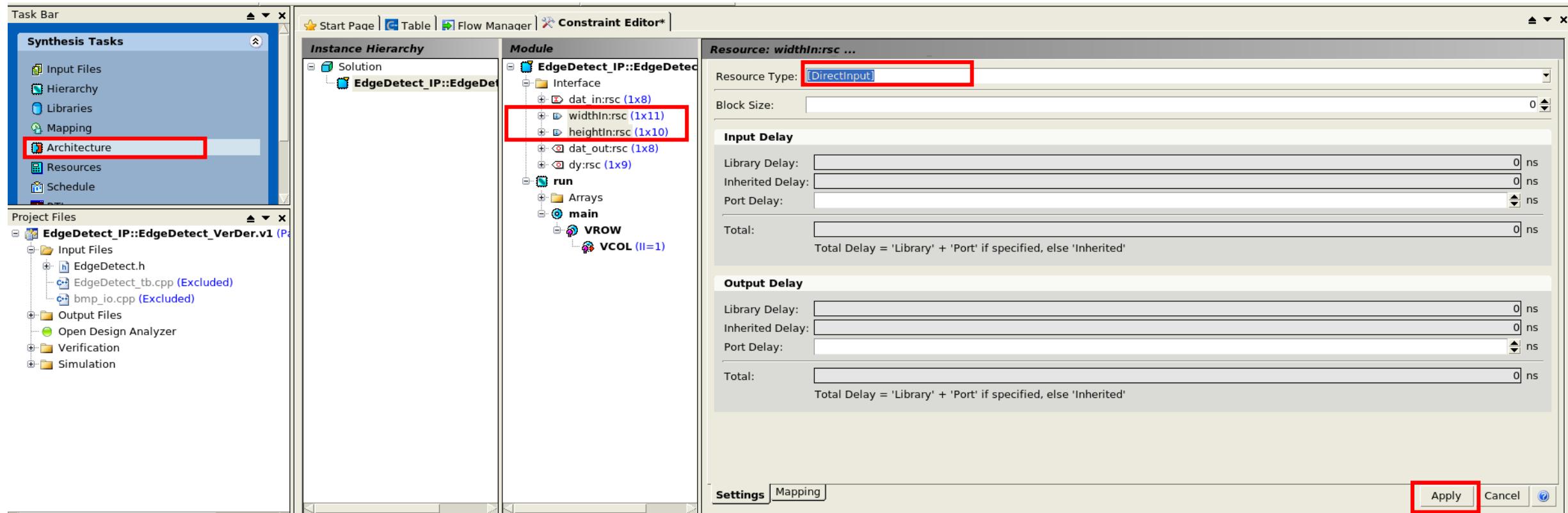
# Mapping (Cont.)

Add synchronous and asynchronous reset



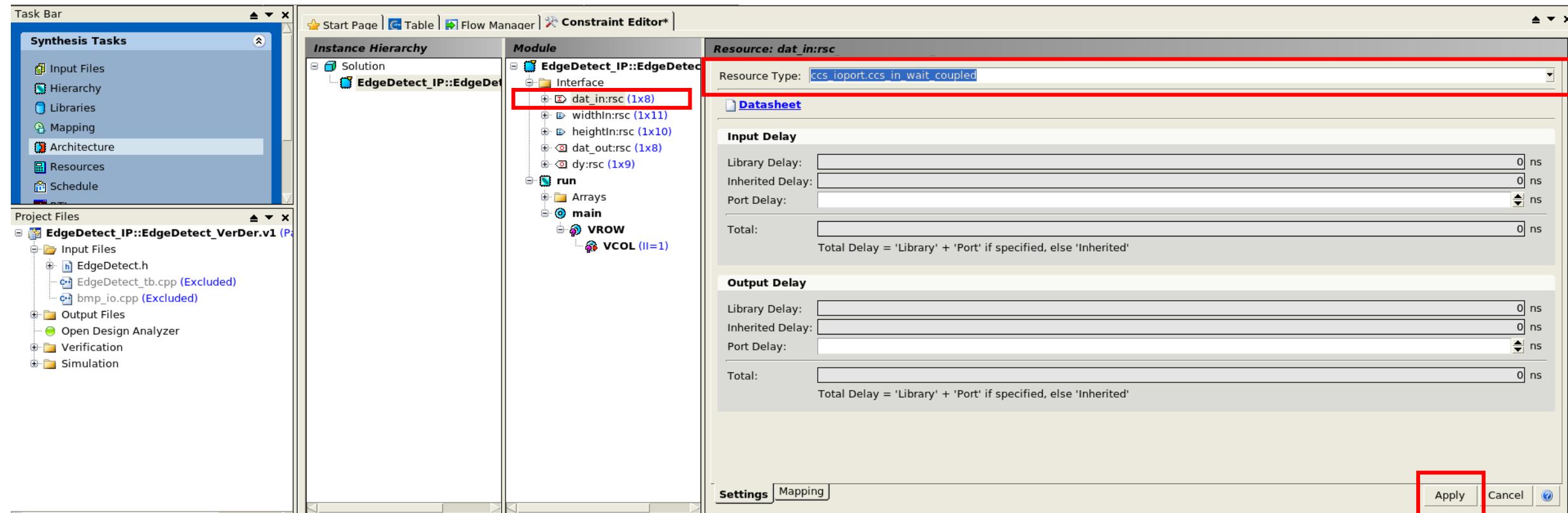
# Architecture

DirectInput is used for the stable ports avoiding the unnecessary pipeline registers inserted



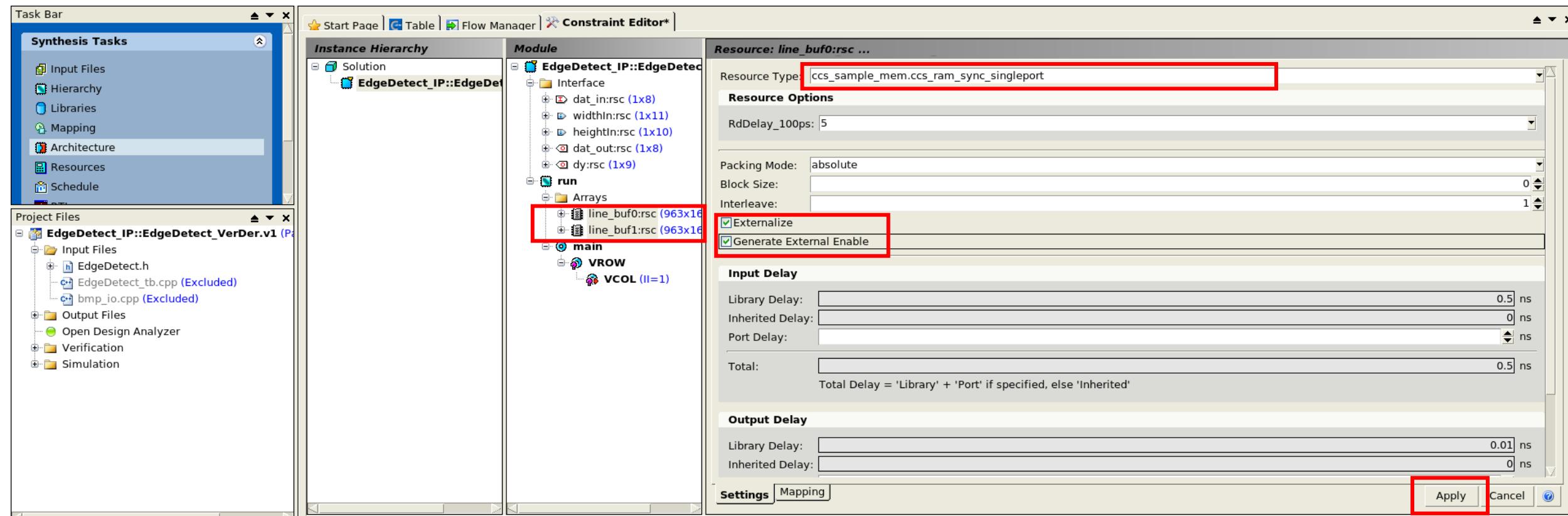
# Architecture (Cont.)

