Step-By-Step Lab2: EdgeDetect

Yuan-Teng Chang, Catapult HLS AE
yuan-teng.chang@siemens.com

SIEMENS

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

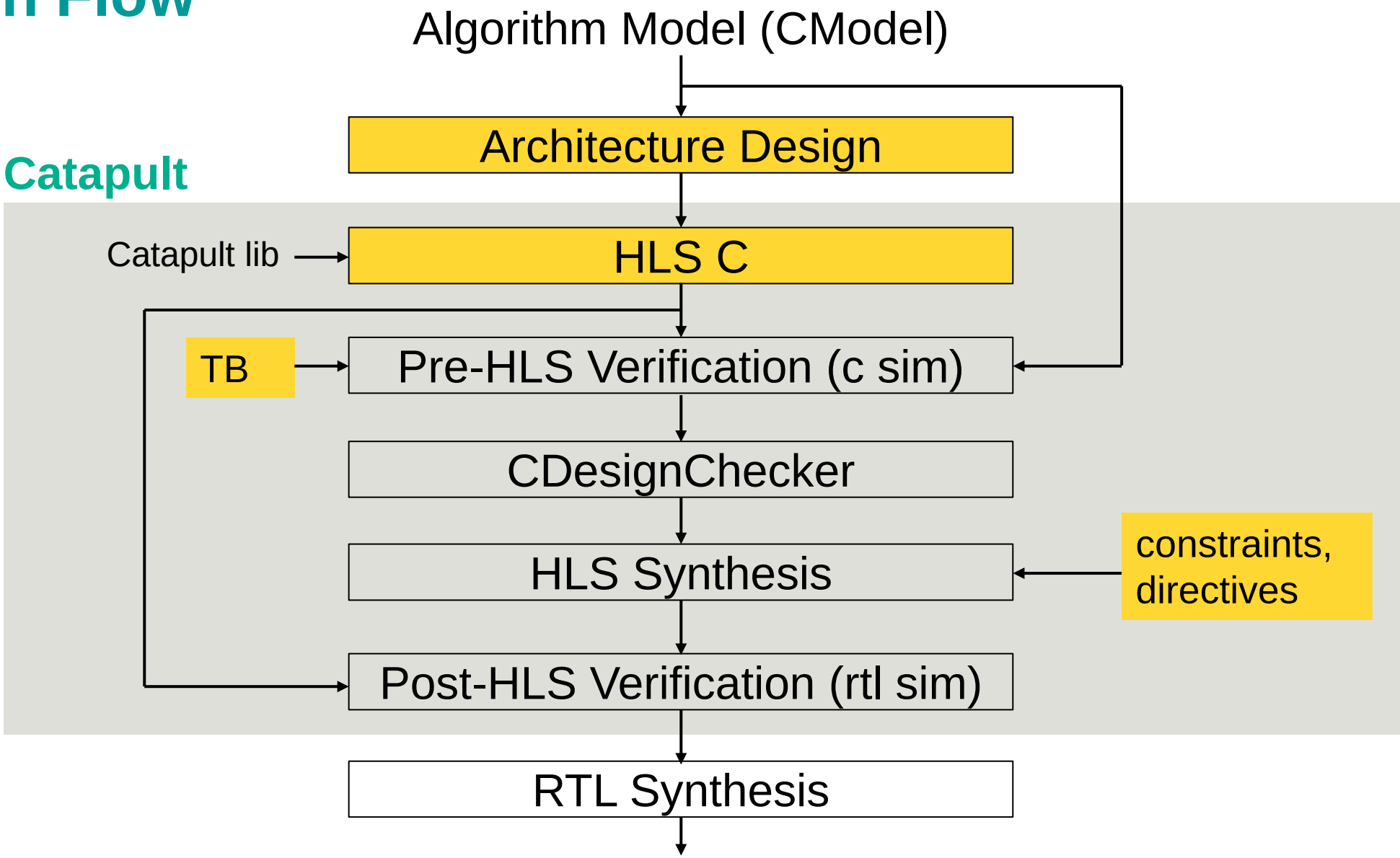
Environment Variables

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

SIEMENS

An Edge Detection Algorithm

Used in many image and video processing algorithms

- People detection, texture matching, etc.

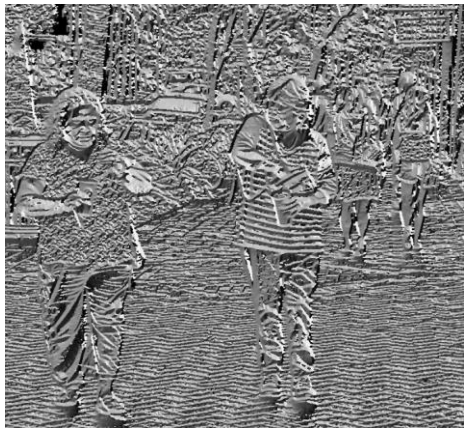
Computing the image gradient detects the edges in an image

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

Image

Gradient Magnitude

Gradient Angle



SIEMENS

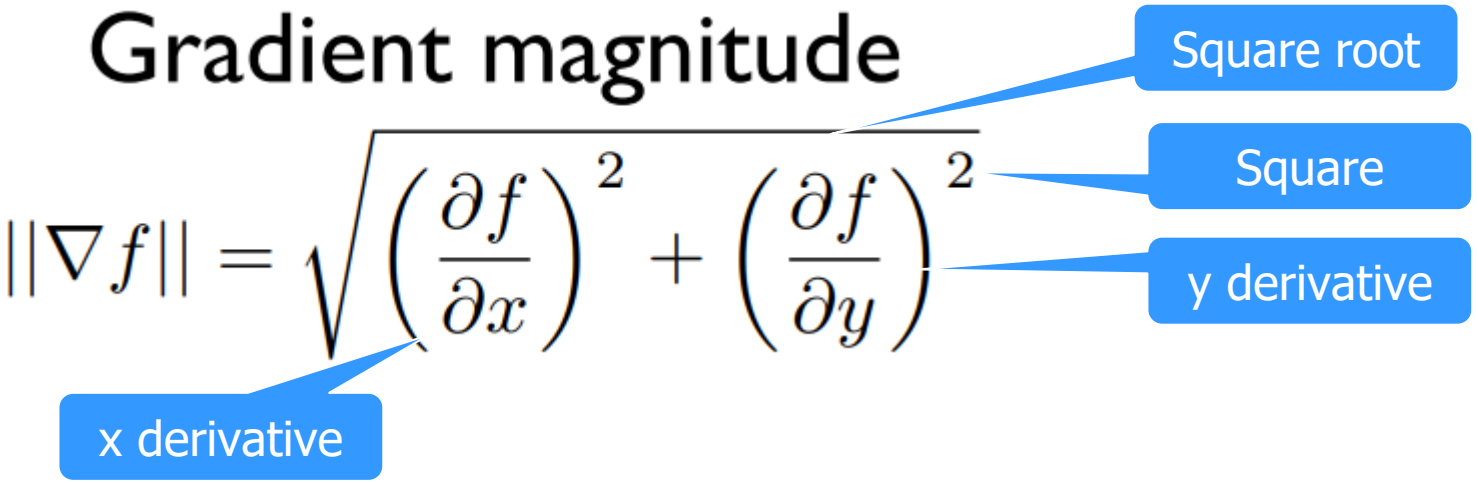
Gradient Magnitude

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

- Computationally simple
- Requires memory architecture for efficient hardware

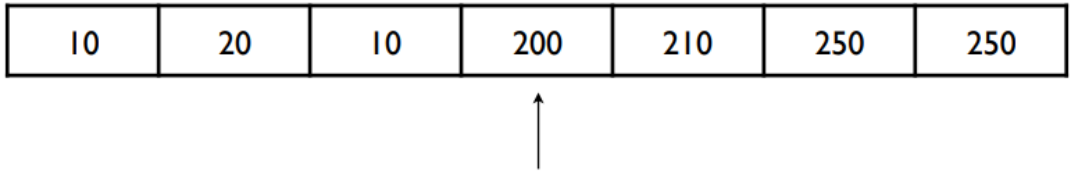
Take the square root of the sum of squares of the derivatives

- Computationally complex

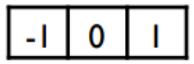


Computing 1-D derivative

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



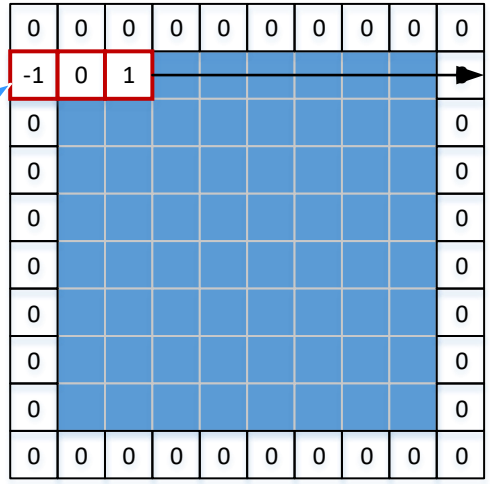
$$f'(x) = \frac{f(x+1) - f(x-1)}{2} = \frac{210 - 10}{2} = 100$$



1x3 kernel

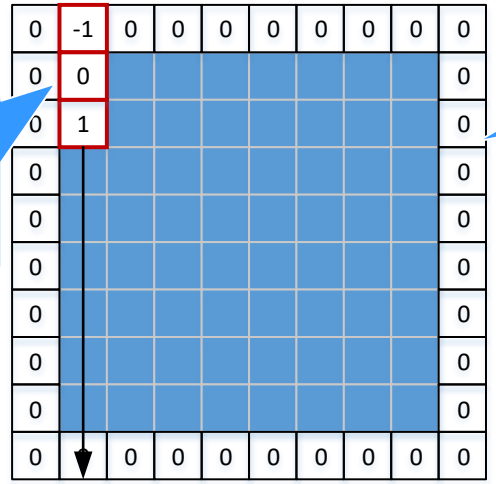
1D derivative filter

Horizontal(x) derivative



Kernel slides "horizontally" across the image

Vertical(y) derivative



Kernel slides "vertically" across the image

Image boundary is zero padded, replicated, etc

SIEMENS

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);
  }
  ...
};
```


SIEMENS

C++ Algorithm Vertical Derivative

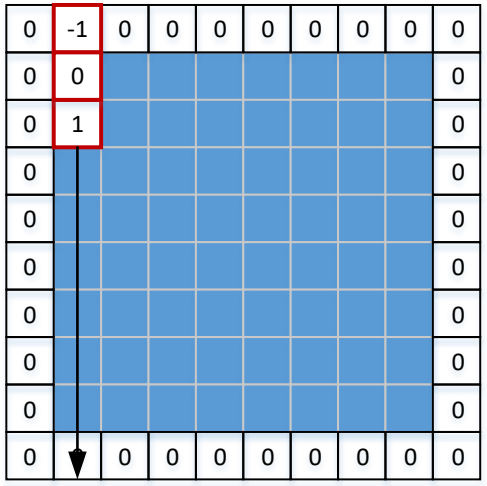
Vertical derivative reads across three different lines of image data (y-1, y, y+1)

Clip function prevents the index from reading out of bounds

- Boundary pixels are replicated

Horizontal derivative is similar

```
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];
    }
  }
}
```


SIEMENS

C++ Algorithm Magnitude Angle

Squares the derivatives

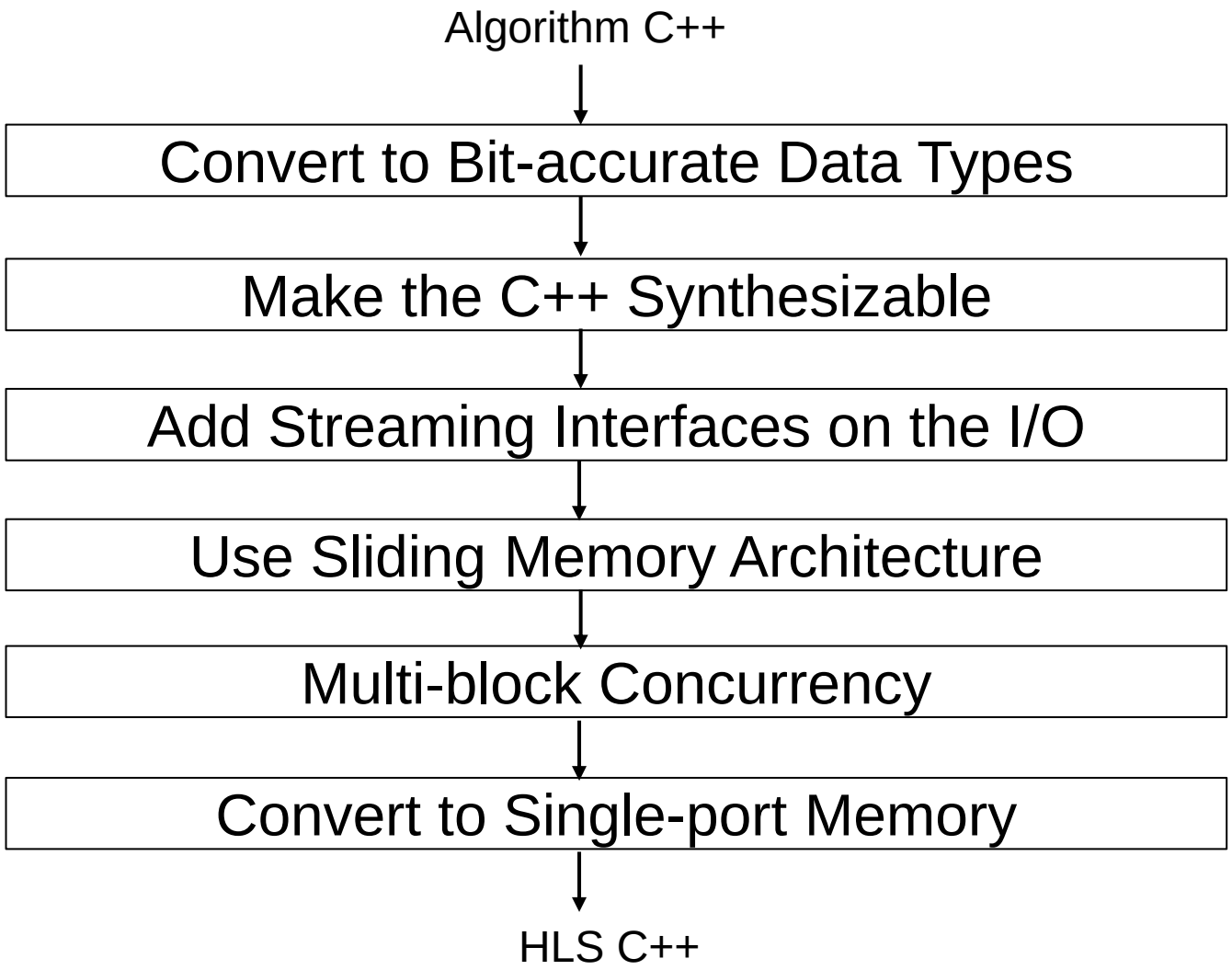
Performs square root (math.h)

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]);
    }
  }
}
```


SIEMENS

Converting Algorithm C++ to HLS C++



SIEMENS

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;
```


SIEMENS

Overall Design Goals

~1 pixel/clock throughput

- Image is 1296x864 = ~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

SIEMENS

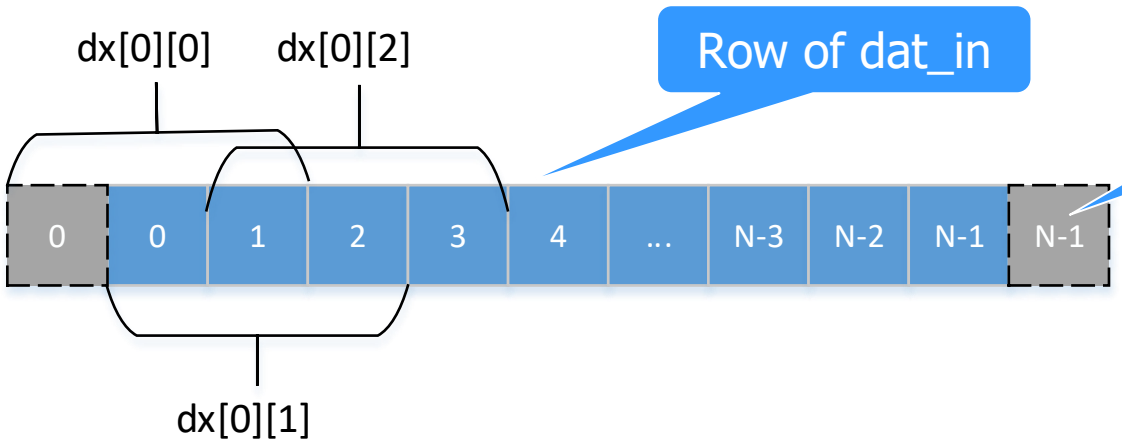
Understanding the Required Memory Architecture

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];
    }
  }
}
```



Replicated boundary pixel

Previous 2 values read from dat_in are reused for each new computation. Data is shifting (Sliding window)

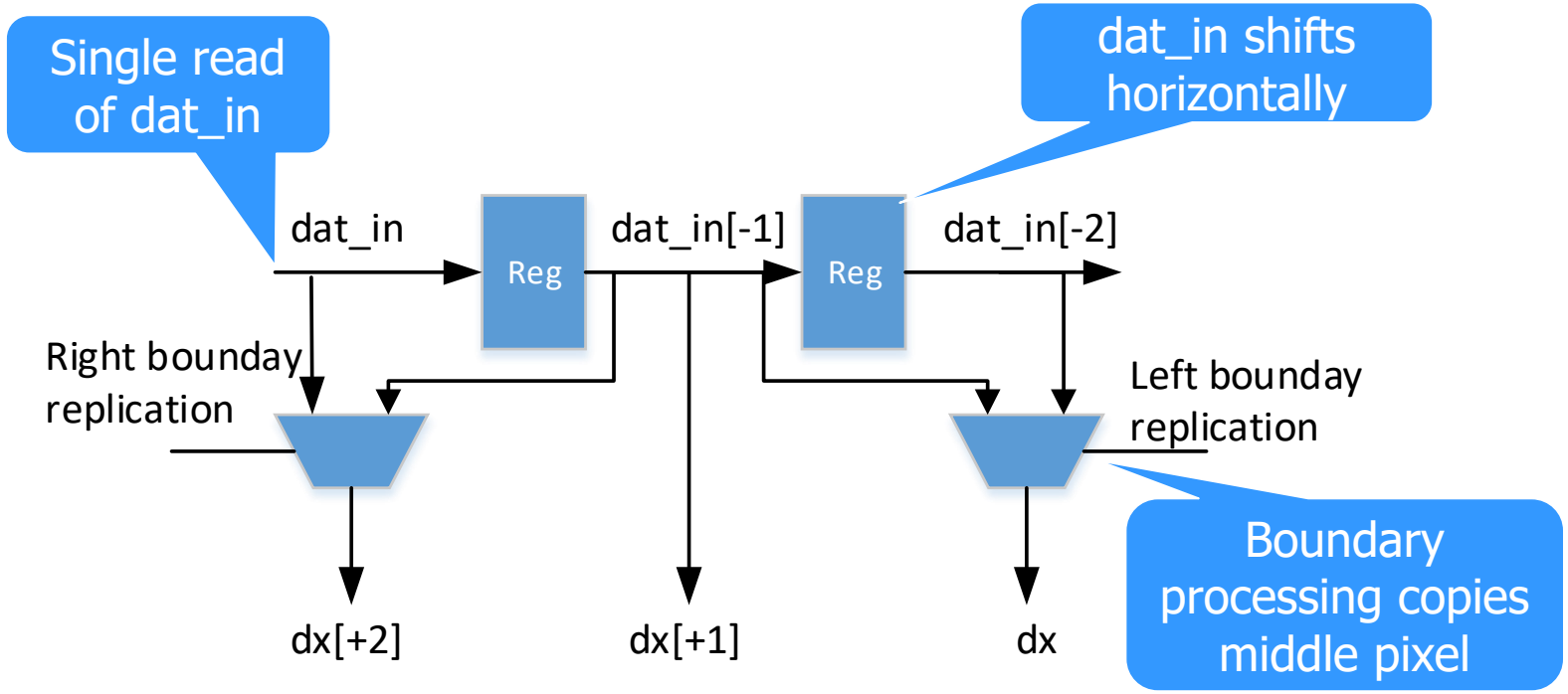
dx[0][0] = dat_in[0][0] * kernel[0] + dat_in[0][0] * kernel[1] + dat_in[0][1] * kernel[2]
dx[0][1] = dat_in[0][0] * kernel[0] + dat_in[0][1] * kernel[1] + data_in[0][2] * kernel[2]
dx[0][2] = dat_in[0][1] * kernel[0] + data_in[0][2] * kernel12] + data_in[0][3] * kernel[2]