data input channel is set as ccs\_in\_wait\_coupled avoiding extra fifo buffer



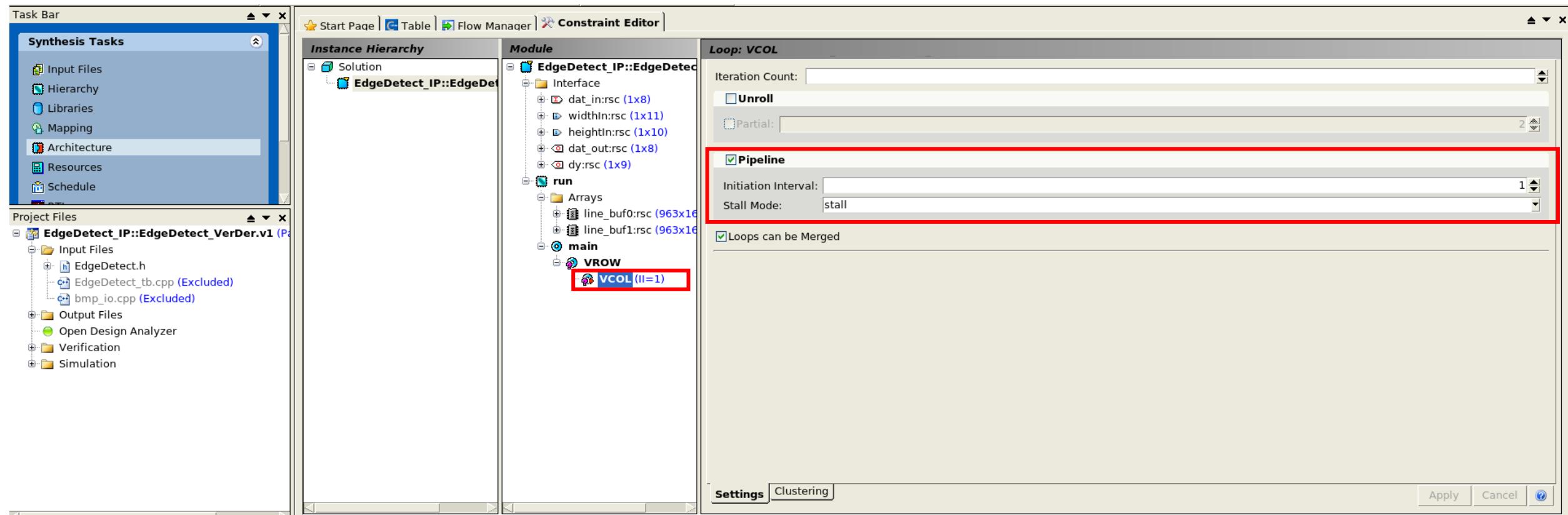
# Architecture (Cont.)

Array line\_buf0 and line\_buf1 are mapped to SinglePort RAMs and externalized



# Architecture (Cont.)

Pipeline initial interval 1 is at the loop VCOL (throughput will be 1 at this loop)



## Architecture (Cont.)

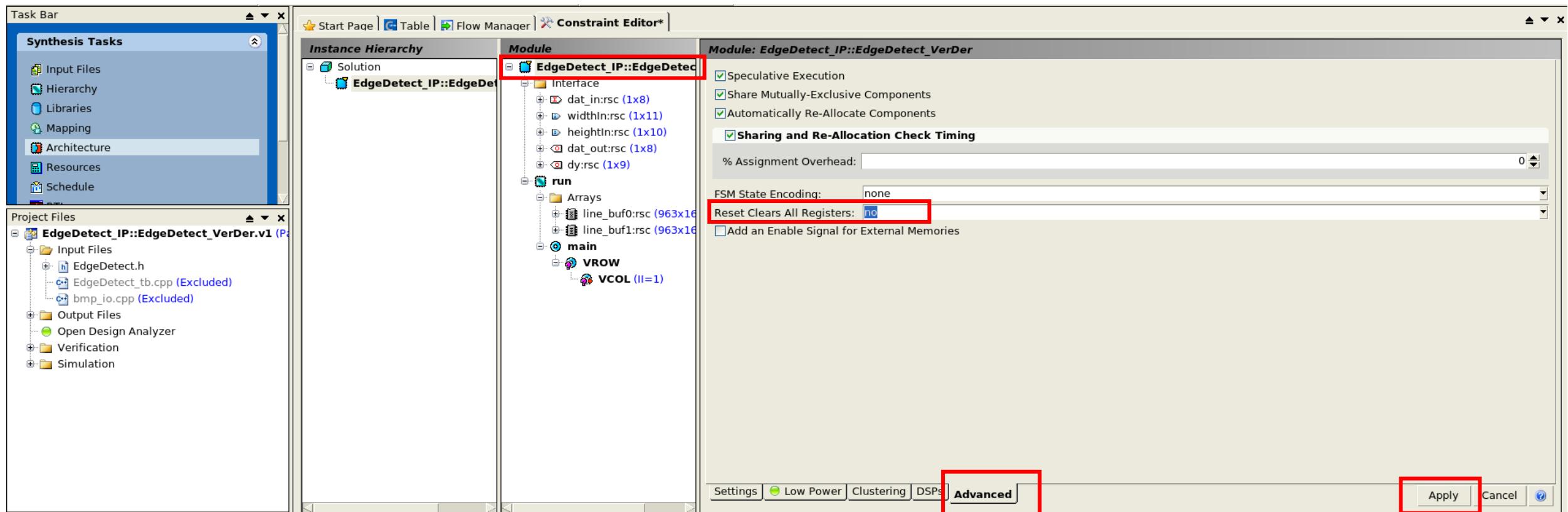
An alternative to set II1 is using the pragma `hls_pipeline_init_interval` 1

```
VROW: for (maxHType y = 0; ; y++) { // VROW has  
#pragma hls_pipeline_init_interval 1  
VCOL: for (maxWType x = 0; ; x++) {  
    if (y <= heightIn-1) {  
        pix0 = dat_in.read(); // Read streaming  
    }  
    (11) Wait for the read operation to complete
```

# Architecture (Cont.)

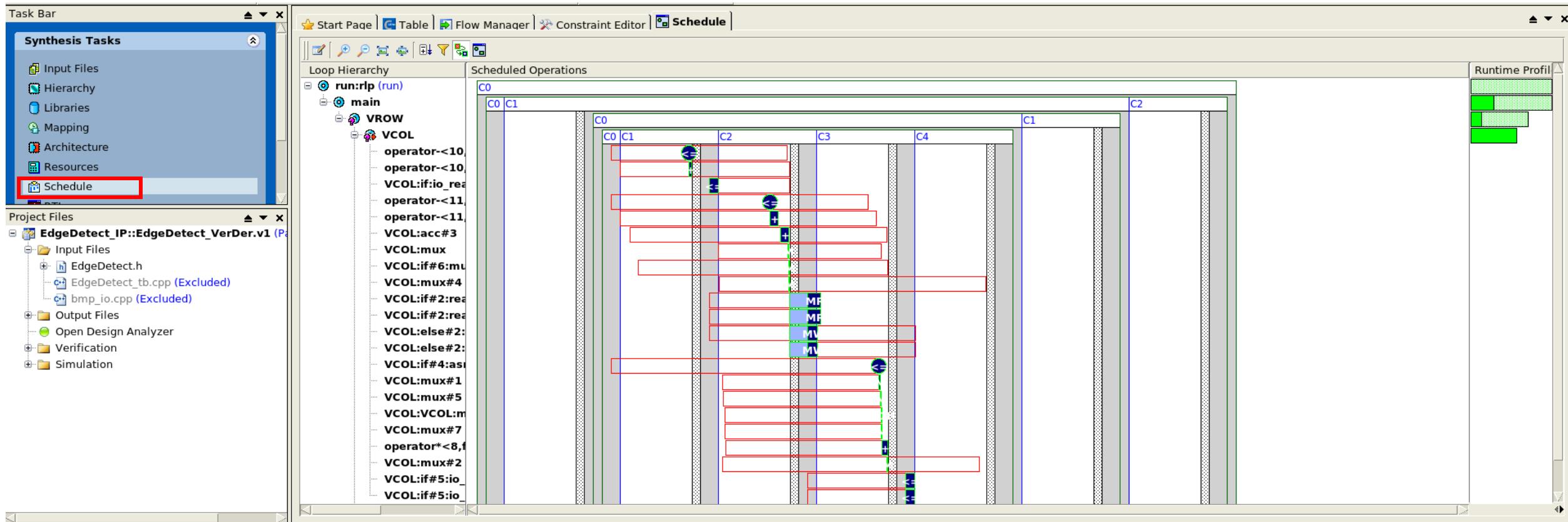
You can turn off resets on all registers for area reduction

- Only registers that need to be reset for correct simulation result are reset



# Scheduling

Pipeline stages are created



# Generate RTL

The screenshot shows a CAD software interface with the following components:

- Task Bar:** On the left, it lists "Libraries", "Mapping", "Architecture", "Resources", "Schedule", "RTL" (which is highlighted with a red box), "Power Report (Pre Power Opt)", and "Power RTL".
- Project Files:** Below the Task Bar, the "EdgeDetect\_IP::EdgeDetect\_VerDer.vl" project file is expanded. It contains:
  - Input Files:** "EdgeDetect.h", "EdgeDetect\_tb.cpp" (Excluded), and "bmp\_io.cpp" (Excluded).
  - Output Files:** Reports, Schedule, Schematics, VHDL, SCVerify, Verilog, and Simulation.
  - Verilog:** "rtl.v" (highlighted with a red box) and its extracts: "concat\_rtl.v (extract)" and "concat\_sim\_rtl.v (extract)".
  - Open Design Analyzer:**
  - Verification:**
  - Simulation:**
- Code Editor:** On the right, the "rtl.v" file is open in the editor. A red box highlights the SRAM interface section of the code. The code is as follows:

```
1085
1086
1087 module EdgeDetect_IP_EdgeDetect_VerDer (
1088     clk, rst, arst_n, dat_in_rsc_dat, dat_in_rsc_vld, dat_in_rsc_rdy, widthIn, heightIn,
1089     dat_out_rsc_dat, dat_out_rsc_vld, dat_out_rsc_rdy, dy_rsc_dat, dy_rsc_vld,
1090     dy_rsc_rdy, line_buf0_rsc_en, line_buf0_rsc_q, line_buf0_rsc_we, line_buf0_rsc_d,
1091     line_buf0_rsc_adr, line_buf1_rsc_en, line_buf1_rsc_q, line_buf1_rsc_we, line_buf1_rsc_d,
1092     line_buf1_rsc_adr
1093 );
1094     input clk;
1095     input rst;
1096     input arst_n;
1097     input [7:0] dat_in_rsc_dat;
1098     input dat_in_rsc_vld;
1099     output dat_in_rsc_rdy;
1100     input [10:0] widthIn;
1101     input [9:0] heightIn;
1102     output [7:0] dat_out_rsc_dat;
1103     output dat_out_rsc_vld;
1104     input dat_out_rsc_rdy;
1105     output [8:0] dy_rsc_dat;
1106     output dy_rsc_vld;
1107     input dy_rsc_rdy;
1108     output line_buf0_rsc_en;
1109     input [15:0] line_buf0_rsc_q;
1110     output line_buf0_rsc_we;
1111     output [15:0] line_buf0_rsc_d;
1112     output [9:0] line_buf0_rsc_adr;
1113     output line_buf1_rsc_en;
1114     input [15:0] line_buf1_rsc_q;
1115     output line_buf1_rsc_we;
1116     output [15:0] line_buf1_rsc_d;
1117     output [9:0] line_buf1_rsc_adr;
1118
1119
1120 // Interconnect Declarations
1121 wire [15:0] line_buf0_rsc_i_d_d;
```

A blue box labeled "SRAM interface" is placed over the SRAM interface declarations (lines 1108 to 1117).