SIEMENS

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 II=1



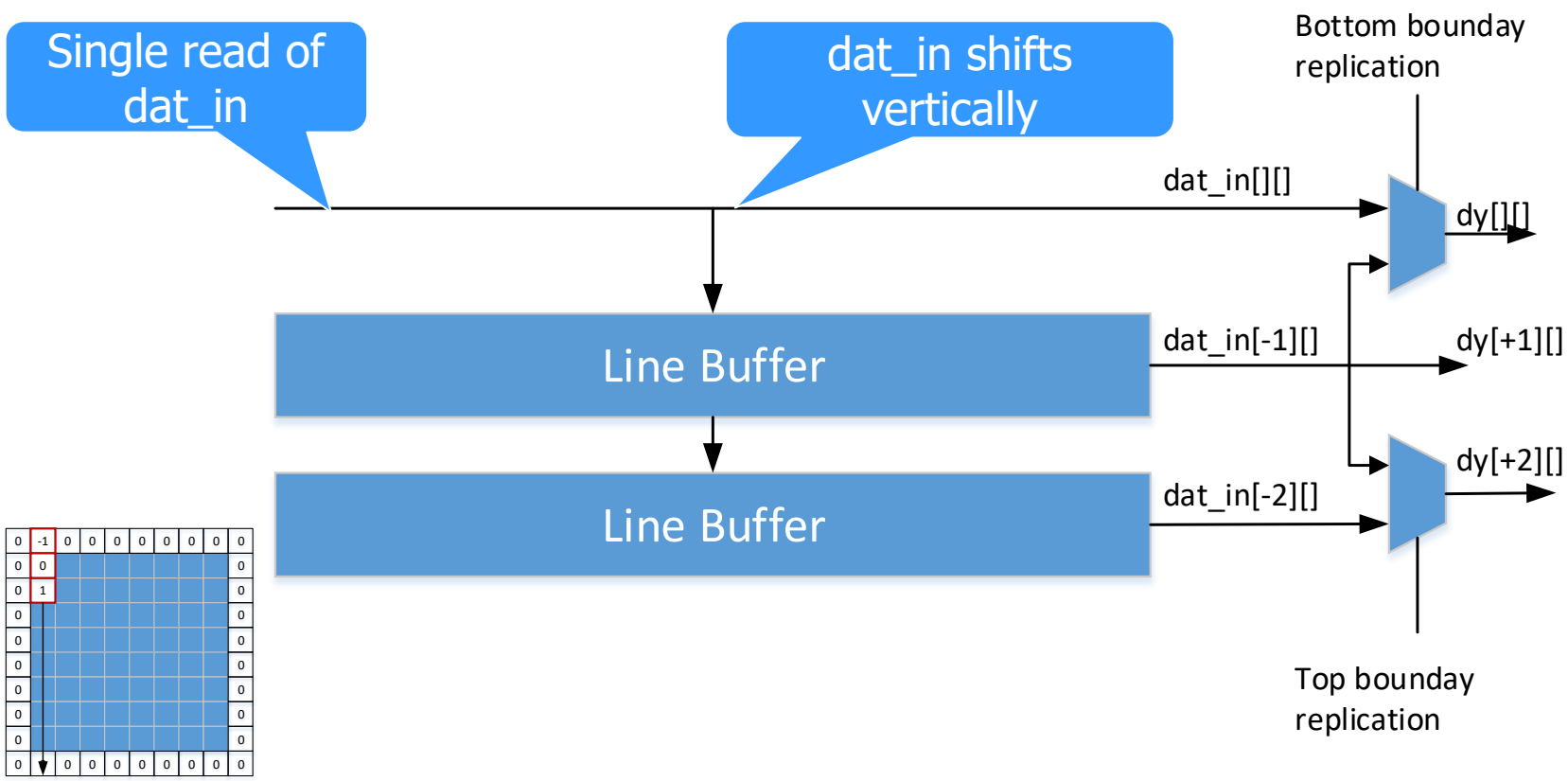

```
void horizontalDerivative(pixelType dat_in[imageHeight][imageWidth],
                          gradType  dx[imageHeight][imageWidth])
{
  // pixel buffers store pixel history
  pixelType pix_buf0;
  pixelType pix_buf1;

  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];
      }
```


SIEMENS

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];
      }
    }
  }
}
```


SIEMENS

Understanding Memory Dependency Failures

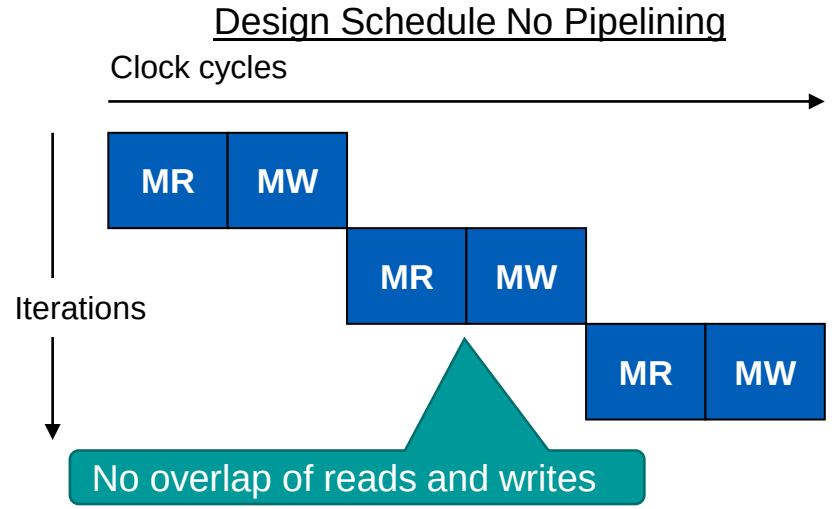
Catapult will schedule memory reads and writes to the same memory in different cycles if simultaneous read/write to the same address is not allowed

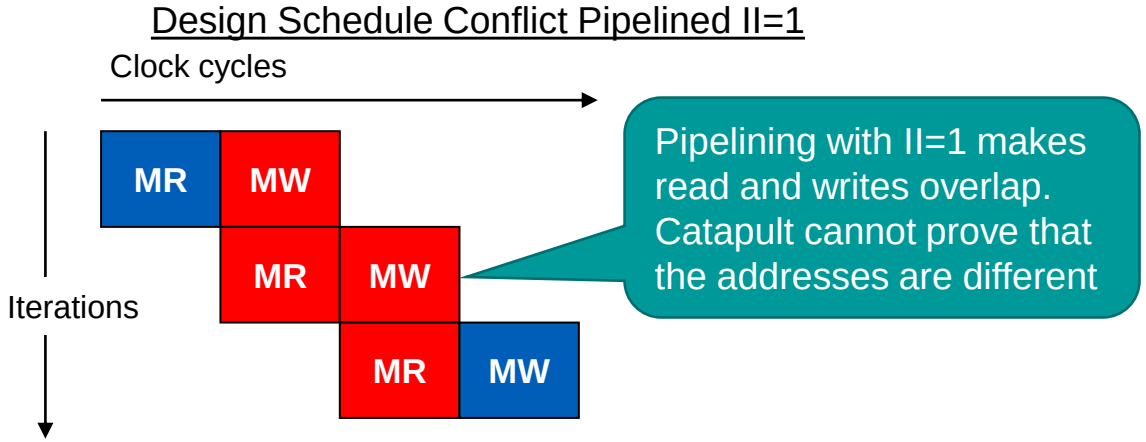
- Pipelining the design will make the reads and writes appear to overlap

```
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 whi
    }
    line_buf1[x] = pix1; // copy previous line
    line_buf0[x] = pix0; // store current line
```

Read mem

Write mem





SIEMENS

Ignoring False Memory Dependencies

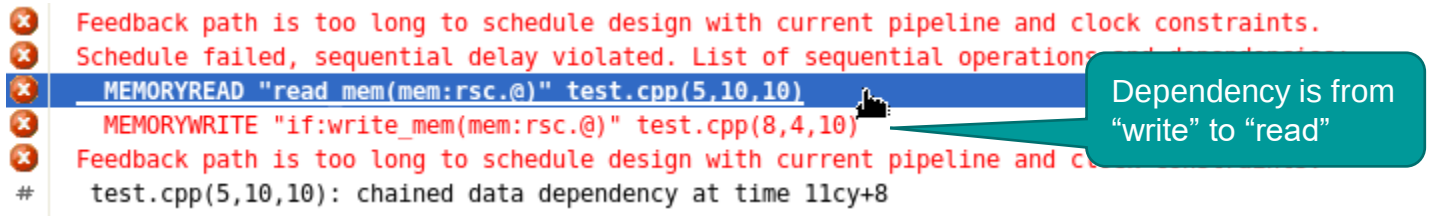
The scheduler can be told to ignore the memory dependency if user knows there can't be address conflict

- 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>

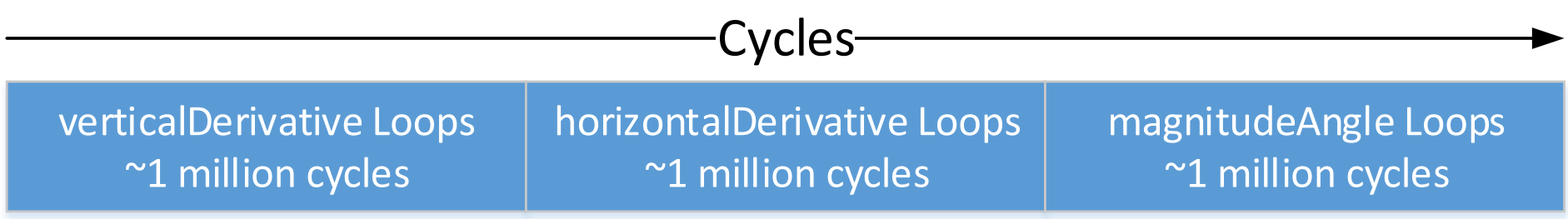



```
go architect
ignore_memory_precedences -from if:write_mem(mem:rsc.@) -to read_mem(mem:rsc.@)
```

Sequential Loops Execute Sequentially in Hardware

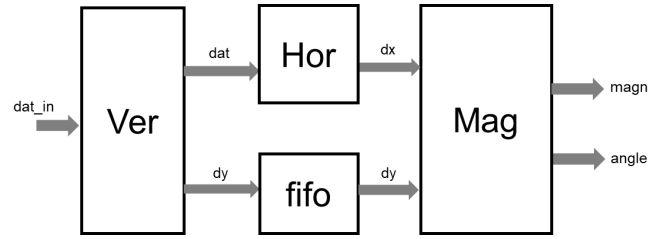
Sequential loops in the same design block/partition will execute sequentially in hardware

```
#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);
  }
```



SIEMENS

Adding Hierarchy in Catapult



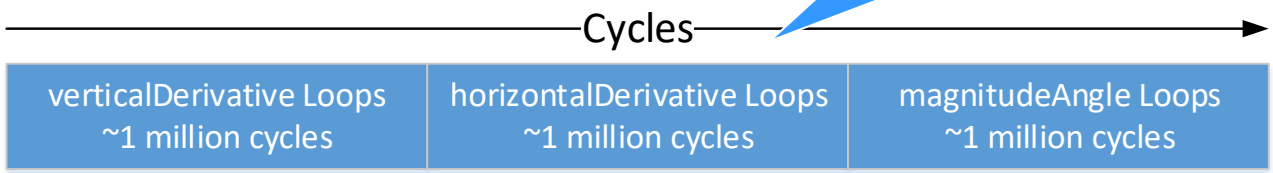
Catapult can synthesize classes/functions as concurrent processes using design constraints

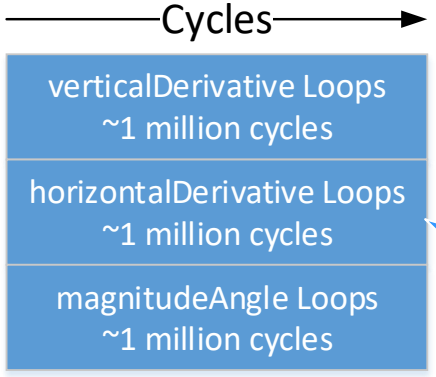
- ac_channel interconnect required
- Allows sequential loops to run concurrently

ac_channel interconnect

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

public:
  EdgeDetect_Hierarchy() {}

  //------------------------------------------------------
  // Function: run
  //   Top interface for data in/out of class. Combines ve
  //   horizontal derivative and magnitude/angle computati
#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)
```

Sequential loops





Functions/loops run concurrently

#pragma to make member function a concurrent process

SIEMENS

Dual Port Memory is Expensive for ASIC

FPGA memory hard macros are dual port

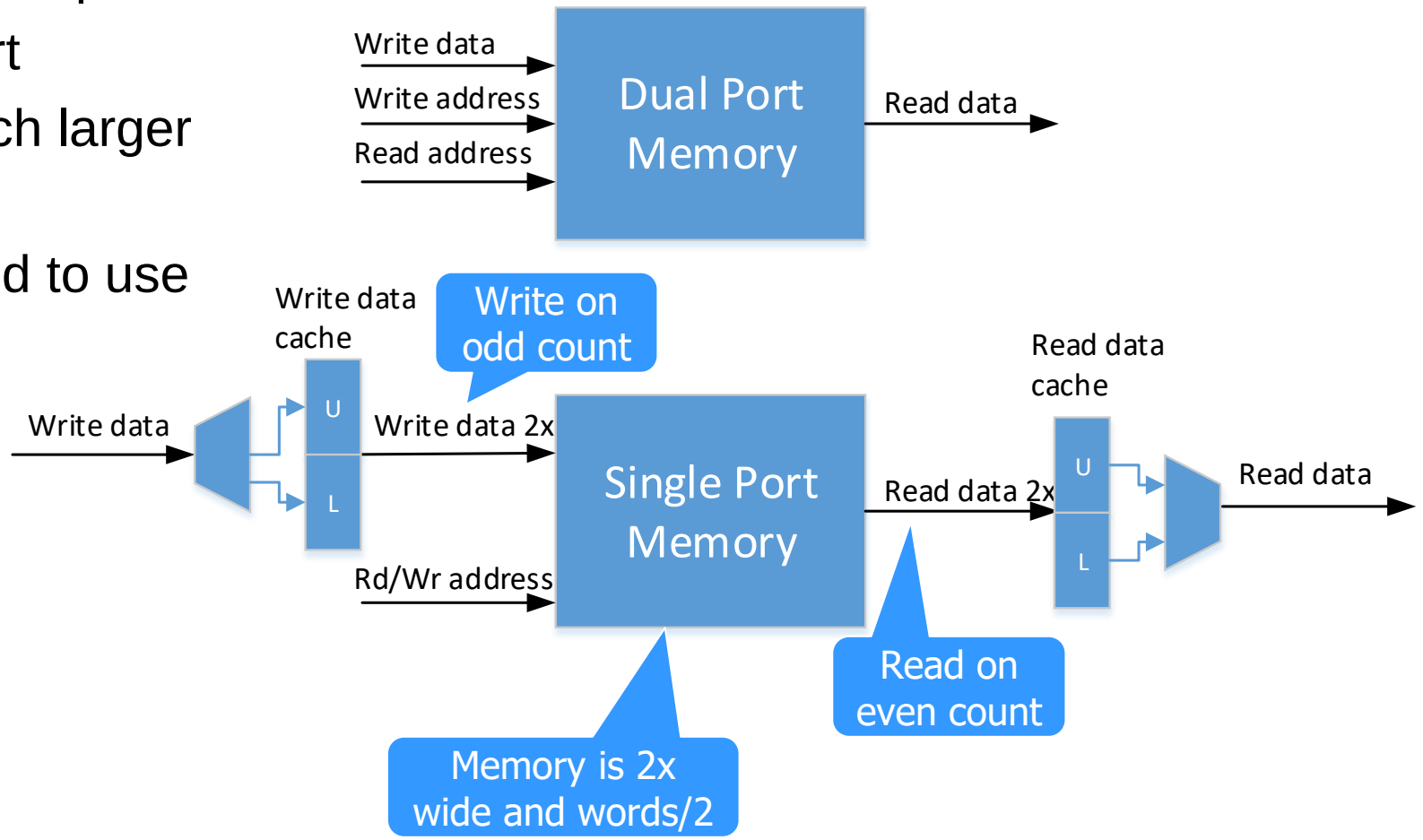
- Cost is the same as a single port

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 c
  line_buf1[x/2] = rdbuf0_pix;
  line_buf0[x/2] = wrbuf0_pix;
}
```



SIEMENS

Catapult Flow – HLS & HLV

01_edgedetect/cmodel – Algorithm C++

01_edgedetect/hls_c – HLS C++ design and TB

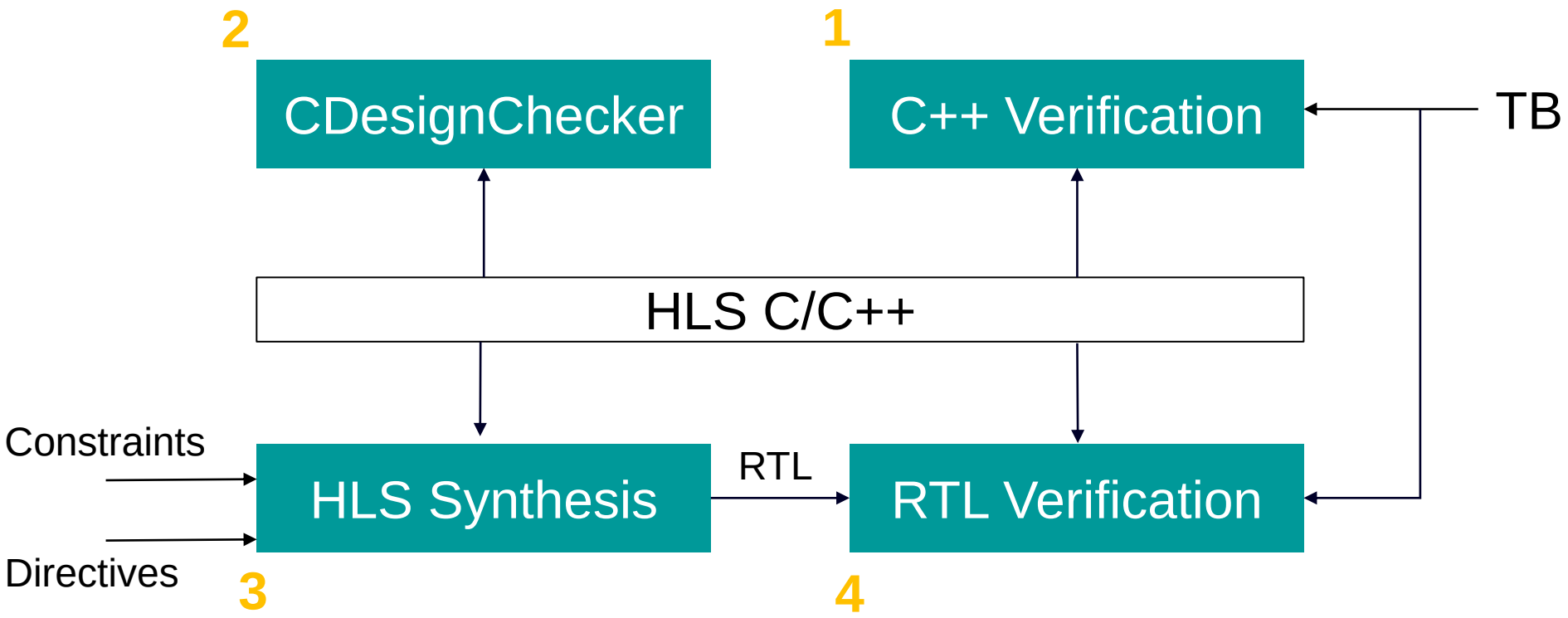
01_edgedetect/bin – c sim directory

01_edgedetect/catapult_work – Catapult working directory

01_edgedetect/Makefile – c sim Makefile

SIEMENS

Catapult Flow – HLS & HLV



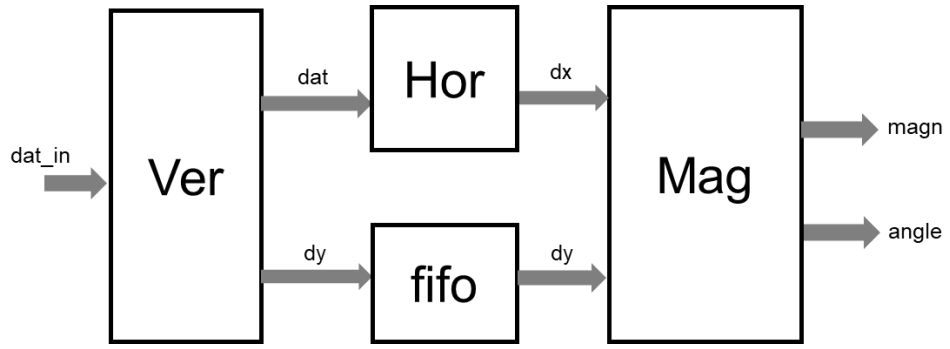
HLS C++ (EdgeDetect_defs.h)

```
#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>

//#define 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 magnitute 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;
}
```


SIEMENS

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)
    {...}
  };
}
```