# Synthesis Report

The screenshot shows a CAD software interface with the following components:

- Task Bar:** Located at the top left, it includes icons for Libraries, Mapping, Architecture, Resources, Schedule, RTL, Power Report (Pre Power Opt), and Power RTL.
- Project Files:** Located at the bottom left, it displays the project structure:
  - EdgeDetect\_IP::EdgeDetect\_VerDer.v1
    - Input Files: EdgeDetect.h, EdgeDetect\_tb.cpp (Excluded), bmp\_io.cpp (Excluded)
    - Output Files: Reports, Schedule, Schematics, VHDL, SCVerify, Verilog (containing rtl.v, concat\_rtl.v (extract), concat\_sim\_rtl.v (extract)), Open Design Analyzer, Verification, Simulation
- Table View:** The main area shows a table with the following data:

Report	General	Run Time	Memory Usage	Timing	Throughput	Slack	Total Area
General				3	30.00	7	70.00
Run Time							8.22
Memory Usage							1502.08
Timing							
Area Score							

The "Table" tab in the menu bar is highlighted with a red box. The "General" report row in the table is also highlighted with a red box.

# Synthesis Report (Cont.)

## Detailed information

- Bill of Materials (BOM), Area Scores, Register-to-Variable, ...

The screenshot shows a software interface for a digital design project. The left side features a 'Task Bar' with icons for Libraries, Mapping, Architecture, Resources, Schedule, RTL (which is selected), Power Report (Pre Power Opt), and Power RTL. Below it is a 'Project Files' tree view for the 'EdgeDetect\_IP::EdgeDetect\_VerDer.v1' project. Under Input Files, 'EdgeDetect.h' and 'EdgeDetect\_tb.cpp' are listed as excluded files. Under Output Files, Reports are expanded, showing sub-folders for Messages, Commands, and RTL, with 'RTL' highlighted by a red box. Other visible output files include Schedule, Schematics, VHDL, SCVerify, and Verilog.

The main workspace displays two tables. The top table is a summary of processes and their resource usage:

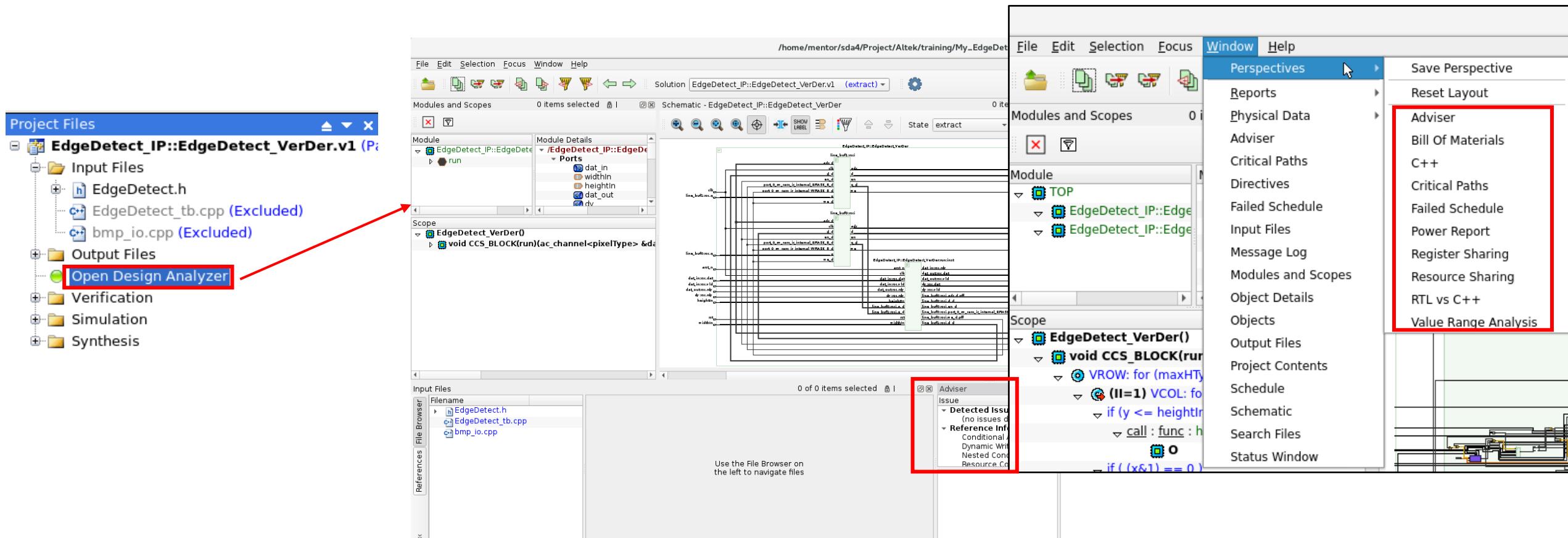
Process	Real Operation(s)	count	Latency	Throughput	Reset	Length	II	Comments
/EdgeDetect_IP::EdgeDetect_VerDer/run	31	3	7	0	0	?		
Design Total:	31	3	7	0	0			

The bottom table is a detailed 'Bill Of Materials (Datapath)' report:

Component Name	Area Score	Area(Combinational)	AreaSeq	Delay	Post Alloc	Post Assign
[Lib: VIRTUAL]						
VIRTUAL.VCOL:equal	20.493	0.000	0.000	0.290	1	0
VIRTUAL.VCOL:equal#1	22.489	0.000	0.000	0.293	1	0
VIRTUAL.VROW:equal	20.493	0.000	0.000	0.290	1	0
[Lib: ccs_ioport]						
ccs_in_wait_coupled(1,8)	0.000	0.000	0.000	0.000	1	1
ccs_out_wait(4,8)	0.000	0.000	0.000	0.000	1	1
ccs_out_wait(5,9)	0.000	0.000	0.000	0.000	1	1
[Lib: ccs_sample_mem]						
ccs_ram_sync_singleport_rwport_en(6,16,10,963,963,16,5)	0.000	0.000	0.000	0.500	1	0
ccs_ram_sync_singleport_rwport_en(7,16,10,963,963,16,5)	0.000	0.000	0.000	0.500	1	0
[Lib: nangate-45nm_beh]						
mgc_add(10,0,2,1,10,7)	45.747	0.000	0.000	0.807	1	0
mgc_add(10,0,8,0,10,7)	39.368	0.000	0.000	0.698	0	1
mgc_add(11,0,1,0,11,7)	30.856	0.000	0.000	0.486	0	1
mgc_add(11,0,2,1,11,7)	50.178	0.000	0.000	0.879	2	1
mgc_add(9,0,8,0,9,7)	36.708	0.000	0.000	0.664	1	0
mgc_add_msb(11,0,10,0,11,6)	24.806	0.000	0.000	0.277	1	1
mgc_and(1,2,4)	1.064	0.000	0.000	0.077	0	30
mgc_and(1,3,4)	1.330	0.000	0.000	0.084	0	8
mgc_and(1,4,4)	1.596	0.000	0.000	0.090	0	7
mgc_and(1,5,4)	2.062	0.547	0.000	0.096	0	1
mgc_and(10,2,4)	10.640	11.128	0.000	0.077	0	2
mgc_and(11,2,4)	11.704	12.630	0.000	0.077	0	1
mgc_equal(11,4)	22.743	22.505	0.000	0.302	0	1
mgc_equal(12,4)	24.738	25.101	0.000	0.306	0	1
mgc_mux(1,1,2,5)	0.000	0.000	0.000	0.075	0	2
mgc_mux(10,1,2,5)	23.940	8.015	0.000	0.126	0	2
mgc_mux(11,1,2,5)	23.142	11.063	0.000	0.132	1	1

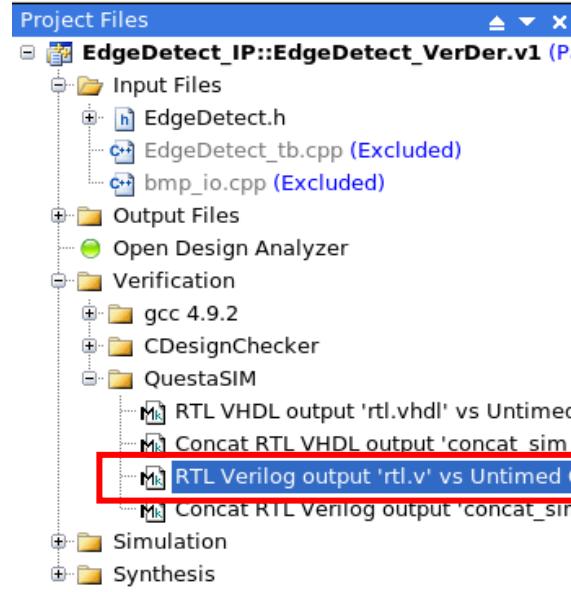
# Design Analyzer

Analyze the design in various perspectives such as Adviser, BOM, RTL vs C, ..., etc.