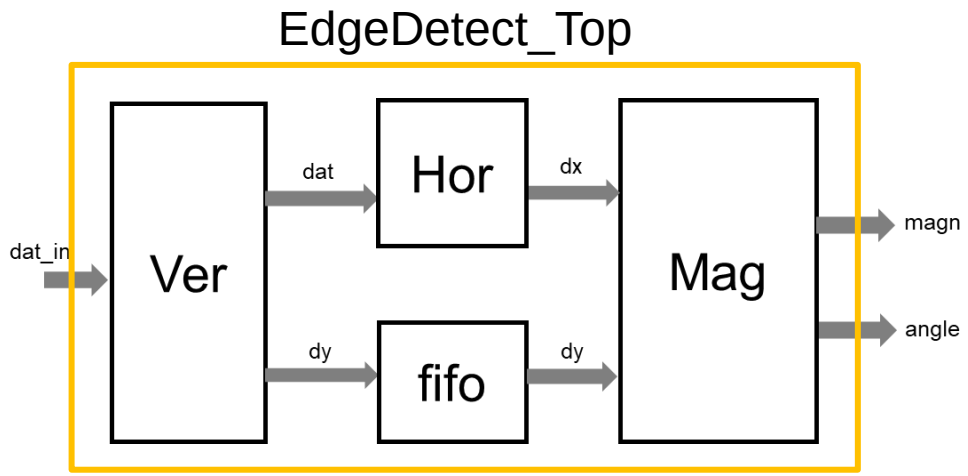

SIEMENS

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 inpu

  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);
    }
  };
}
```

DirectInput

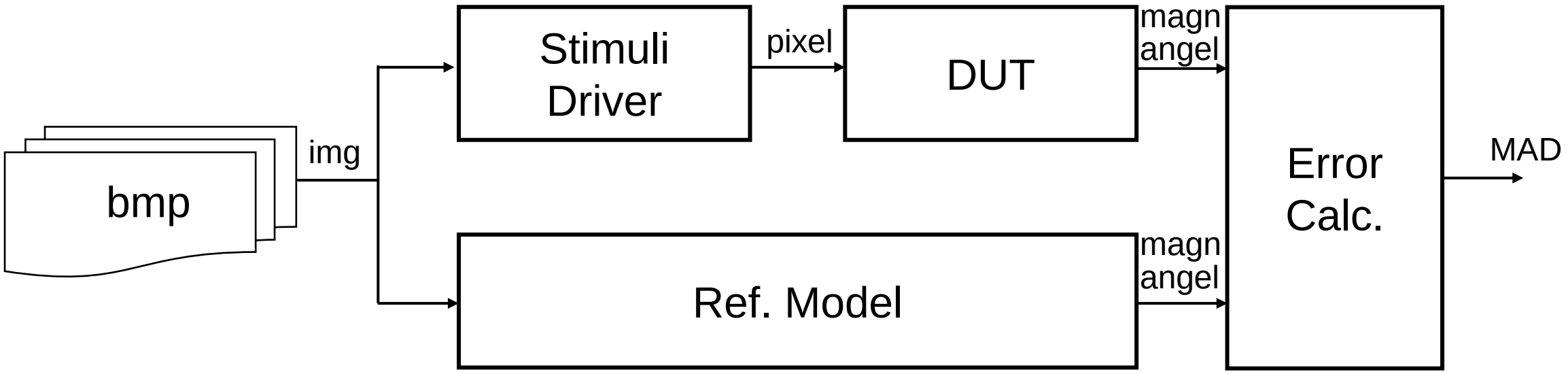
- Design blocks are connected with ac_channel
- Glue logic is not allowed, only connected in the Top module



SIEMENS

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 monochro
      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

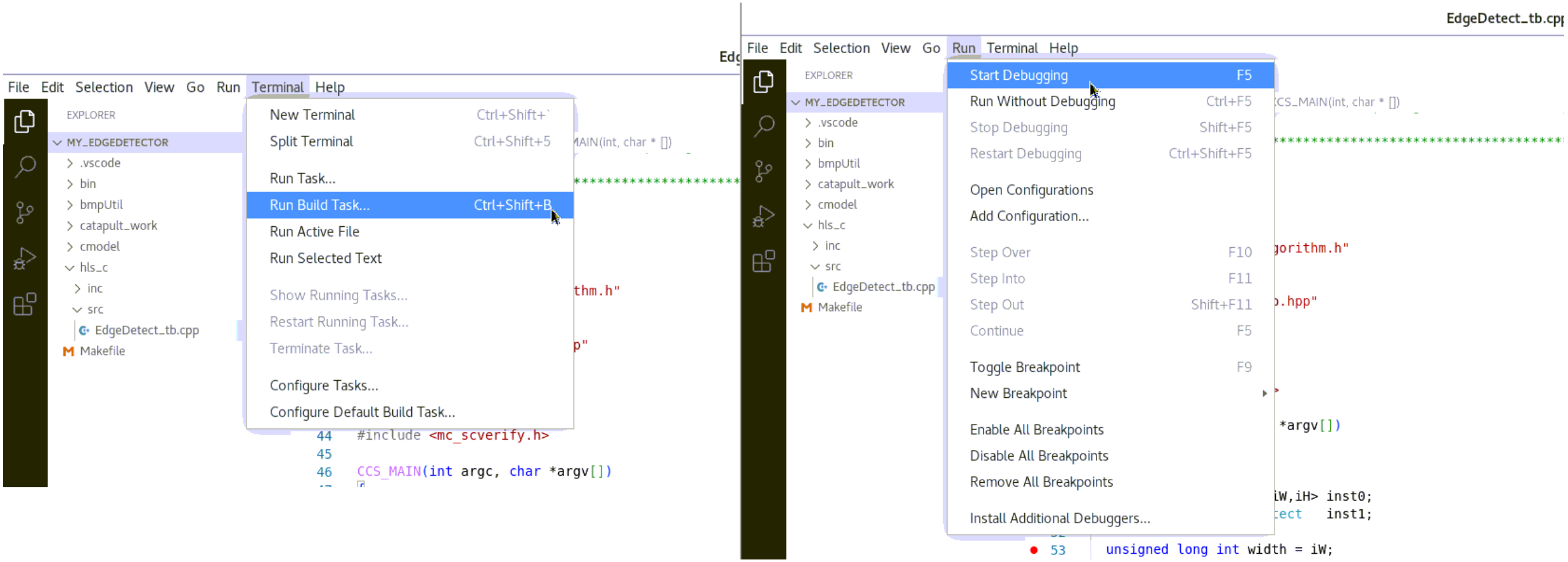
SIEMENS

VSCode Environment

A friendly IDE for source code editor, debugger, …

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

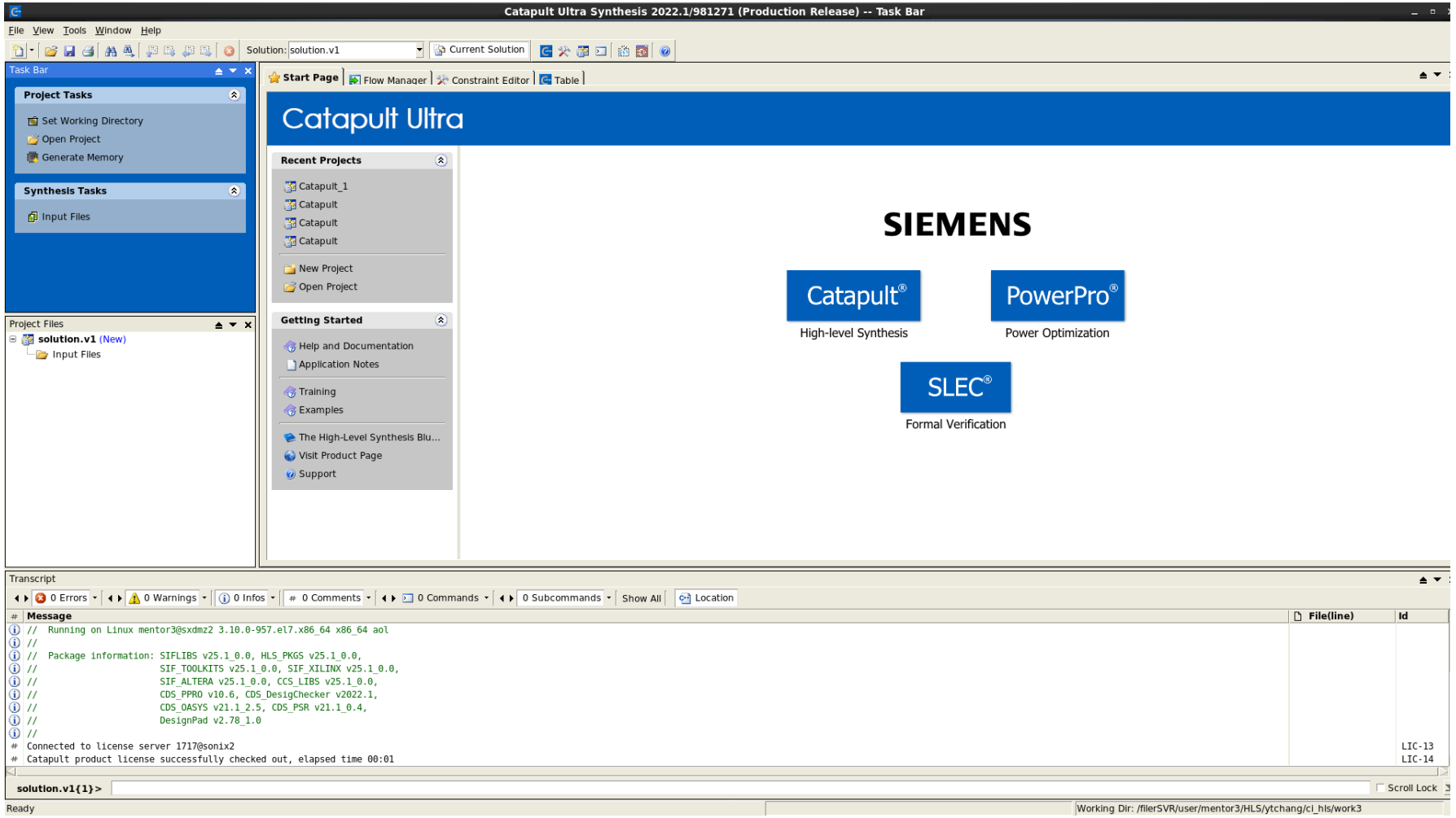
```
hang@a4sa46 My EdgeDetector]$ code . &
```



SIEMENS

Launch Catapult

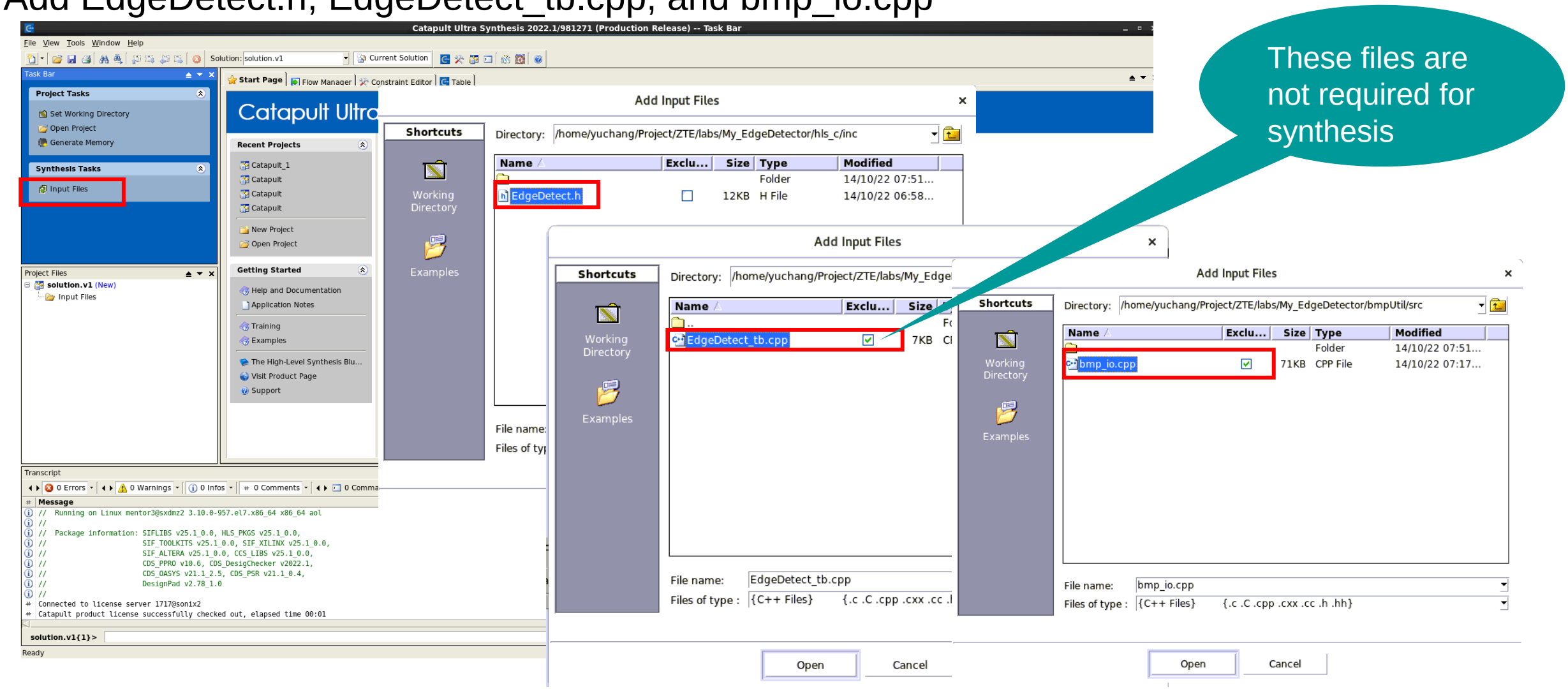
catapult &



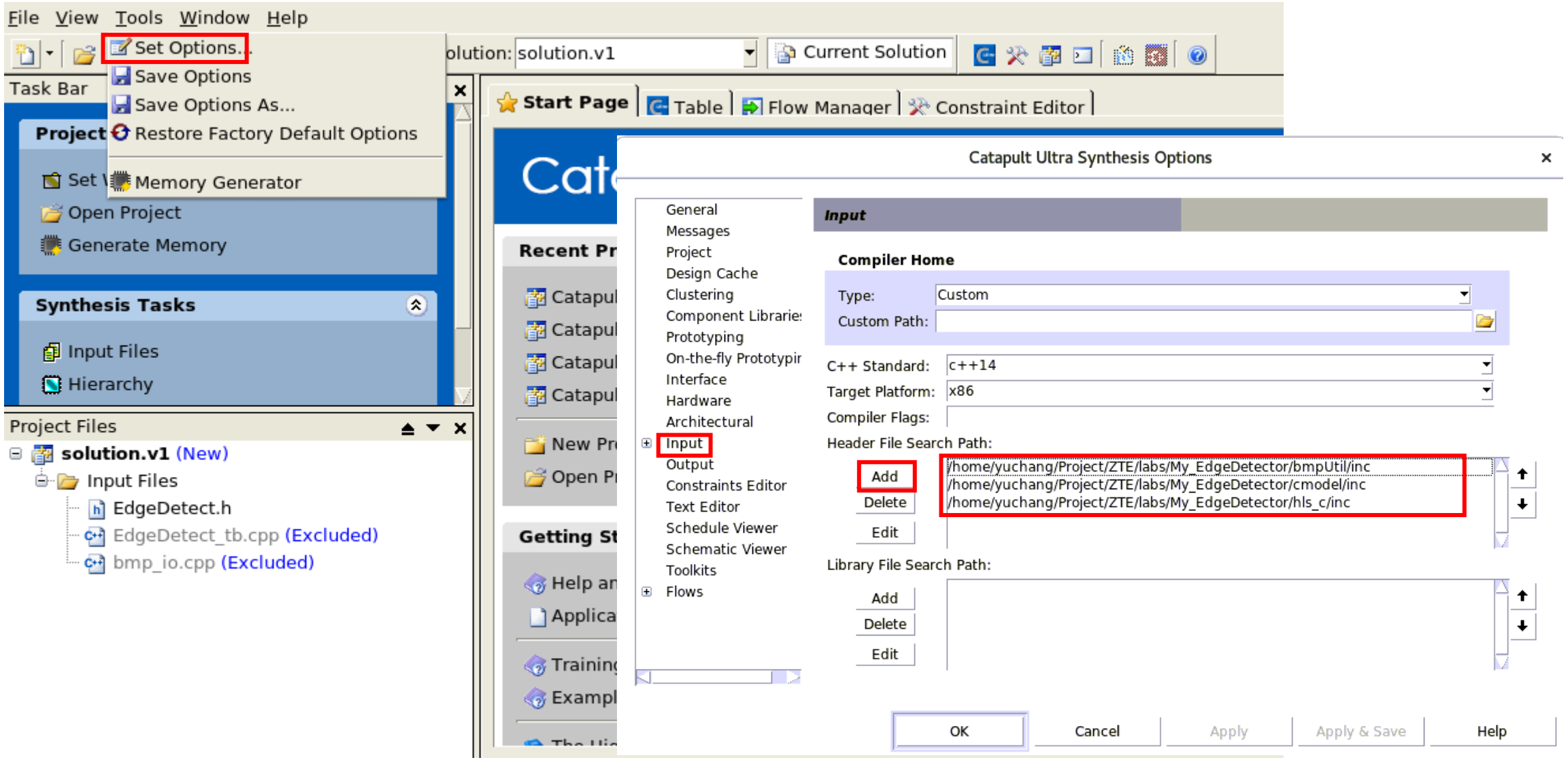
SIEMENS

Add input file

Add EdgeDetect.h, EdgeDetect_tb.cpp, and bmp_io.cpp



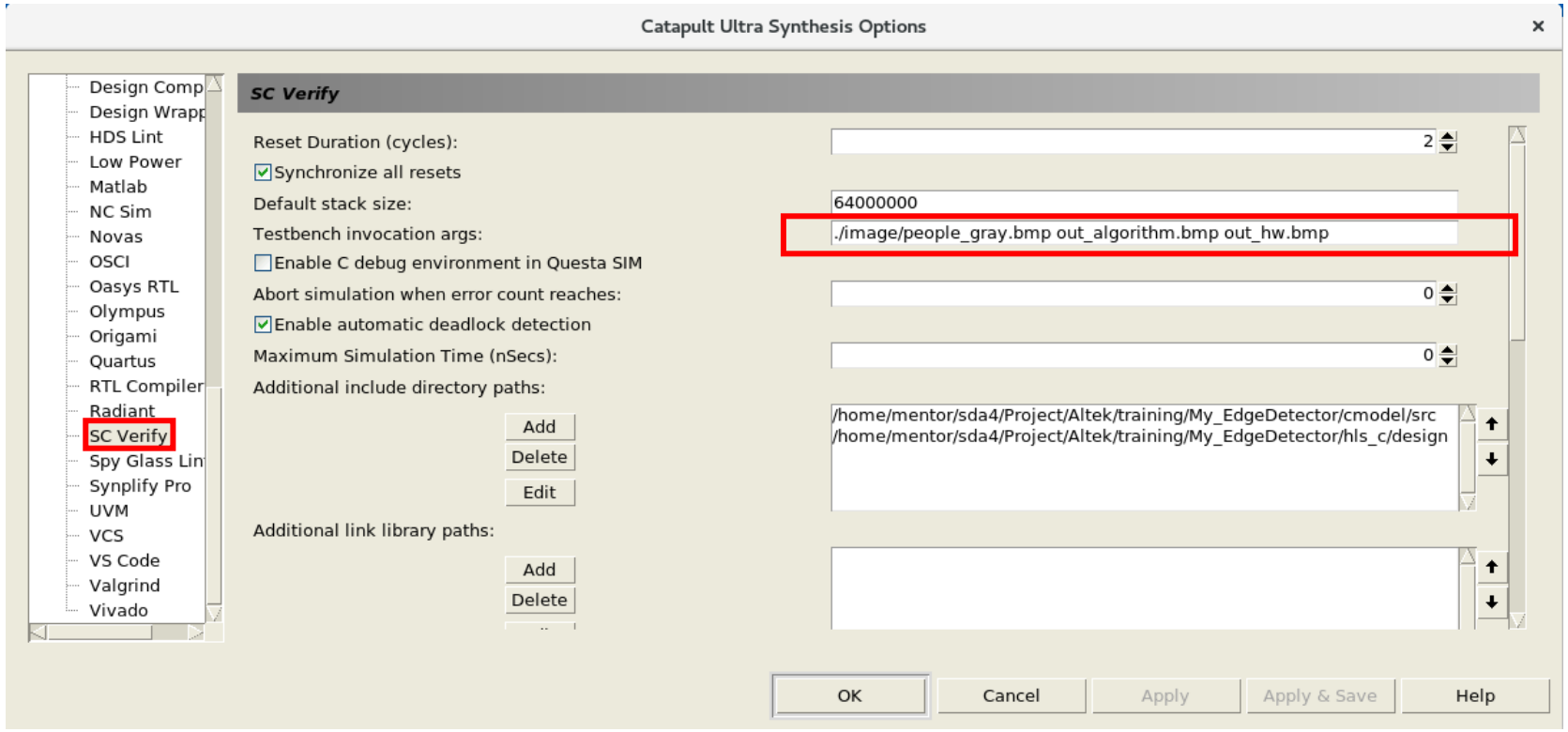
Add include path



Add testbench invocation args

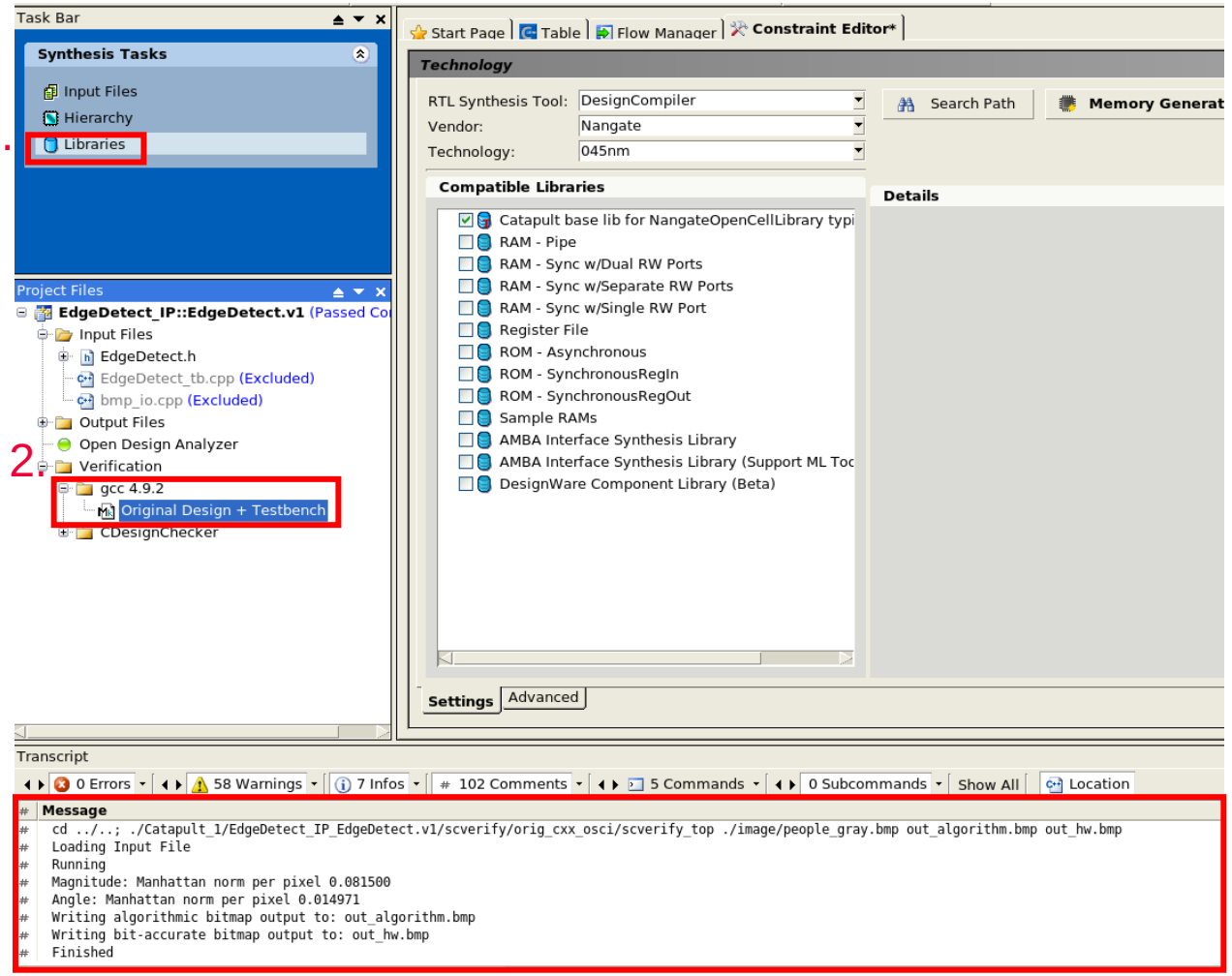
Assign arguments for running the testbench

- SCVerify_EdgeDetect.exe **./image/people_gray.bmp out_algorithm.bmp out_hw.bmp**



SIEMENS

Run C++ Verification (Pre-HLS Verification)



SIEMENS

Run C++ Verification (Cont.)

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

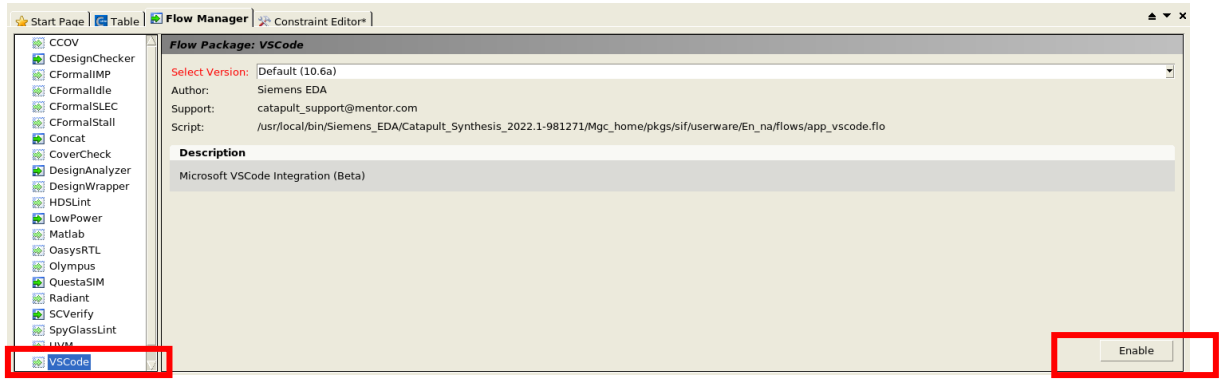
```
[yuchang@a4sa46 catapult_work]$ ll ./Catapult/EdgeDetect_IP_EdgeDetect.v1/scverify/orig_cxx_osci/scverify_top
```

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

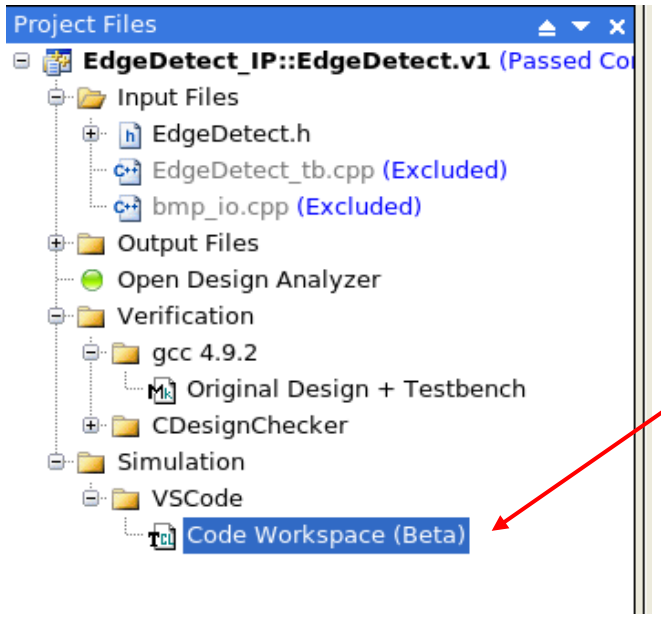
```
[yuchang@a4sa46 bin]$ ./SCVerify_EdgeDetect.exe ./image/people_gray.bmp out_algorithm.bmp out_hw.bmp
Loading Input File
Running
Magnitude: Manhattan norm per pixel 0.081500
Angle: Manhattan norm per pixel 0.014971
Writing algorithmic bitmap output to: out_algorithm.bmp
Writing bit-accurate bitmap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

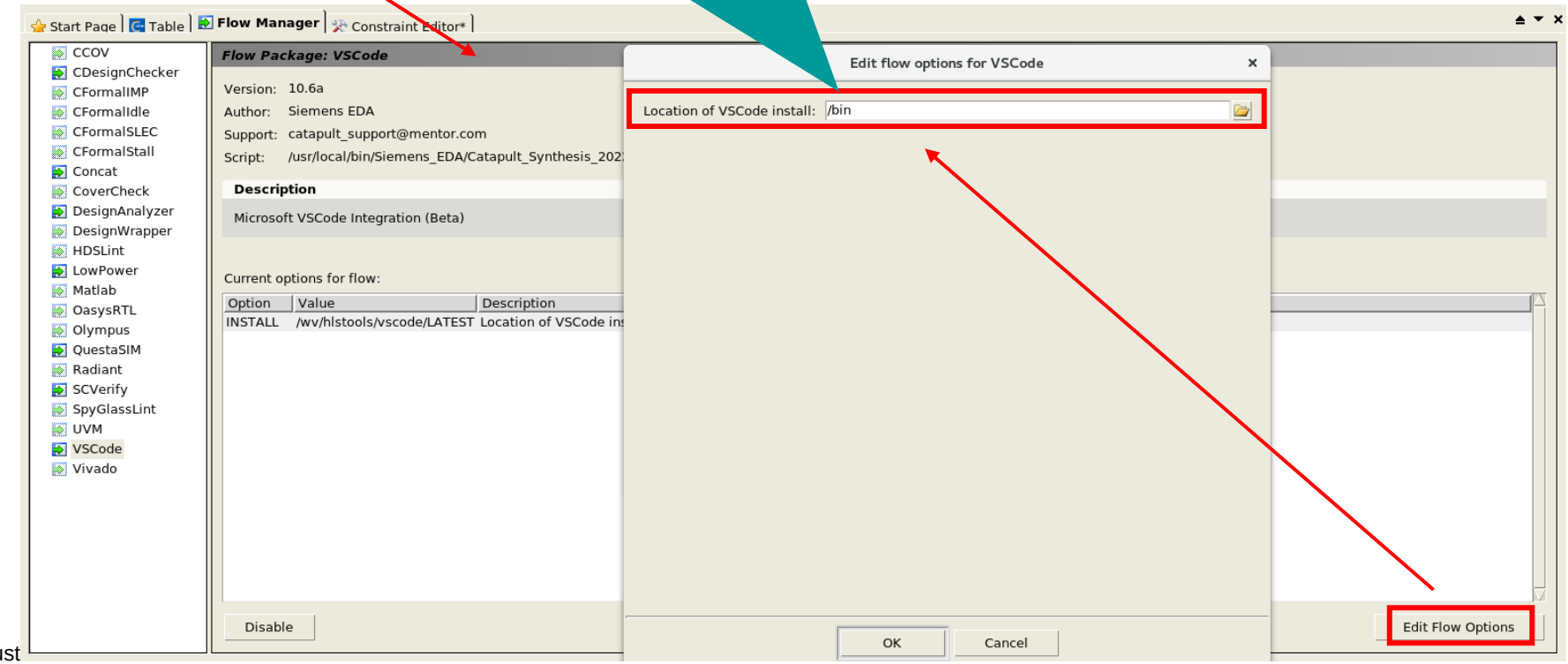

SIEMENS

Catapult can help build VSCode environment



Set VSCode install path

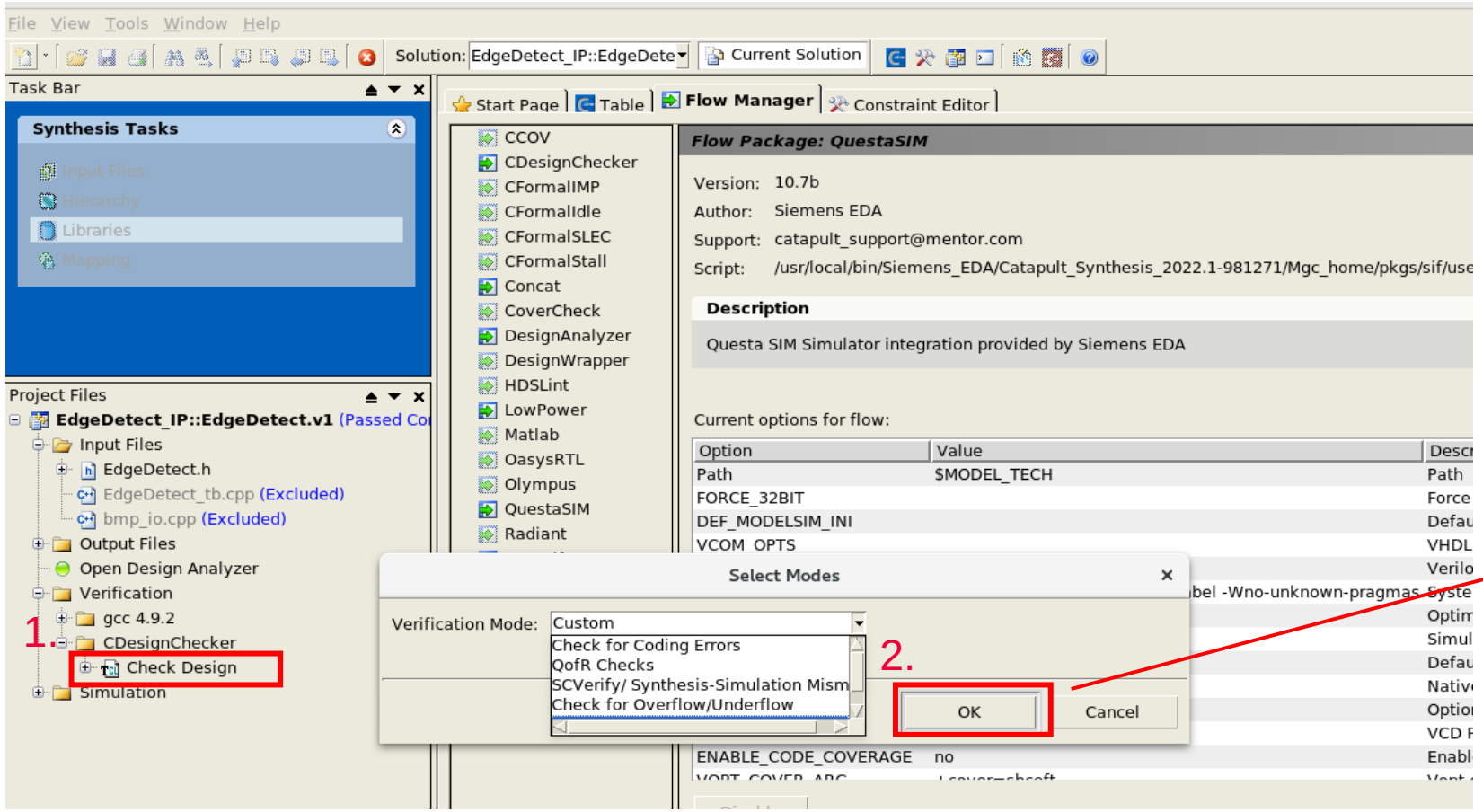


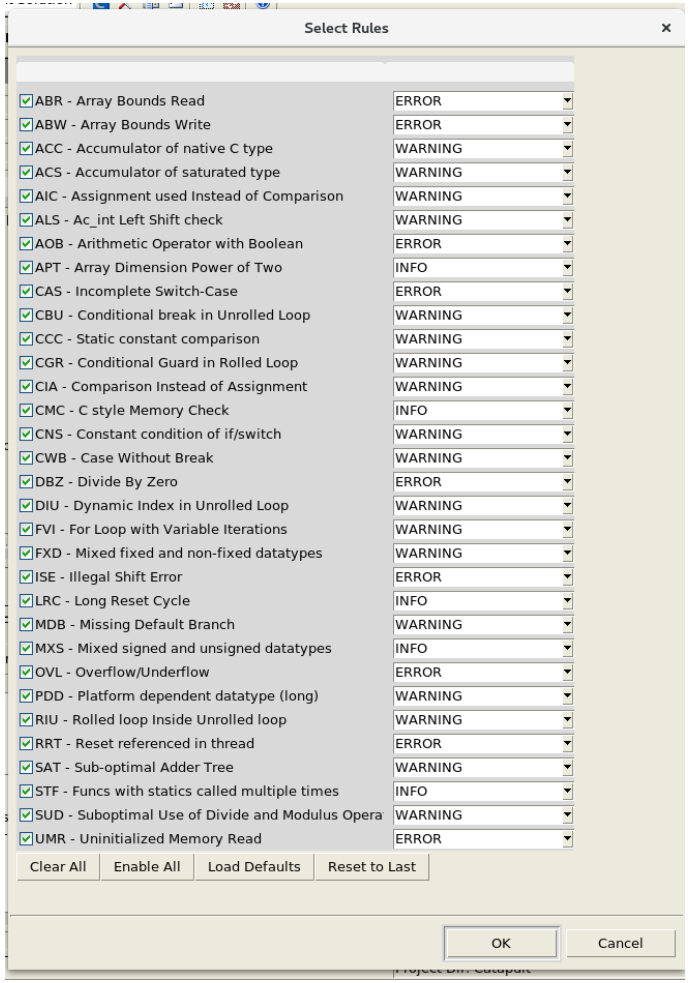


Run CDesignChecker

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

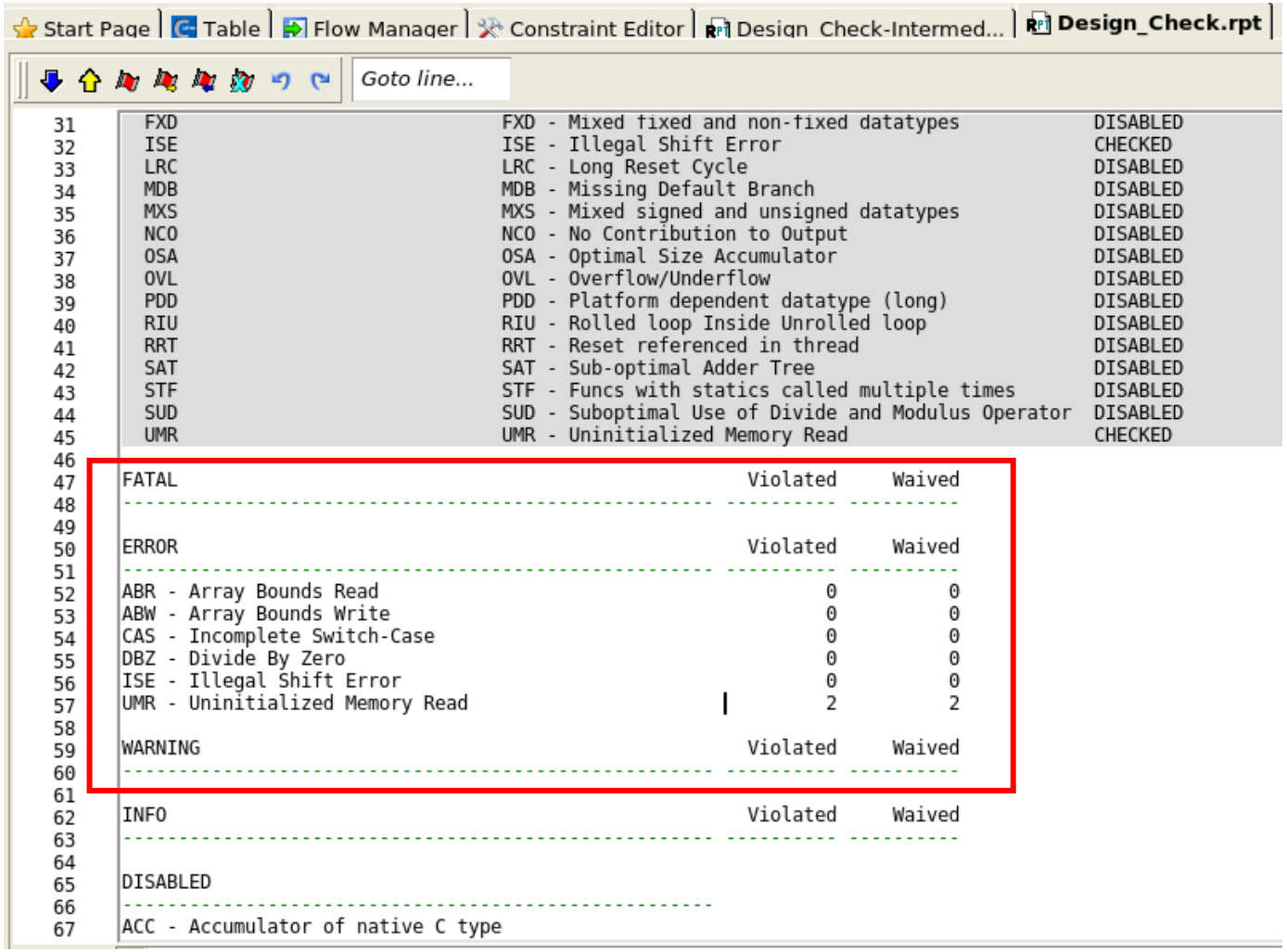
Very important to find the design issue before synthesis