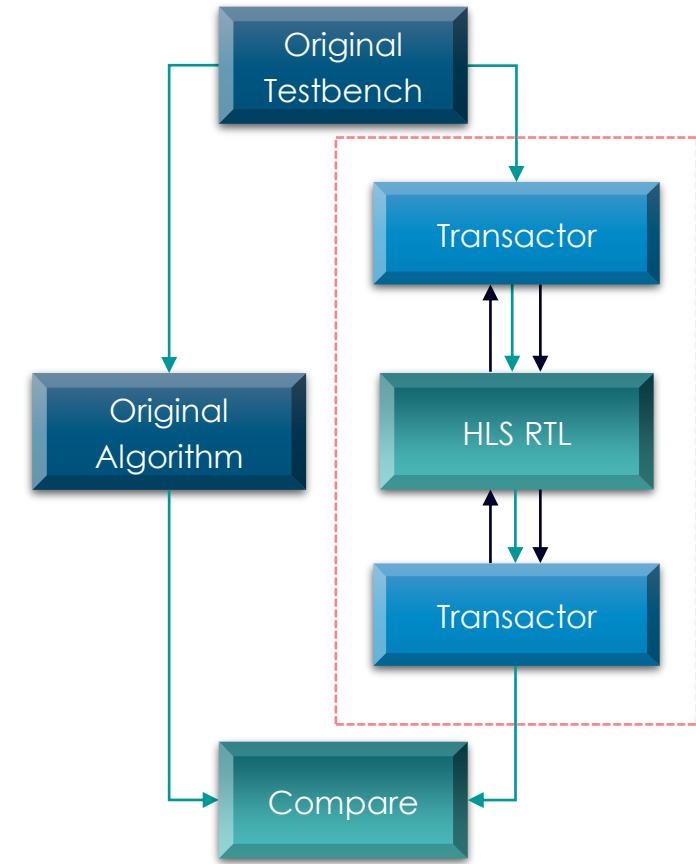
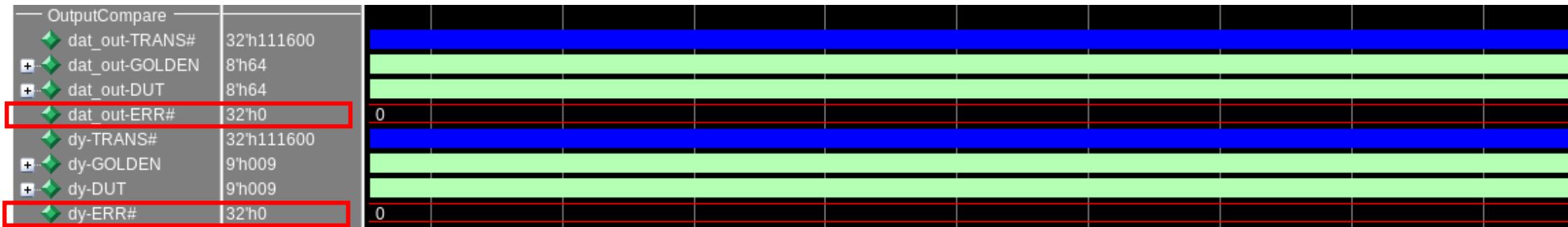


# RTL Verification (Cont.)

Automatically compare the outputs of HLC C and RTL



```
Transcript :  
# captured 1119744 values of dat_out  
# captured 1119744 values of dy  
# Info: scverify_top/user_tb: Simulation completed  
  
# Checking results  
# 'dat_out'  
# capture count      = 1119744  
# comparison count   = 1119744  
# ignore count       = 0  
# error count        = 0  
# stuck in dut fifo = 0  
# stuck in golden fifo = 0  
# 'dy'  
# capture count      = 1119744  
# comparison count   = 1119744  
# ignore count       = 0  
# error count        = 0  
# stuck in dut fifo = 0  
# stuck in golden fifo = 0  
  
# Info: scverify_top/user_tb: Simulation PASSED @ 11245016 ns  
# ** Note: (vsim-6574) SystemC simulation stopped by user.  
# 1
```



# Synthesize \*\_HorDer and \*\_MagAng by yourself

Task Bar

Synthesis Tasks

- Input Files
- Hierarchy**
- Libraries
- Mapping
- Architecture
- Resources
- Schedule
- RTL
- Power Report (Pre Power Opt)
- Power RTL

Start Page | Table | Flow Manager | Constraint Editor | Schedule | rtl.v | rtl.rpt |

Function: EdgeDetect\_IP::EdgeDetect

EdgeDetect.h

- EdgeDetect\_IP::EdgeDetect\_VerDer
- void EdgeDetect\_IP::EdgeDetect\_VerDer::EdgeDetect()
- void EdgeDetect\_IP::EdgeDetect\_VerDer::run(ac\_channel<ac\_int<32>> &dat, ac\_int<32> &dx, ac\_int<32> &dy)
- EdgeDetect\_IP::EdgeDetect\_HorDer**
- void EdgeDetect\_IP::EdgeDetect\_HorDer::EdgeDetect()
- void EdgeDetect\_IP::EdgeDetect\_HorDer::run(ac\_channel<ac\_int<32>> &dat, ac\_int<32> &dx, ac\_int<32> &dy)
- EdgeDetect\_IP::EdgeDetect\_MagAng
- void EdgeDetect\_IP::EdgeDetect\_MagAng::EdgeDetect()
- void EdgeDetect\_IP::EdgeDetect\_MagAng::run(ac\_channel<ac\_int<32>> &dat, ac\_int<32> &magn, ac\_int<32> &angle)
- EdgeDetect\_IP::EdgeDetect
- void EdgeDetect\_IP::EdgeDetect::EdgeDetect()
- void EdgeDetect\_IP::EdgeDetect::run(ac\_channel<ac\_int<32>> &dat, ac\_int<32> &dx, ac\_int<32> &dy)

Hierarchy Setting

- Top
- Block
- Inline
- Mapped
- Modular IO
- Blackbox

Design block attributes

Interface

Task Bar

Synthesis Tasks

- Input Files
- Hierarchy**
- Libraries
- Mapping
- Architecture
- Resources
- Schedule
- RTL
- Power Report (Pre Power Opt)
- Power RTL

Start Page | Table | Flow Manager | Constraint Editor | Schedule | rtl.v | rtl.rpt |

Function: EdgeDetect\_IP::EdgeDetect

EdgeDetect.h

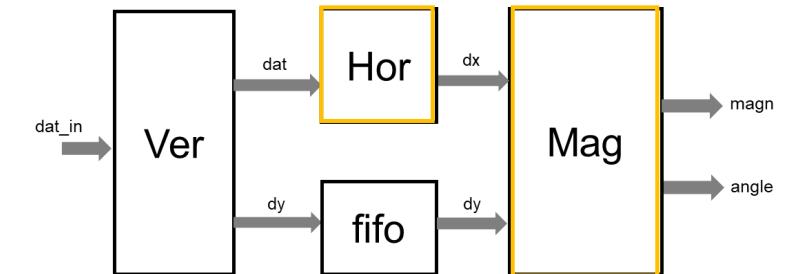
- EdgeDetect\_IP::EdgeDetect\_VerDer
- void EdgeDetect\_IP::EdgeDetect\_VerDer::EdgeDetect()
- void EdgeDetect\_IP::EdgeDetect\_VerDer::run(ac\_channel<ac\_int<32>> &dat, ac\_int<32> &dx, ac\_int<32> &dy)
- EdgeDetect\_IP::EdgeDetect\_HorDer**
- void EdgeDetect\_IP::EdgeDetect\_HorDer::EdgeDetect()
- void EdgeDetect\_IP::EdgeDetect\_HorDer::run(ac\_channel<ac\_int<32>> &dat, ac\_int<32> &dx, ac\_int<32> &dy)
- EdgeDetect\_IP::EdgeDetect\_MagAng**
- void EdgeDetect\_IP::EdgeDetect\_MagAng::EdgeDetect()
- void EdgeDetect\_IP::EdgeDetect\_MagAng::run(ac\_channel<ac\_int<32>> &dat, ac\_int<32> &magn, ac\_int<32> &angle)
- EdgeDetect\_IP::EdgeDetect
- void EdgeDetect\_IP::EdgeDetect::EdgeDetect()
- void EdgeDetect\_IP::EdgeDetect::run(ac\_channel<ac\_int<32>> &dat, ac\_int<32> &dx, ac\_int<32> &dy)