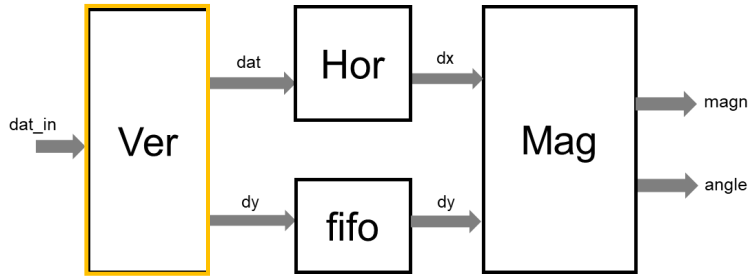
SIEMENS

CDesignChecker Report



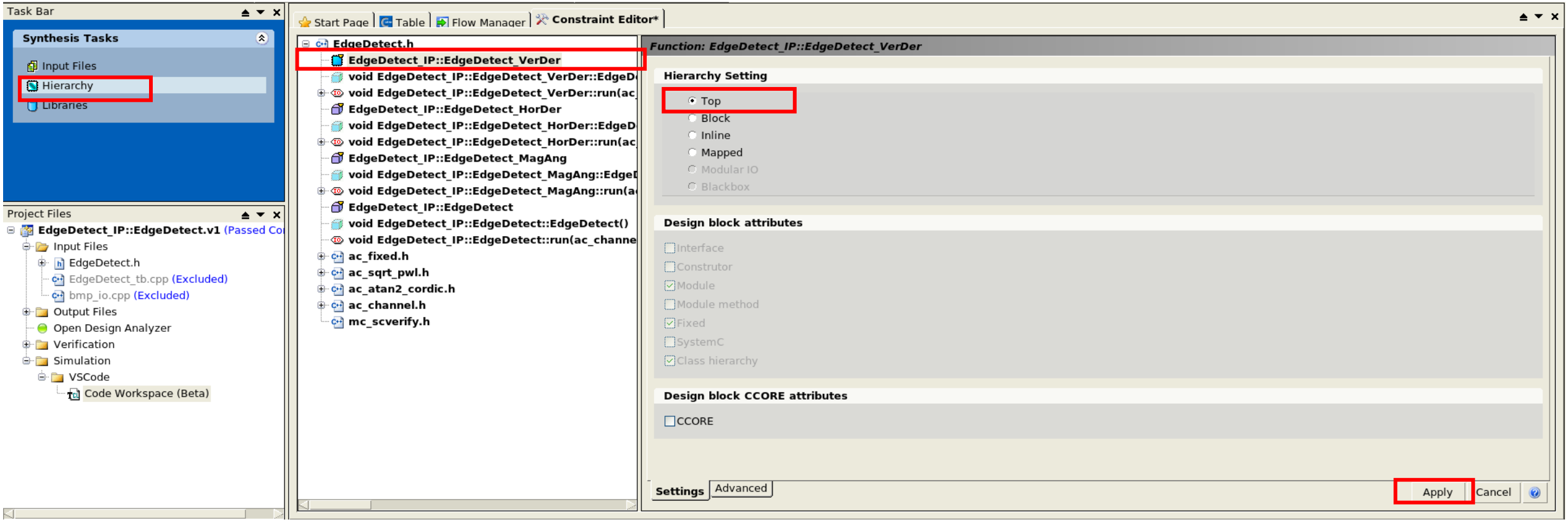
SIEMENS

Synthesize EdgeDetect_VerDer



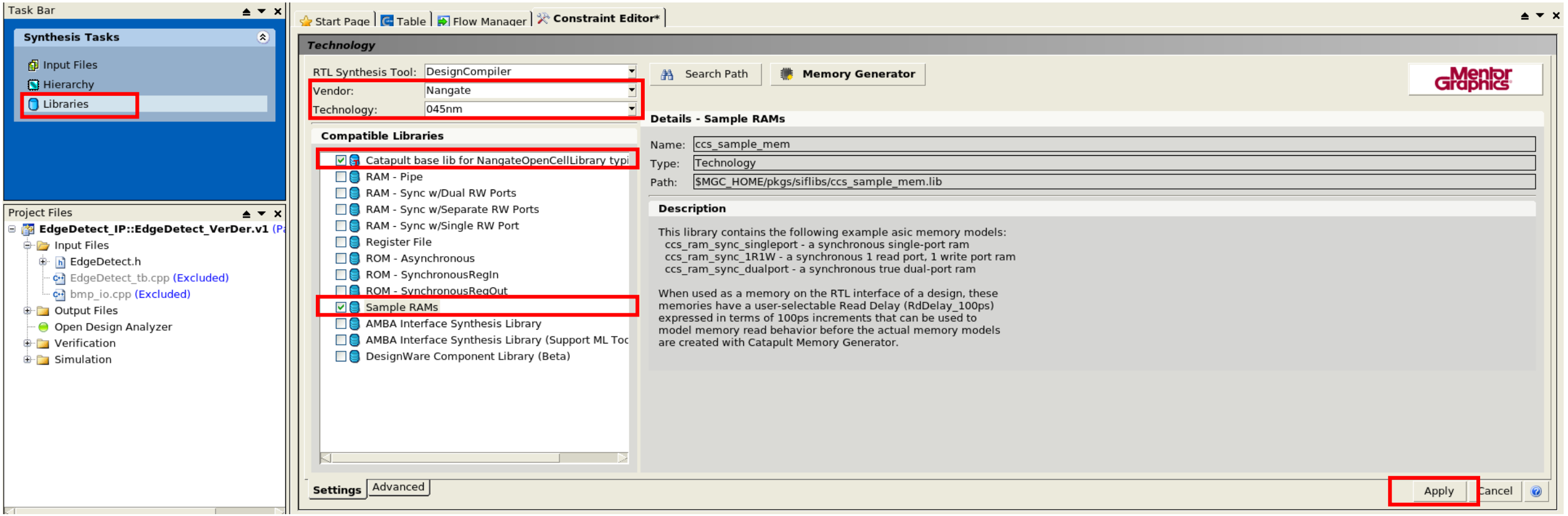
Bot-up synthesis

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



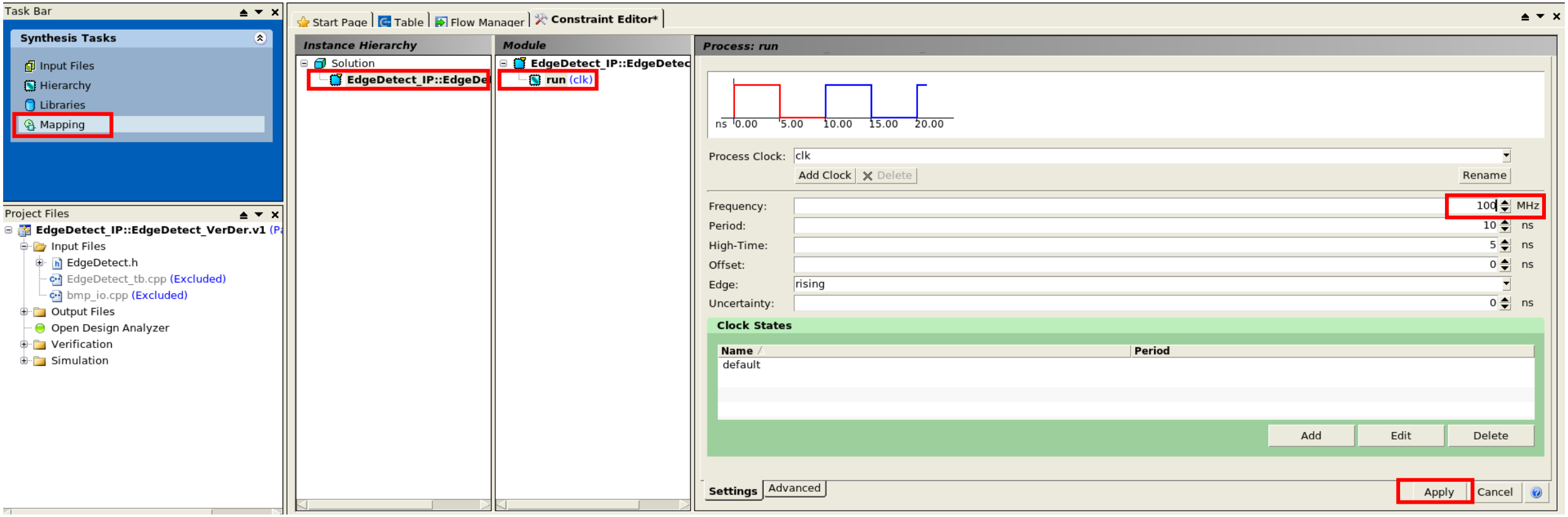
Library Setup

Use Nangate ASIC library and Sample RAMs



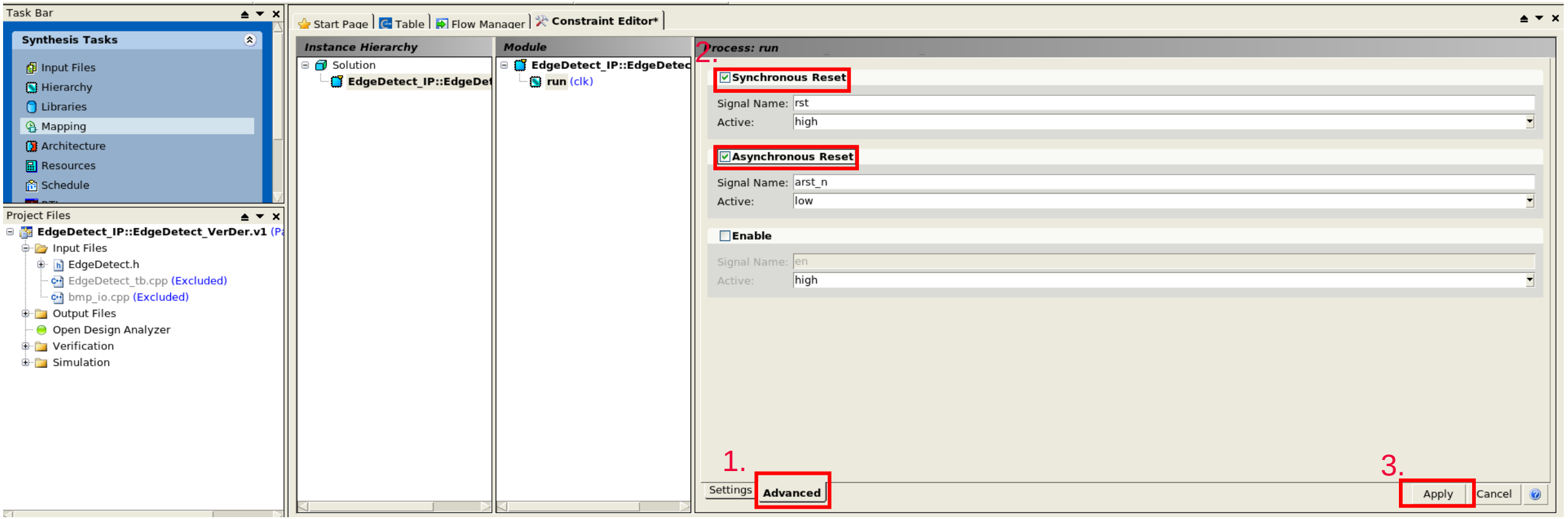
Mapping

Set frequency as 100MHz



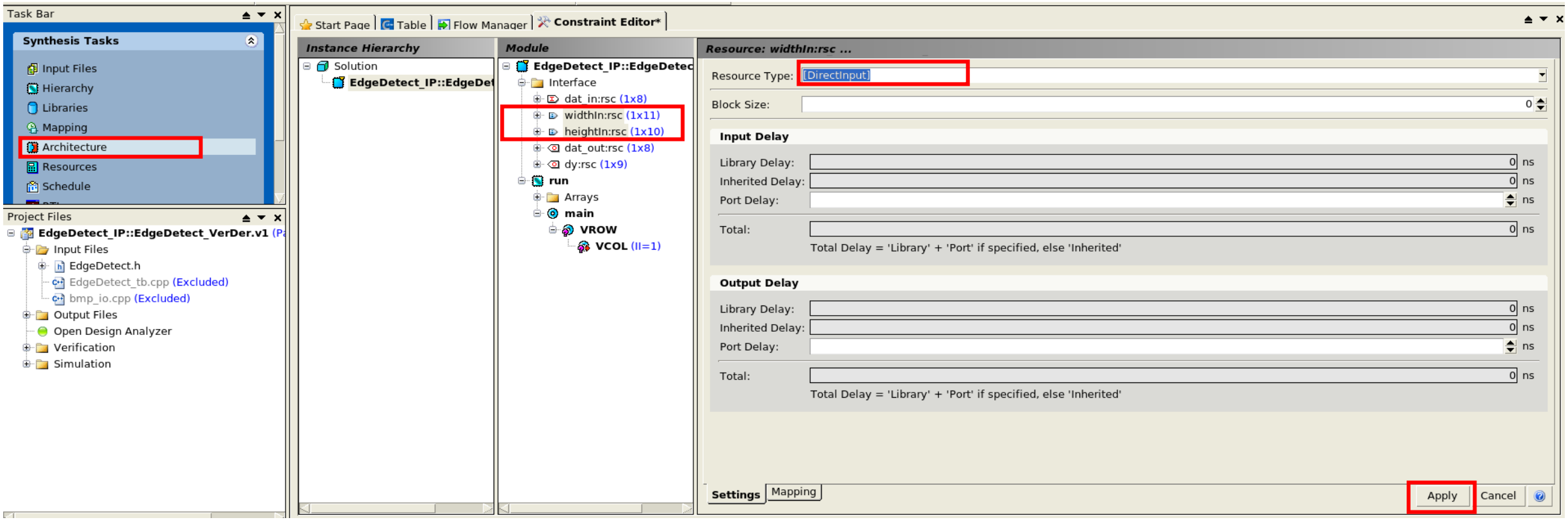
Mapping (Cont.)

Add synchronous and asynchronous reset



Architecture

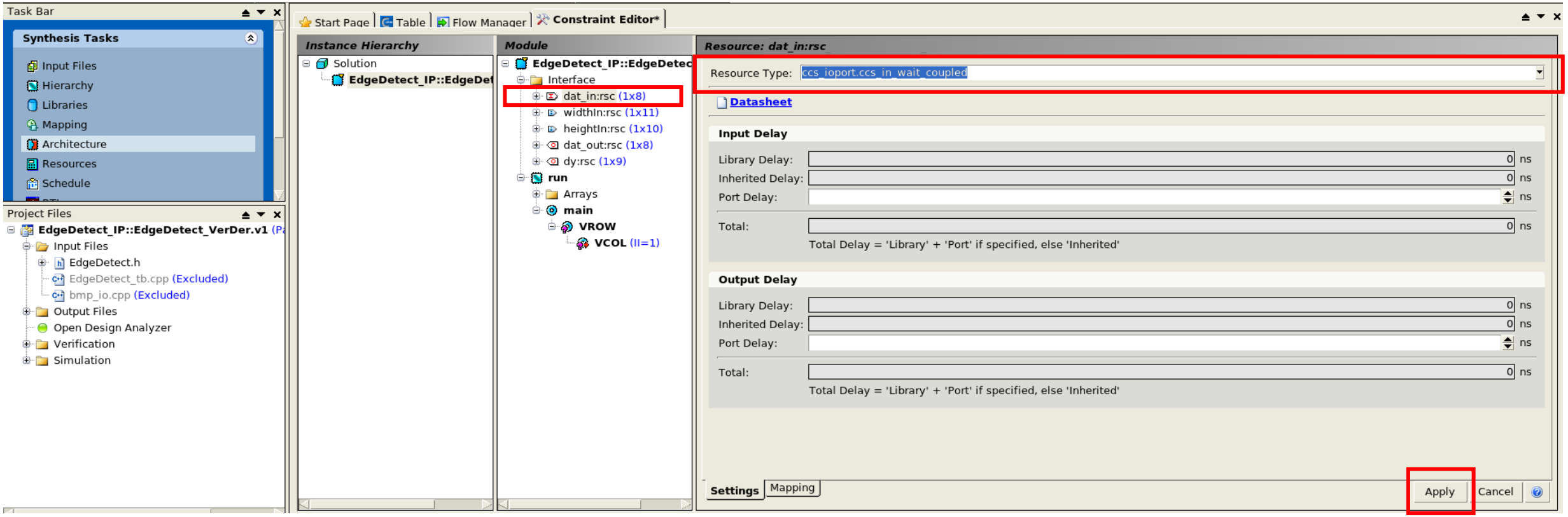
DirectInput is used for the stable ports avoiding the unnecessary pipeline registers inserted



SIEMENS

Architecture (Cont.)

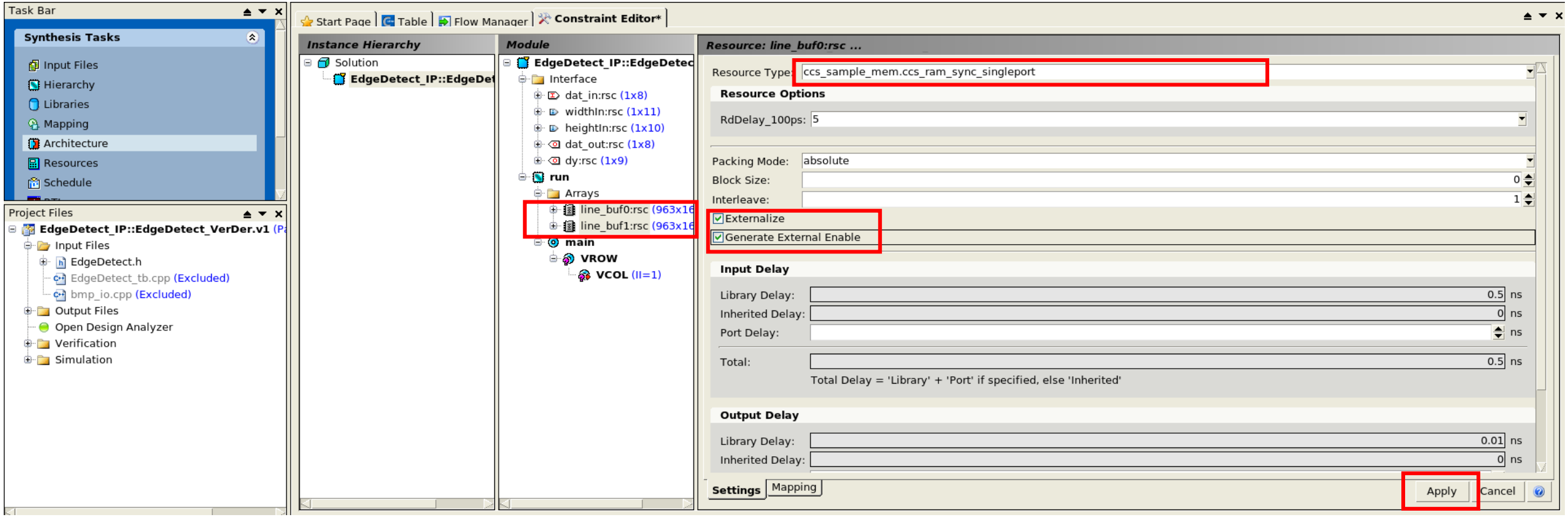
data input channel is set as ccs_in_wait_coupled avoiding extra fifo buffer



SIEMENS

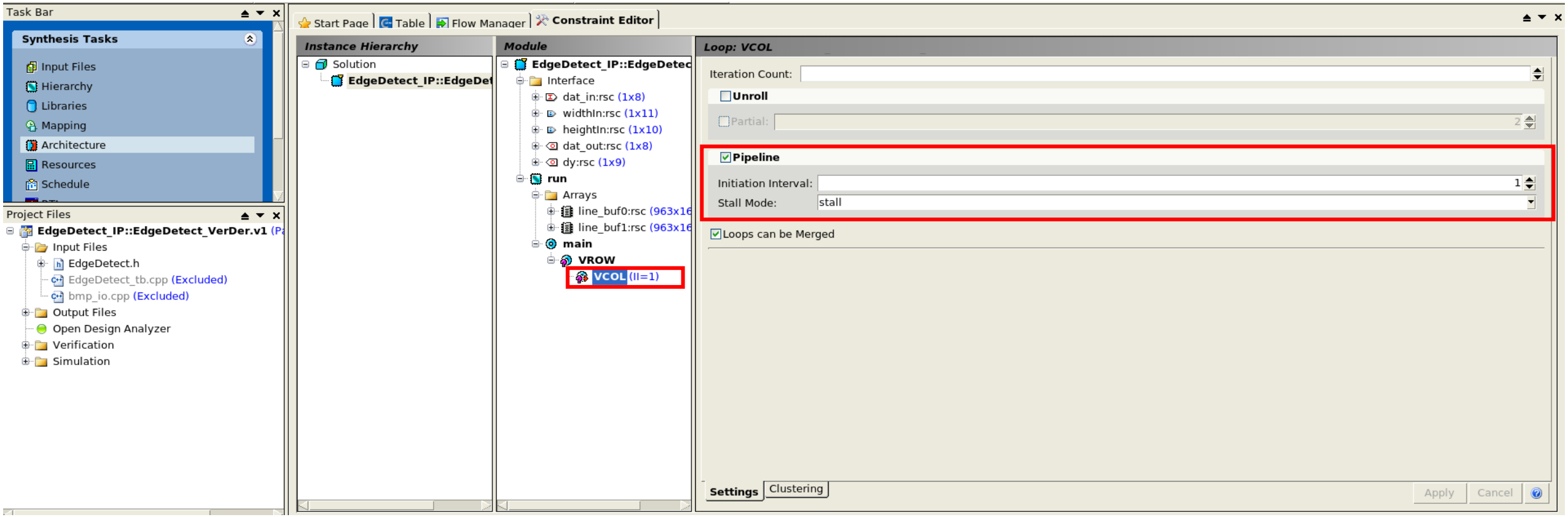
Architecture (Cont.)

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.)

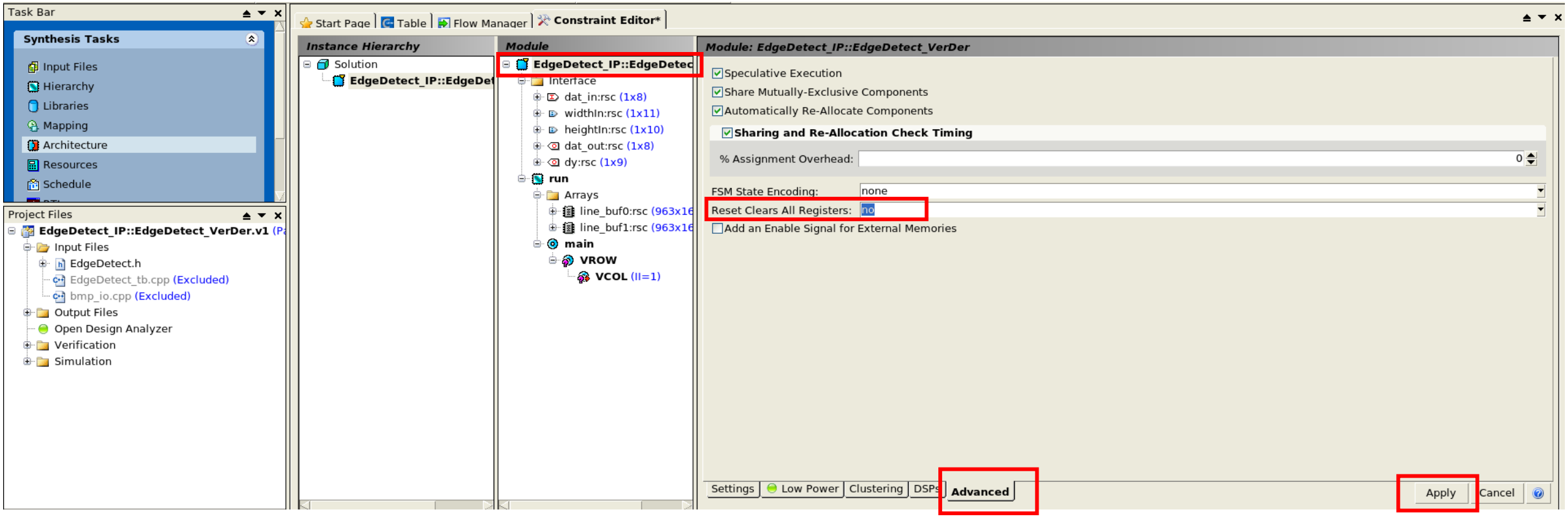
An alternative to set II1 is using the pragma hls_pipeline_init_interval 1

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

Architecture (Cont.)

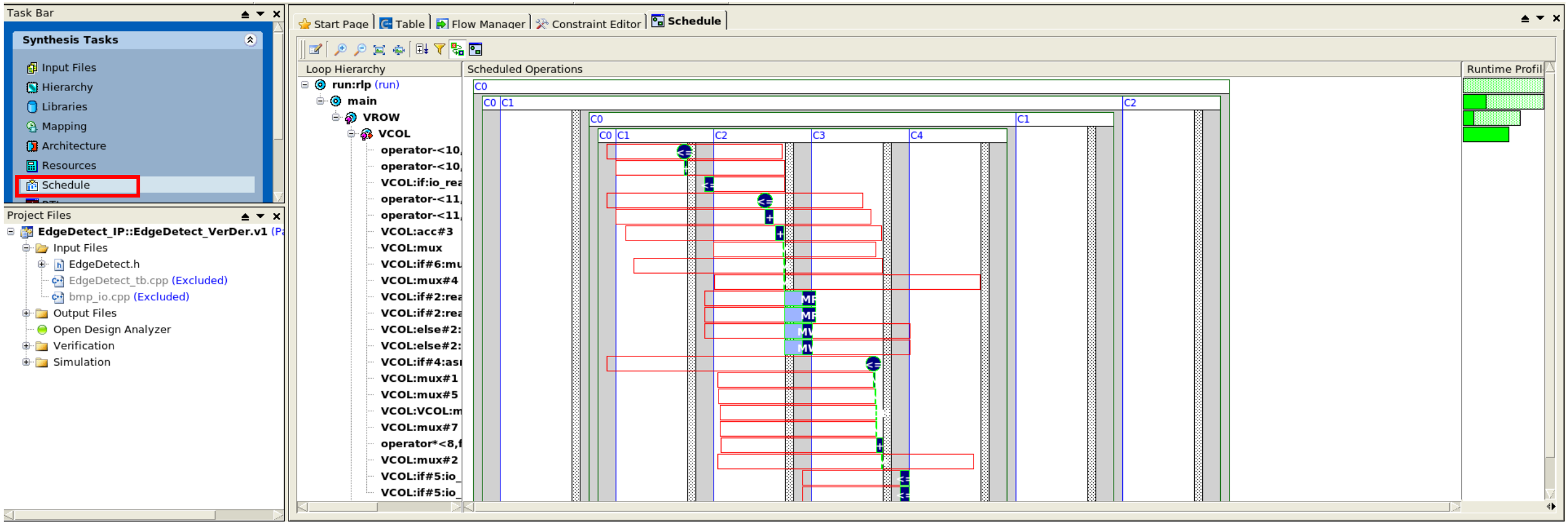
You can turn off resets on all registers for area reduction