Hierarchy Setting

- Top
- Block
- Inline
- Mapped
- Modular IO
- Blackbox

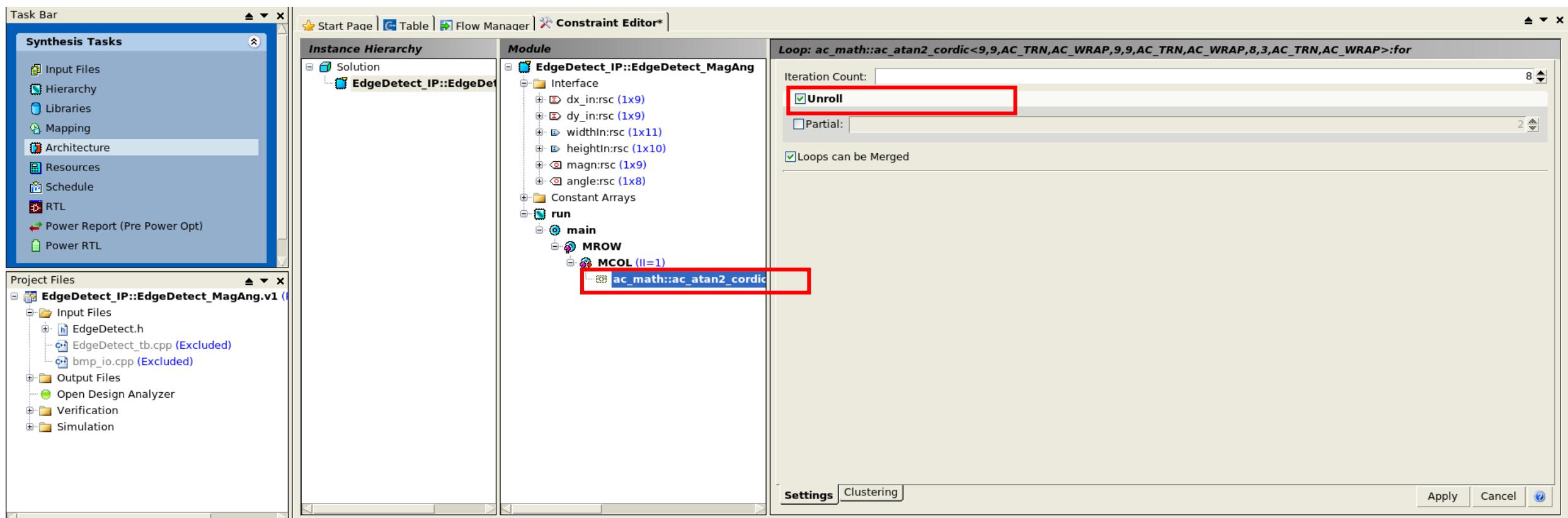
Design block attributes

Interface



# Architecture of \*\_MagAng

Unroll the for loop within MCOL  
MCOL is II1

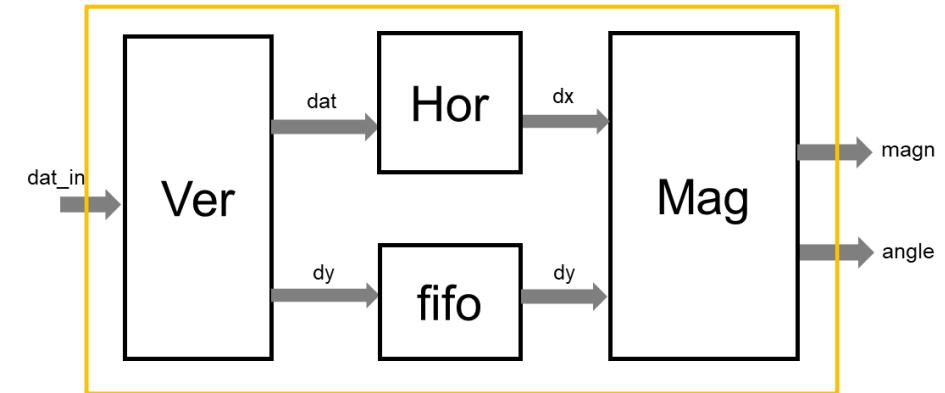


# Synthesize the \*\_Top

Now \*\_VerDer, \*\_HorDer, and \*\_MagAng have been synthesized and verified  
It's time to synthesize \*\_Top

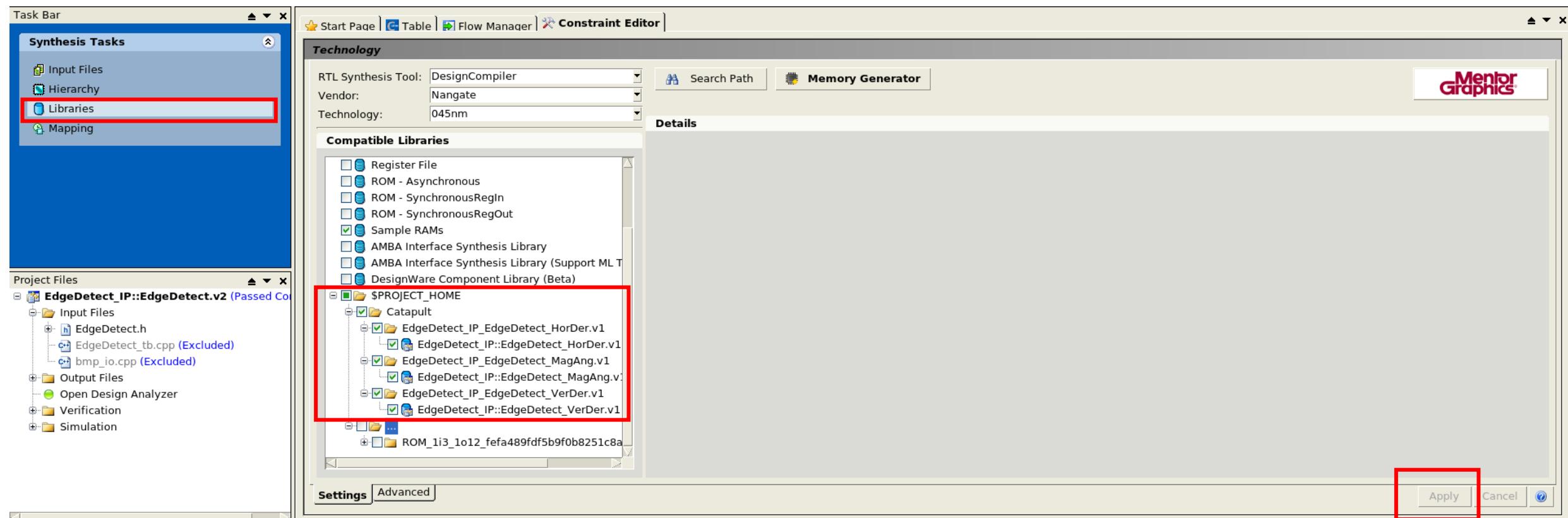
The screenshot shows the Synthesis Tasks interface. The left sidebar has a 'Hierarchy' item highlighted with a red box. The main area displays a tree view of files under 'EdgeDetect.h'. A right-click context menu is open over 'EdgeDetect\_IP::EdgeDetect\_Top'. The menu options are: Start Page, Table, Flow Manager, Constraint Editor, Design Check-Intermediate..., and a separator line followed by 'Function: EdgeDetect\_IP::EdgeDetect\_Top'. The 'Function' tab is selected. It shows 'Design block classification' with 'Hierarchy Type: function-based' and 'Scope Type: module'. Under 'Hierarchy Settings', 'Block implementation:' is set to 'Top' (radio button selected), which is also highlighted with a red box. Other options include Block, Inline, Mapped, Modular IO, and Blackbox. There are also 'Interface' and 'CCORE' buttons.

EdgeDetect\_Top



# Synthesize the \*\_Top (Cont.)

Add the libraries (\*\_VerDer, \*\_HorDer, and \*\_MagAng)



# Synthesize the \*\_Top (Cont.)

Mapping VerDer\_inst, HorDer\_inst, and MagAng\_inst to the corresponding blocks

The screenshot shows the Vivado interface with the following details:

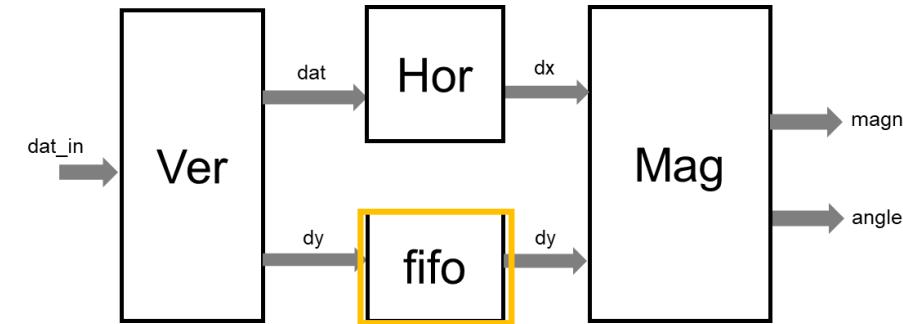
- Task Bar:** Shows "Synthesis Tasks" with "Mapping" selected (highlighted by a red box).
- Project Files:** Shows the project structure for "EdgeDetect\_IP::EdgeDetect.v2 (Passed Lib)".
- Instance Hierarchy:** Shows the hierarchy: Solution > EdgeDetect\_IP::EdgeDetect > VerDer\_inst, HorDer\_inst, MagAng\_inst. The VerDer\_inst node is highlighted by a red box.
- Module:** Displays the module configuration for "EdgeDetect\_IP::EdgeDetect\_VerDer".
  - Block Type:** [Block] EdgeDetect\_IP::EdgeDetect\_VerDer.v1
  - Timing Report:**
    - Latency Cycles: 3
    - Latency Time: 30.00
    - Throughput Cycles: 7
    - Throughput Time: 70.00
  - Area Report:**
    - Register: 760.76
    - Mux: 354.80
    - Functional: 192.69
    - Logic: 170.31
    - Memory: 0.00
    - ROM: 0.00
    - FSM: 23.53
    - Datapath: 1478.55

# Synthesize the \*\_Top (Cont.)

Decide the FIFO depth according to the latency of \*\_Ver and

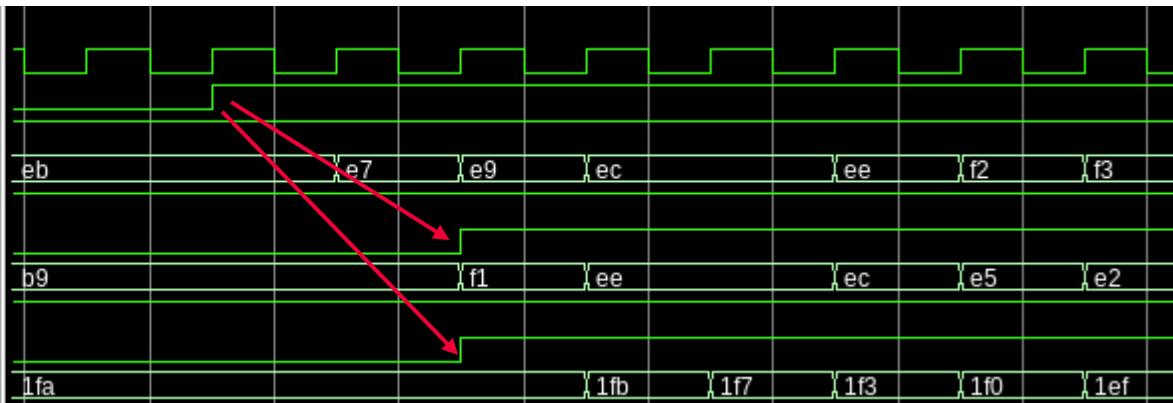
\*\_Hor

- dat\_out : 2T
- dy\_out : 2T
- dx\_out : 2T



Ver

DUT	
clk	1'h1
dat_in_rsc_rdy	1'h0
dat_in_rsc_vld	1'h1
+ dat_in_rsc_dat	8'h64
dat_out_rsc_rdy	1'h1
dat_out_rsc_vld	1'h0
+ dat_out_rsc_dat	8'h64
dy_rsc_rdy	1'h1
dy_rsc_vld	1'h0
+ dy_rsc_dat	9'h009



Hor

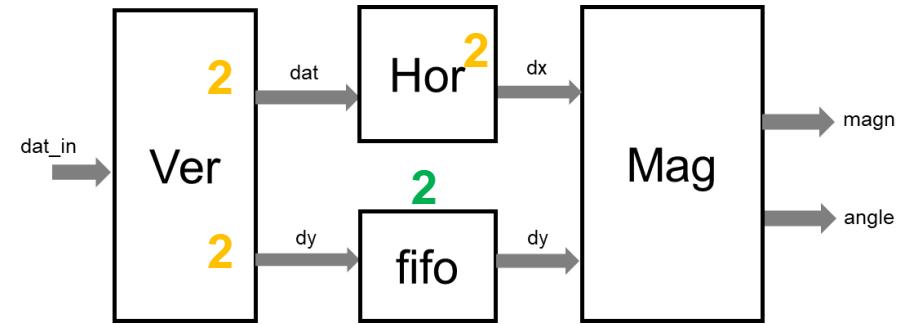
DUT	
clk	1'h1
dat_in_rsc_rdy	1'h0
dat_in_rsc_vld	1'h1
+ dat_in_rsc_dat	8'h64
dx_rsc_rdy	1'h1
dx_rsc_vld	1'h0
+ dx_rsc_dat	9'h1fe



# Synthesize the \*\_Top (Cont.)

FIFO depth for each channel between blocks

- dat\_chn: 0
- dx\_chn: 0
- dy\_chn:  $2 + 2 - 2 = 2$



Synthesis Tasks

Architecture

Resource: dat:cns

Resource Type: ccs\_ioprt.ccs\_pipe

Datasheet

FIFO Depth:

Input Delay

Library Delay: 0 ns

Inherited Delay: 0 ns

Synthesis Tasks

Architecture

Resource: dy:cns

Resource Type: ccs\_ioprt.ccs\_pipe

Datasheet

FIFO Depth: 2

Input Delay

Library Delay: 0 ns

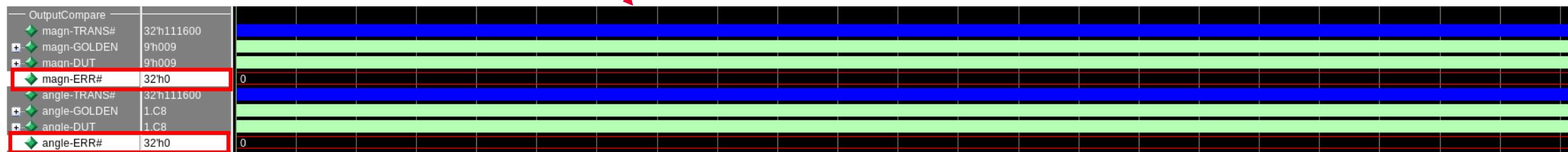
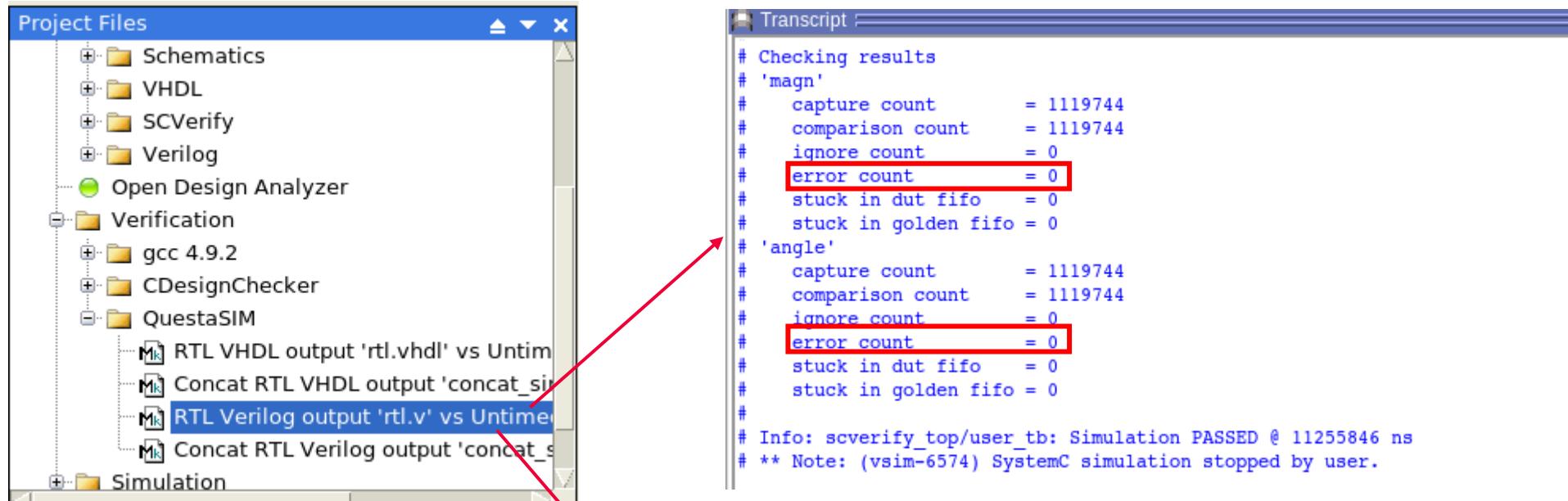
Inherited Delay: 0 ns

# Synthesize the \*\_Top (Cont.)

The screenshot shows the Vivado IDE interface with the following components:

- Task Bar:** Shows "Start Page", "Table", "Flow Manager", "Constraint Editor", "Design Check-Intermedia...", "Design Check.rpt", and "rtl.v".
- Synthesis Tasks:** A sidebar with options: Input Files, Hierarchy, Libraries, Mapping, Architecture, Resources, Schedule, and **RTL** (which is highlighted with a red box).
- Project Files:** A tree view showing Output Files (Reports, Schematics), Verilog (rtl.v, concat\_rtl.v (extract), concat\_sim\_rtl.v (extract)), SCVerify, Open Design Analyzer, Verification (gcc 10.3.0), and Original Design + Testbench. **rtl.v** is also highlighted with a red box.
- Code Editor:** Displays the Verilog code for the `EdgeDetect_IP_EdgeDetect_Top` module. The code defines the module with various input and output ports, including `clk`, `rst`, `arst_n`, and various data and control signals for line buffers and magnification.
- Report Table:** A separate window showing a report table titled "Report: General". The table has columns for Solution, Latency..., Throughput..., Slack, and Total Area. It lists several entries, with the last entry being `EdgeDetect_IP::EdgeDetect.v1 (extract)` which has a latency of 0, throughput of 0, slack of -1.08, and total area of 9833.78.