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



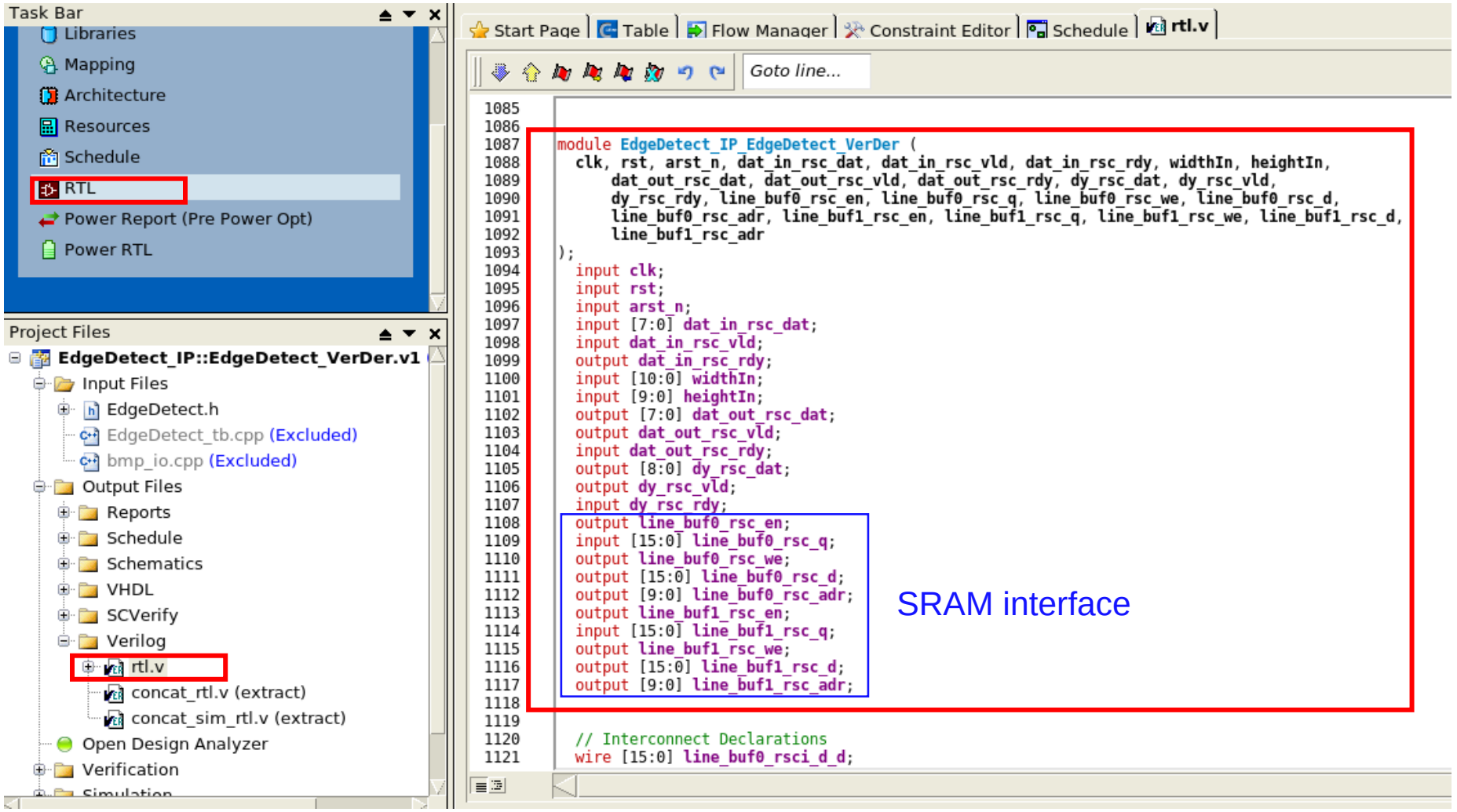
Scheduling

Pipeline stages are created



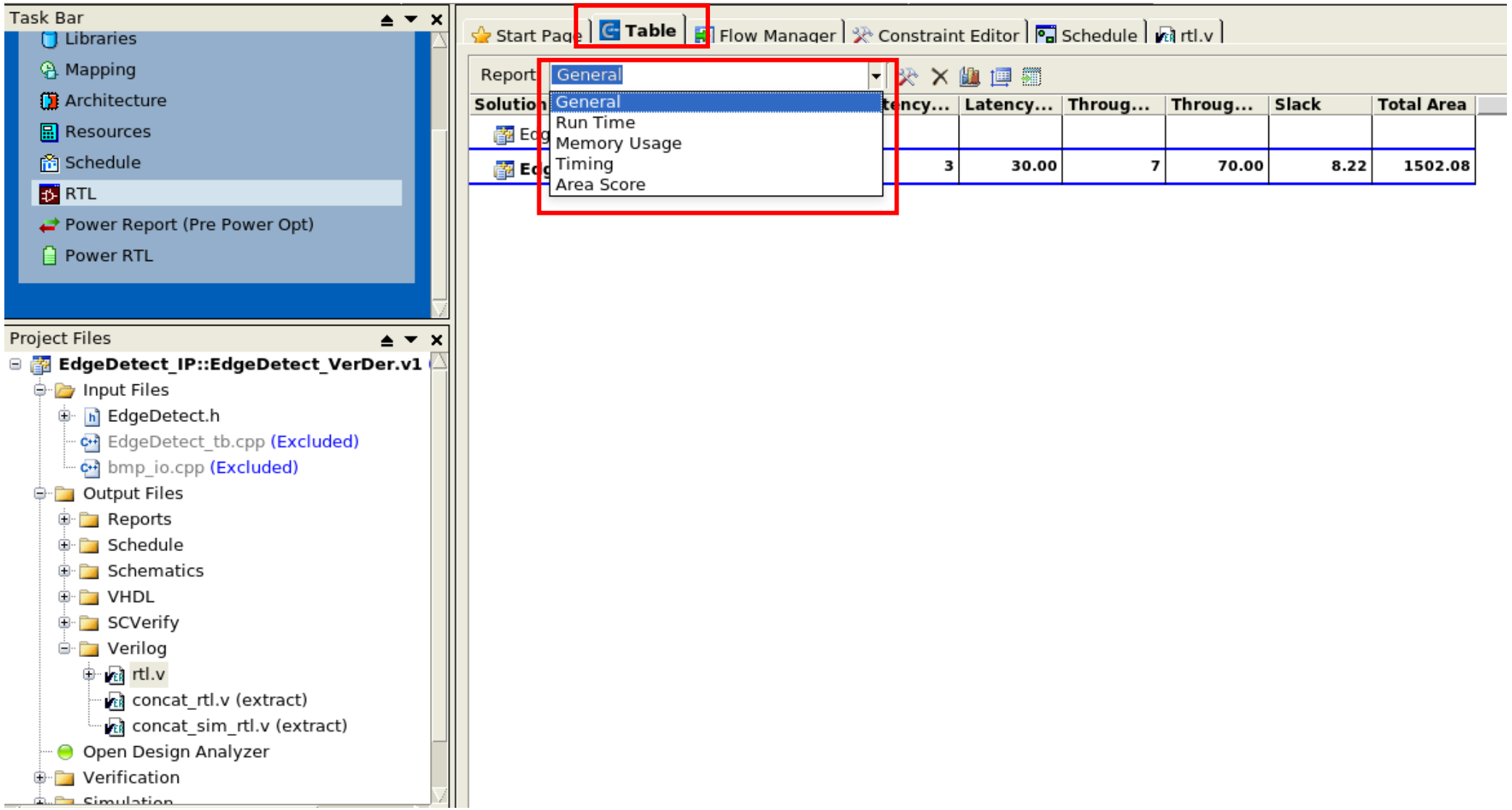
SIEMENS

Generate RTL



SIEMENS

Synthesis Report

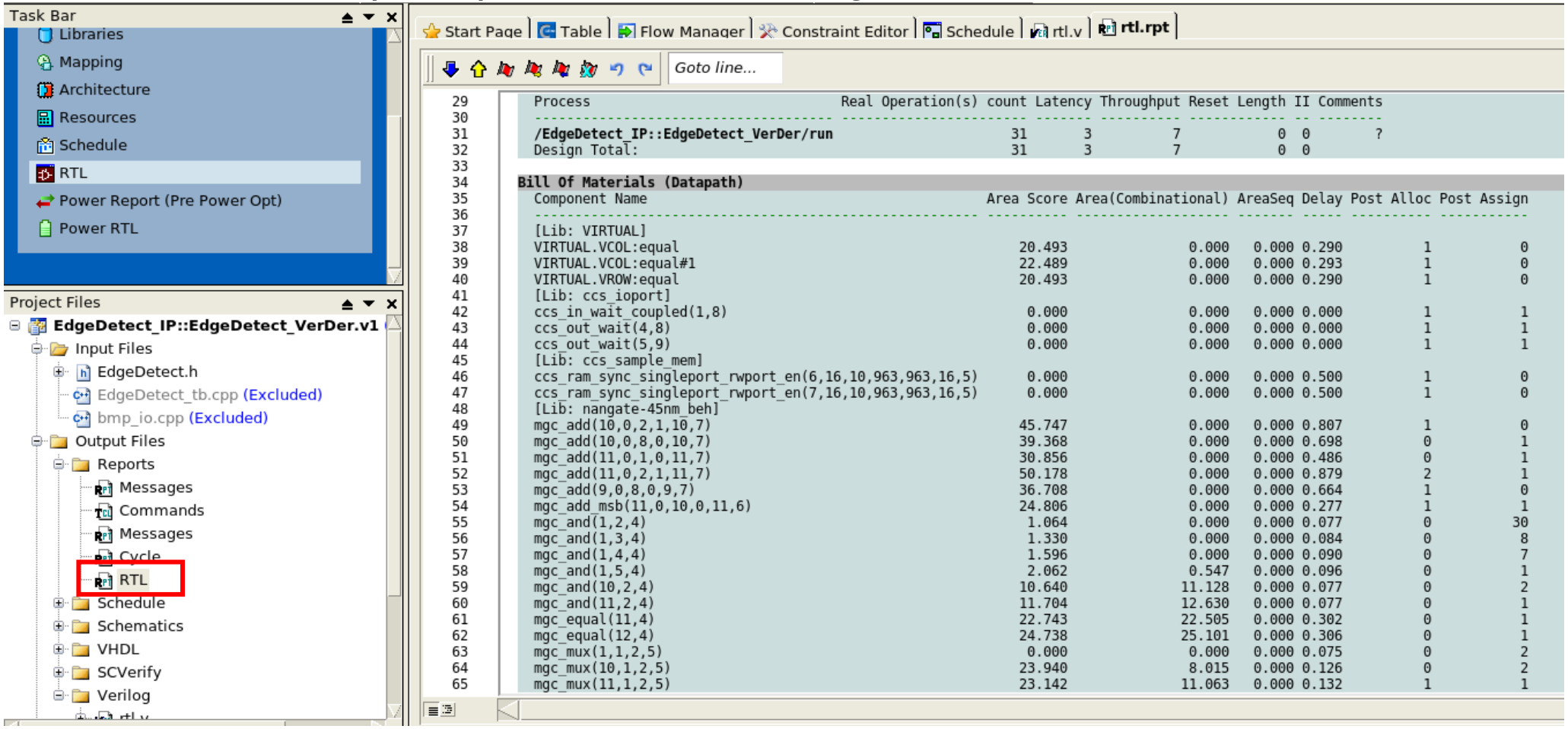


SIEMENS

Synthesis Report (Cont.)

Detailed information

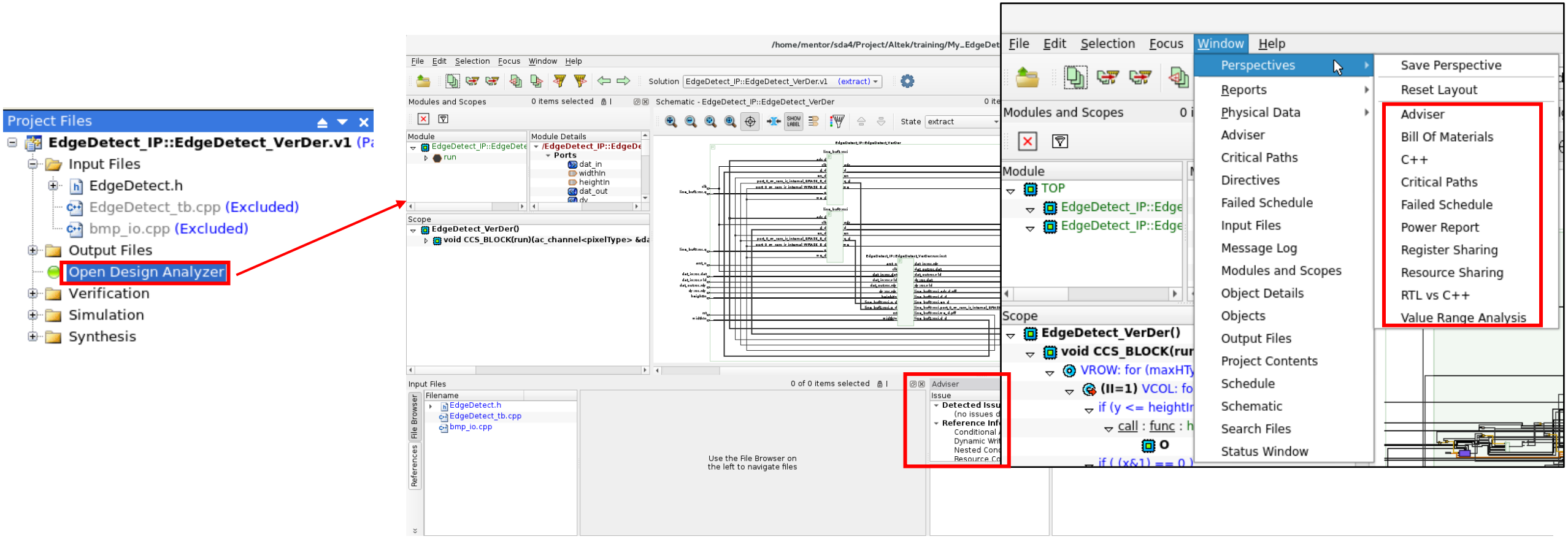
- Bill of Materials (BOM), Area Scores, Register-to-Variable, …



SIEMENS

Design Analyzer

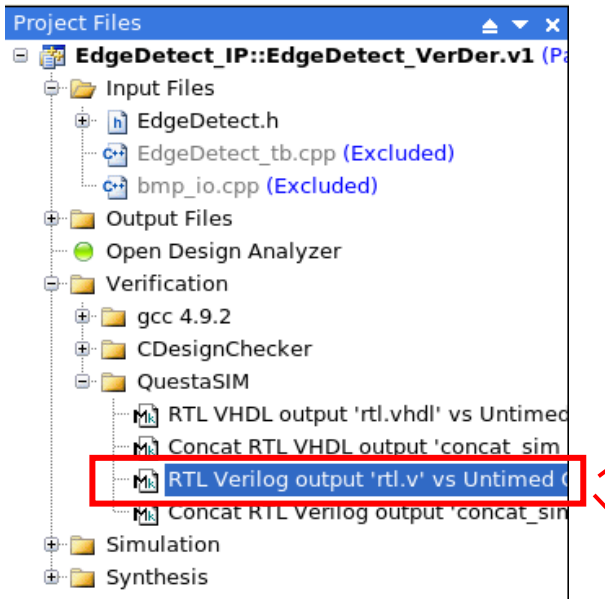
Analyze the design in various perspectives such as Advier, BOM, RTL vs C, …, etc.

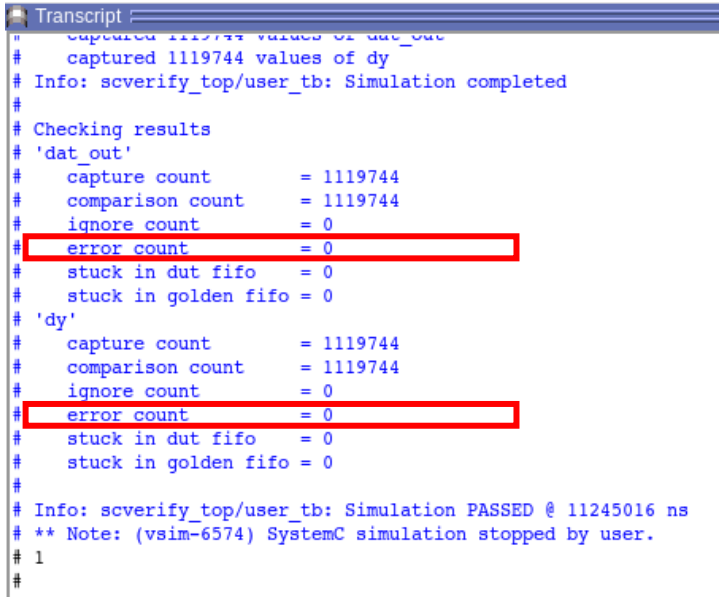


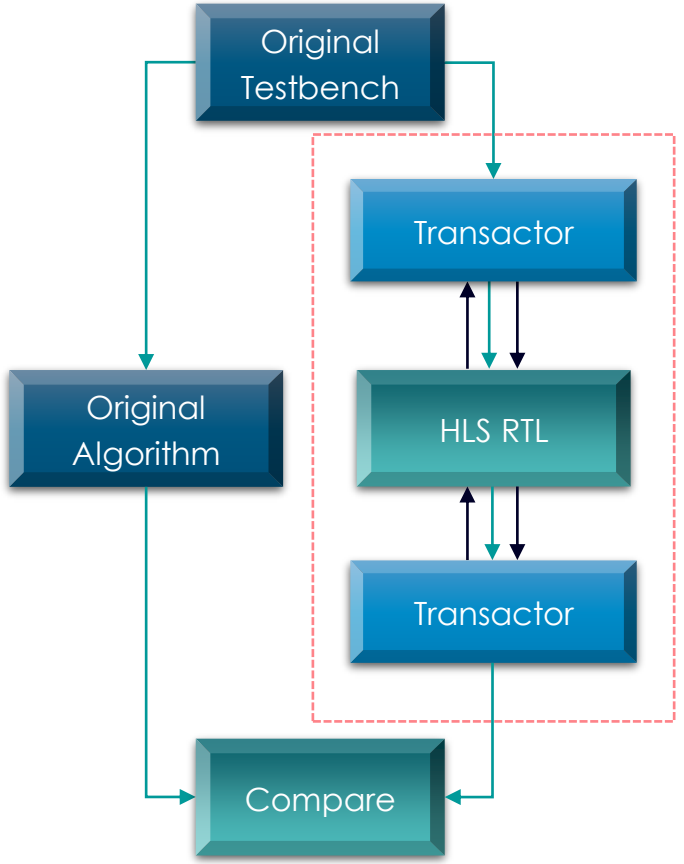
SIEMENS

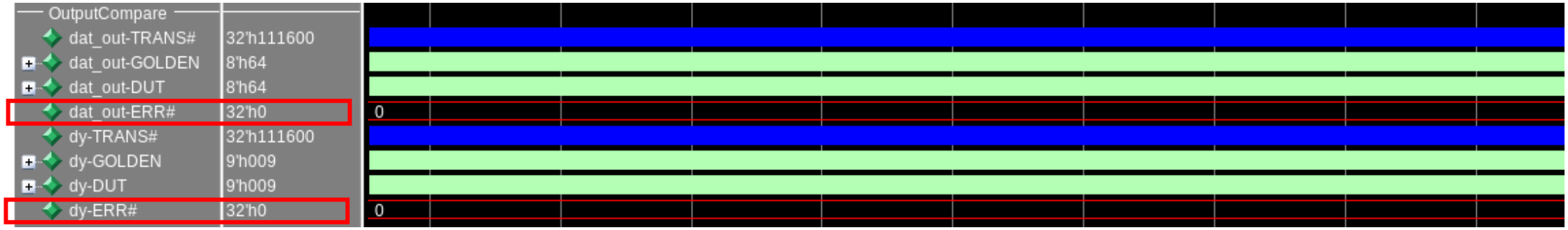
RTL Verification (Cont.)