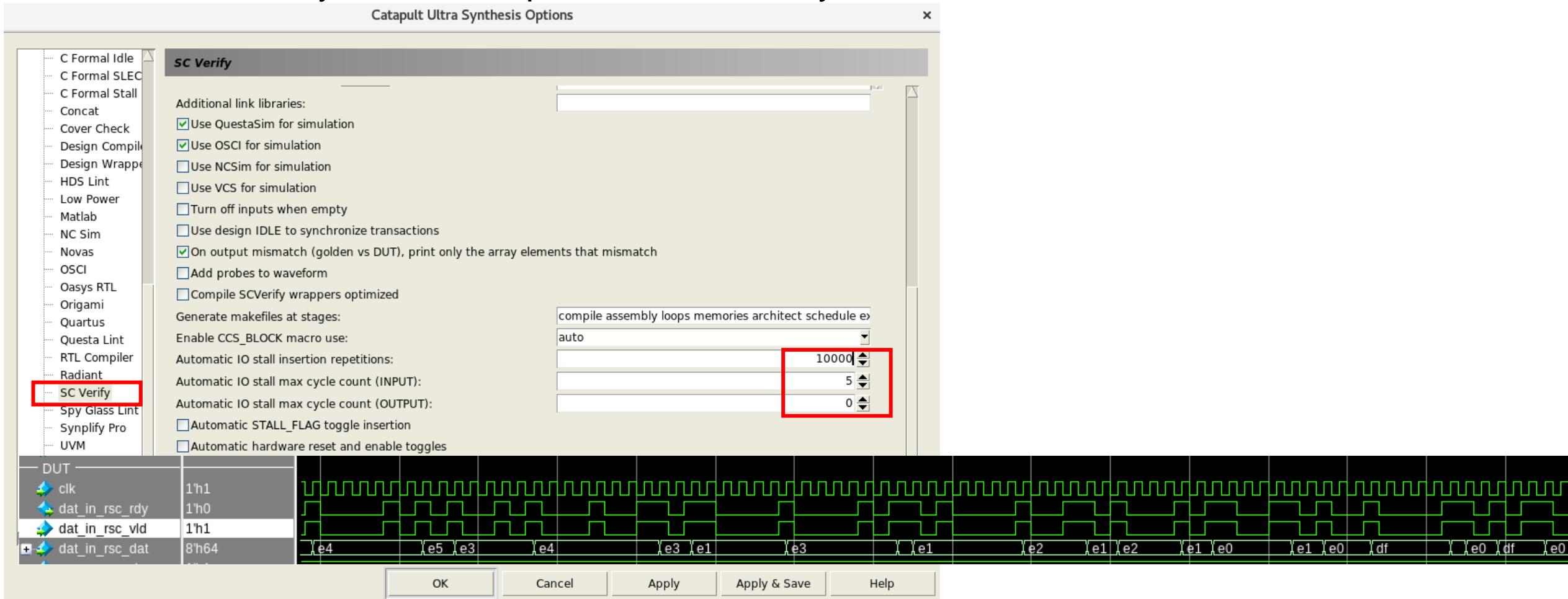
# RTL Verification for \*\_Top



# RTL Verification for \*\_Top (Cont.)

Controlling stalling behavior in the testbench

- Stall dat\_in, totally 10000 samples, for random cycles between 0 and 5



# Wrap Up

## Build c++ source

- make

## Run c++ simulation

- cd bin; ./run.sh

## Run HLS synthesis

- cd catapult\_work; catapult -f directives.tcl &

```
drwxr-xr-x. 3 mentor mentor 107 Dec 12 15:02 bin
drwxr-xr-x. 4 mentor mentor 28 Dec 12 14:42 bmpUtil
drwxr-xr-x. 7 mentor mentor 4096 Dec 12 15:34 catapult_work
drwxr-xr-x. 3 mentor mentor 17 Dec 12 14:42 cmodel
drwxr-xr-x. 4 mentor mentor 28 Dec 12 14:42 hls_c
-rwxr-X---. 1 mentor mentor 860 Dec 12 14:42 Makefile
-rw-r--r--. 1 mentor mentor 379 Dec 12 14:42 readme.txt
[mentor@RHEL74 01_edgedetect]$ █
```

# | EdgeDetect IP for FSIC Project

02\_edgedetect\_fsic

# Specification

Max. resolution: 640x480 (VGA) Monochrome

Max. throughput: 4 pixels per cycle

The port definition of EdgeDetect IP is in the right table

Port Name	In / Out	Width	Function
clk	in	1	Posedge clock
rst	in	1	synchrouse reset; high active
rst_n	in	1	asynchronous reset; low active
widthIn	in	10	image width
heightIn	in	9	image height
sw_in	in	1	output image switching; 1: edge magnitute, 0: input image
dat_in_rsc_dat	in	34	image input; [0:31]: four 8bit pixels, [32]: is first pixel in a frame, [33]: is last pixel in an image line
dat_in_rsc_vld	in	1	image input valid
dat_in_rsc_rdy	out	1	IP is ready to accept an image input
dat_out_rsc_dat	out	34	image output; [0:31]: four 8bit pixels, [32]: is first pixel in a frame, [33]: is last pixel in an image line
dat_out_rsc_vld	out	1	image output valid
dat_out_rsc_rdy	in	1	Outside environment is ready to accept an image output
crc32_pix_in_rsc_dat	out	32	crc32 code for an input image
crc32_pix_in_triosy_lz	out	1	crc32 code is valid
crc32_dat_out_rsc_dat	out	32	crc32 code for an output image
crc32_dat_out_triosy_lz	out	1	crc32 code is valid
line_buf0_rsc_en	out	1	ram0 clock enable
line_buf0_rsc_q	in	64	ram0 read data
line_buf0_rsc_we	out	1	ram0 write enable
line_buf0_rsc_d	out	64	ram0 write data
line_buf0_rsc_adr	out	7	ram0 rd/wr address
line_buf1_rsc_en	out	1	ram1 clock enable
line_buf1_rsc_q	in	64	ram1 read data
line_buf1_rsc_we	out	1	ram1 write enable
line_buf1_rsc_d	out	64	ram1 write data
line_buf1_rsc_adr	out	7	ram1 rd/wr address

# Few Modifications of Original EdgeDetect

Process four pixels per clock cycle

Use sum of absolute difference (SAD) for edge magnitude calculation

Add two crc32 calculation on image input / output

Select the output source from input image or the calculated magnitude

# Wrap Up

Enter the directory 02\_edgedetect\_fsic

Build c++ source

- make

Run c++ simulation

- cd bin; ./run.sh
- .hex files are dumped used for RTL verification being integrated with FSIC

Run HLS synthesis

- cd catapult\_work; catapult -f directives.tcl &
- top-down flow is used
- concat\_EdgeDetect\_Top.v will be generated in Catapult/EdgeDetect\_Top.v1

```
drwxr-xr-x. 4 mentor mentor 213 Dec 12 14:04 bin
drwxr-xr-x. 4 mentor mentor 28 Dec 12 14:04 bmpUtil
drwxr-xr-x. 7 mentor mentor 4096 Dec 12 14:04 catapult_work
drwxr-xr-x. 3 mentor mentor 17 Dec 12 14:04 cmodel
drwxr-xr-x. 4 mentor mentor 28 Dec 12 16:28 hls_c
-rwxr-x--. 1 mentor mentor 860 Dec 12 14:04 Makefile
-rw-r--r--. 1 mentor mentor 288 Dec 12 14:04 readme.txt
[mentor@RHEL74 02_edgedetect_fsic]$
```

```
rw-rw-r--. 1 mentor mentor 9 Dec 12 17:33 crc32_in_img.hex
rw-rw-r--. 1 mentor mentor 9 Dec 12 17:33 crc32_out_img.hex
rwxr-xr-x. 2 mentor mentor 80 Dec 12 14:04 image
rw-rw-r--. 1 mentor mentor 691200 Dec 12 17:33 in_img.hex
rw-rw-r--. 1 mentor mentor 691254 Dec 12 17:33 out_algorithm.bmp
rw-rw-r--. 1 mentor mentor 691254 Dec 12 17:33 out_hw.bmp
rw-rw-r--. 1 mentor mentor 691200 Dec 12 17:33 out_img.hex
rwxr-xr-x. 1 mentor mentor 175 Dec 12 14:04 run.sh
rwxrwxr-x. 1 mentor mentor 841440 Dec 12 17:32 SCVerify_EdgeDetect.exe
rwxrwxr-x. 2 mentor mentor 49 Dec 12 14:04 tmp
mentor@RHEL74 bin]$
```

# | RTL Integration and Verification

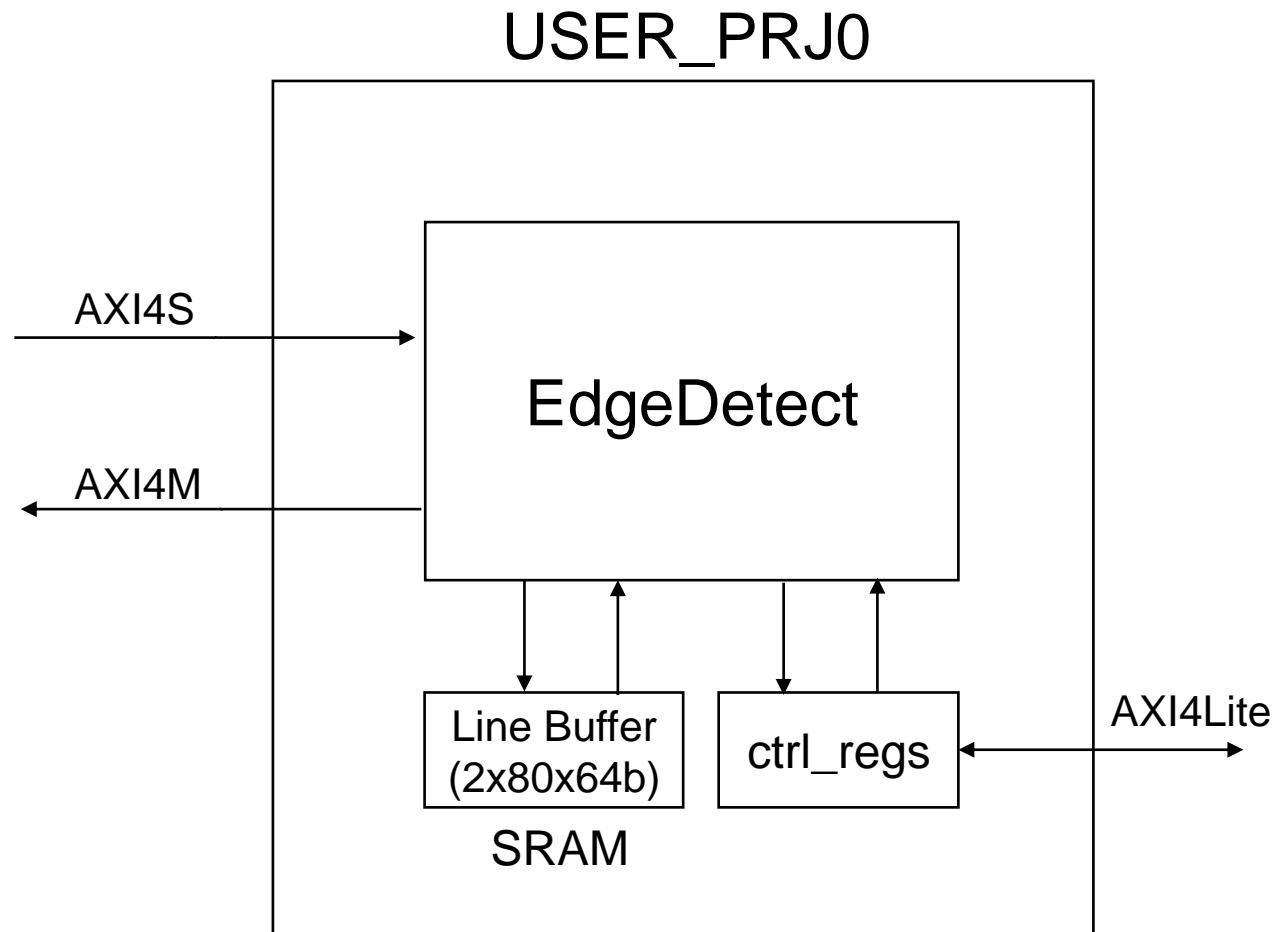
03\_fsic\_prj

# RTL Integration with FSIC

Copy concat\_EdgeDetect\_Top.v to be as concat\_EdgeDetect\_Top\_fsic.v in 03\_fsic\_prj/dsn/rtl

Instantiate EdgeDetect in USER\_PRJ0 in 03\_fsic\_prj/dsn/rtl/user\_prj0.v

Also, add ctrl\_regs rd/wr circuits and two SRAMs



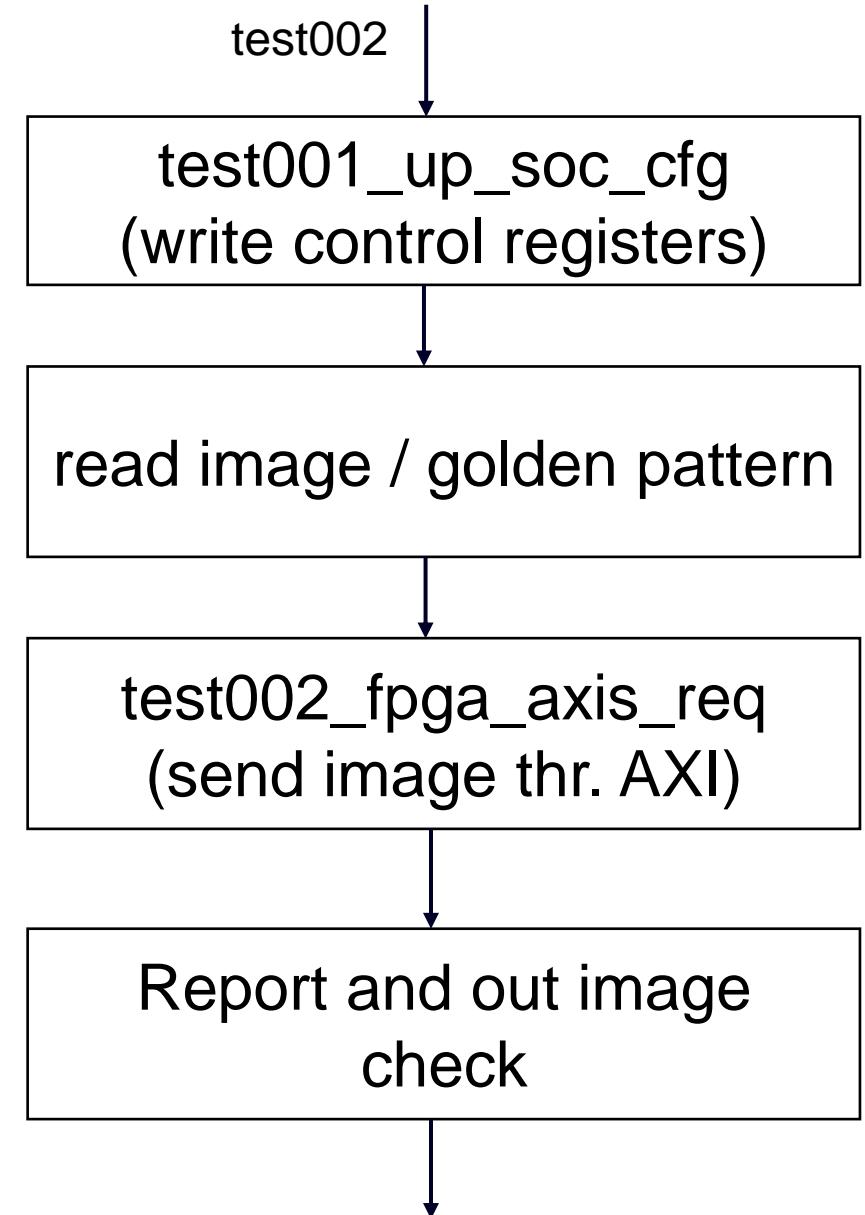
# FSIC Testbench

Modify some functions in FSIC TB (tb\_fsic.v) for verifying the EdgeDetect IP

- test001\_up\_soc\_cfg: control register read / write test
- test002, test002\_fpga\_axis\_req, fpga\_axis\_req: send / receive input / output image through AXI

Image and golden pattern are put in dsn/testbench/vsim/pattern which is generated in 02\_edgedetect\_fsic

Search the keyword 'USE\_EDGEDETECT\_IP' for detail



# Register Map

```
//widthIn  
cfg_read_data_expect_value = TST_FRAME_WIDTH;  
soc_up_cfg_write('h4, 4'b0111, cfg_read_data_expect_value);  
soc_up_cfg_read('h4, 4'b0111);
```

Register	WordOffset	StartBit	Length	ResetVal	Access	HADDRS	description
reg_RST	0	0	1	0	R/W	0	1: reset edgedetect IP
reg_widthIn	1	0	10	640	R/W	4	frame width
reg_heightIn	2	0	9	480	R/W	8	frame height
reg_sw_in	3	0	1	1	R/W	12	1: output is edge map; 0: output is src image
reg_CRC32_stream_in	4	0	32	0	R-O	16	crc32 code for stream in data
reg_CRC32_stream_out	5	0	32	0	R-O	20	crc32 code for stream out data
reg_edgedetect_done	6	0	1	0	R/W	24	Read it for checking if a frame is done. Write it for clearing this signal.

# Wrap Up

Put the edgedetect rtl into dsn/rtl and \*.hex test pattern into dsn/testbench/vsim/pattern

Modify the dsn/rtl/user\_prj0.v and dsn/testbench/tb\_fsic.v

Add concat\_EdgeDetect\_Top\_fsic.v in dsn/testbench/vsim/filelist

Run rtl simulation with the script run\_vsim

```
[mentor@RHEL74 vsim]$ ./run_vsim  
|VSIM 1> run -all|
```

```
=====  
=====  
===== 4753925=> Final result [PASS], check_cnt = 115301, error_cnt = 0  
=====  
=====  
===== ** Note: $finish : ..../tb_fsic.v(437)  
Time: 4753925 ns Iteration: 0 Instance: /tb_fsic  
=====
```

Check if there is any errors during simulation

Trace or debug rtl using Visualizer

- vis -designfile design.bin -wavefile qwave.db &