Automatically compare the outputs of HLC C and RTL



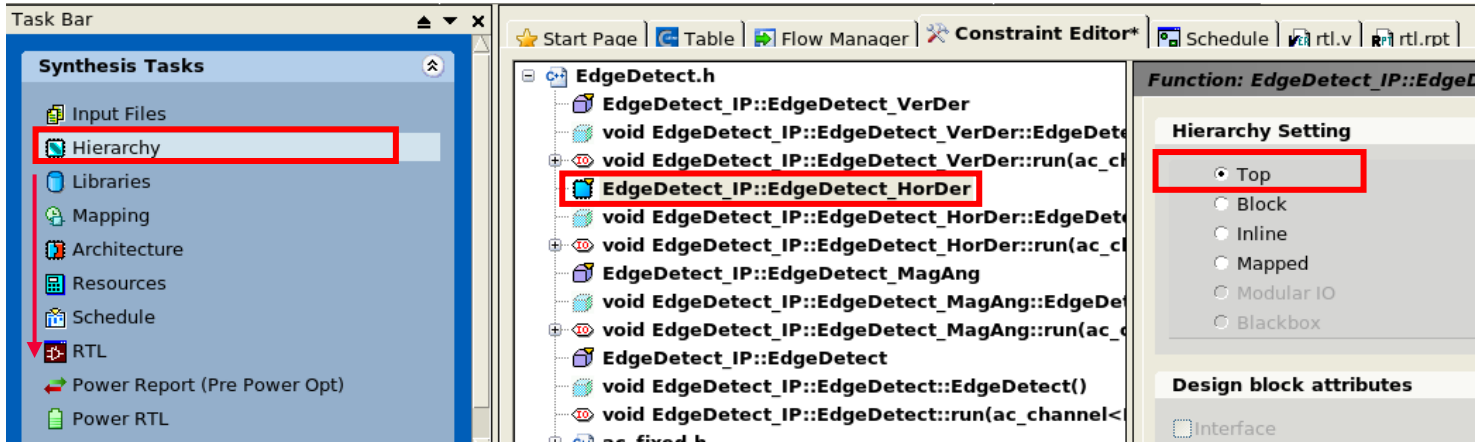


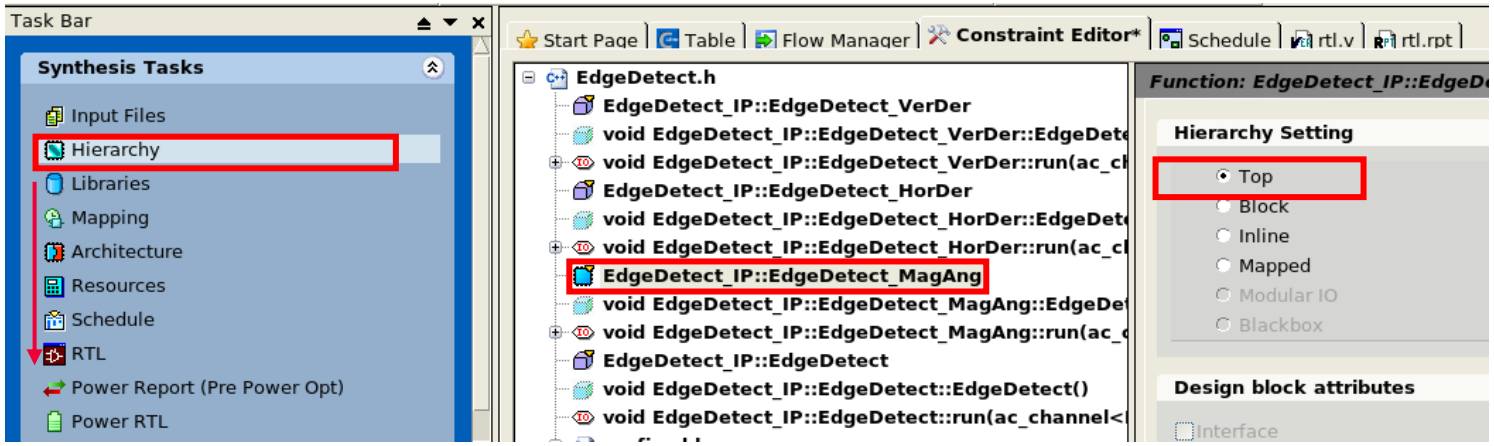


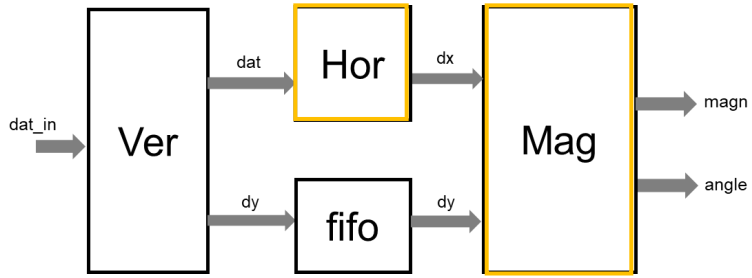


SIEMENS

Synthesize *_HorDer and *_MagAng by yourself



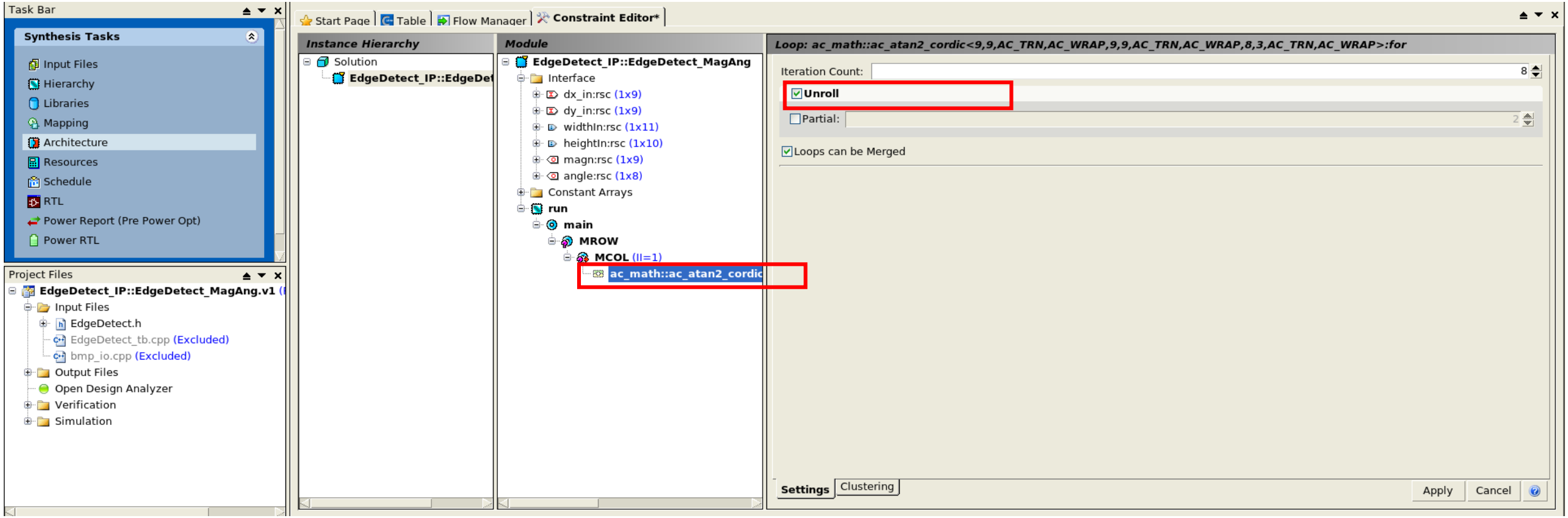




Architecture of *_MagAng

Unroll the for loop within MCOL

MCOL is II1

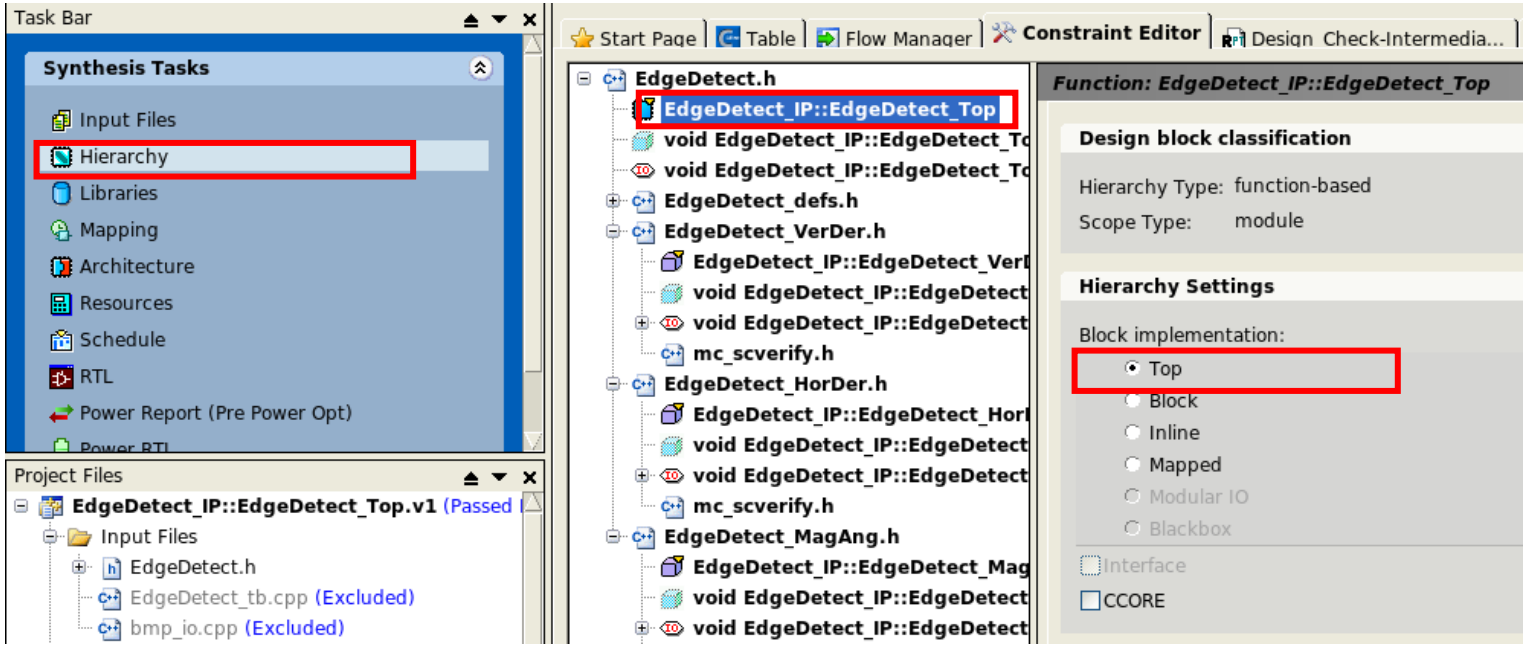


SIEMENS

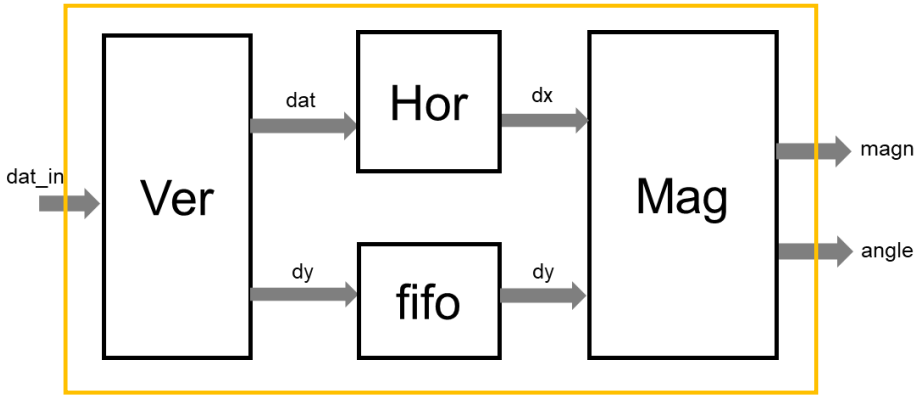
Synthesize the *_Top

Now *_VerDer, *_HorDer, and *_MagAng have been synthesized and verified

It’s time to synthesize *_Top



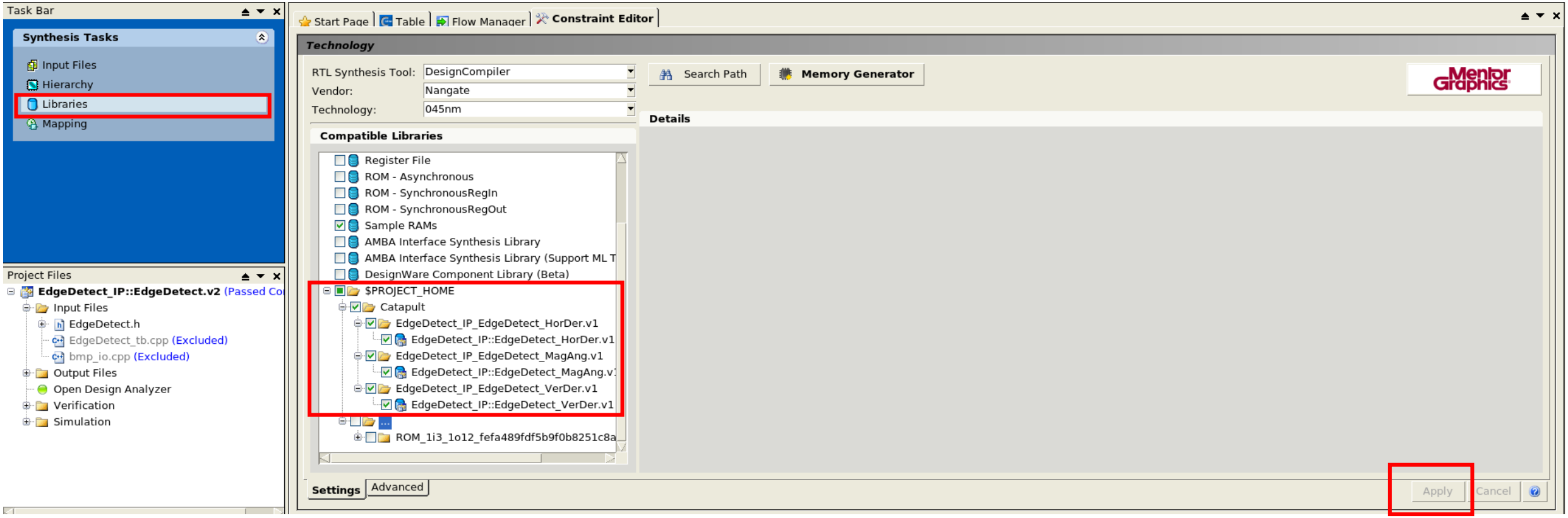
EdgeDetect_Top



SIEMENS

Synthesize the *_Top (Cont.)

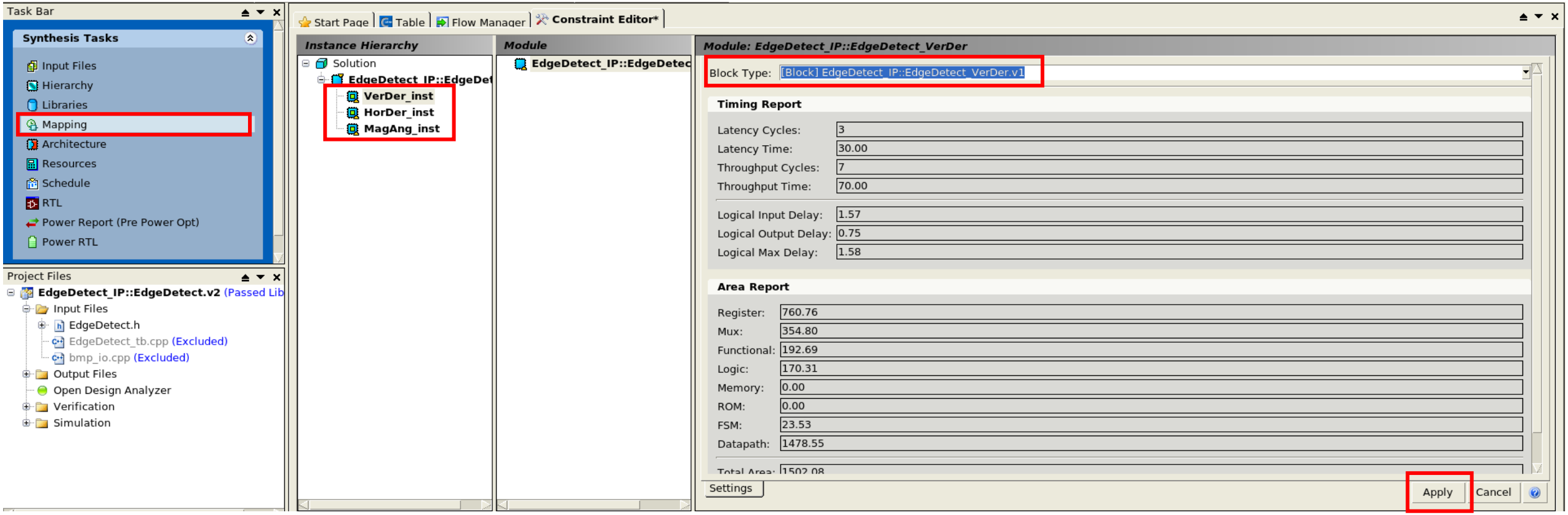
Add the libraries (*_VerDer, *_HorDer, and *_MagAng)



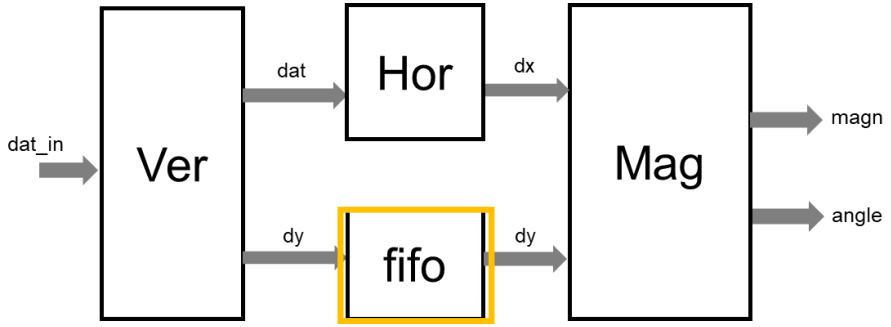
SIEMENS

Synthesize the *_Top (Cont.)

Mapping VerDer_inst, HorDer_inst, and MagAng_inst to the corresponding blocks



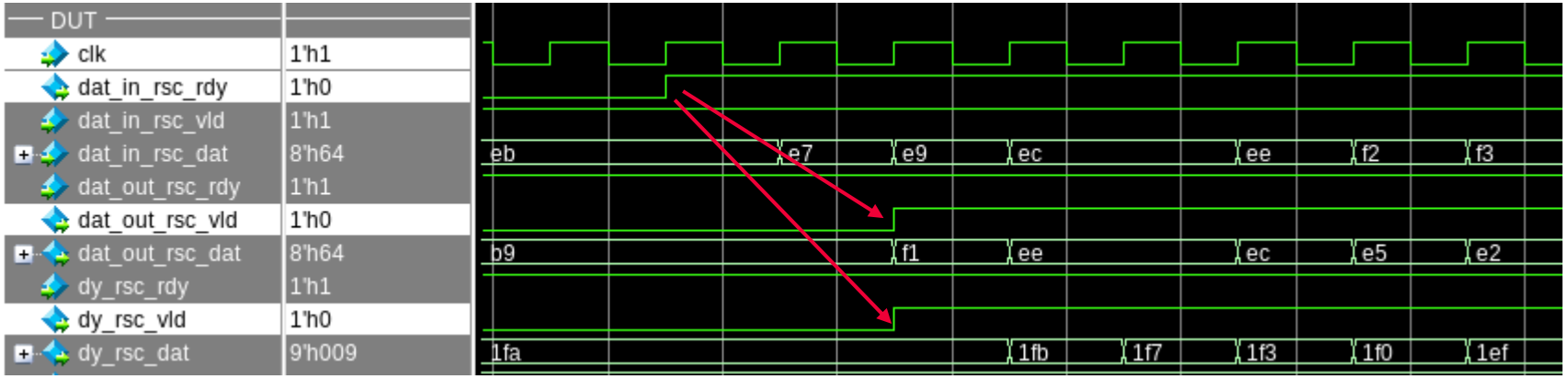
Synthesize the *_Top (Cont.)



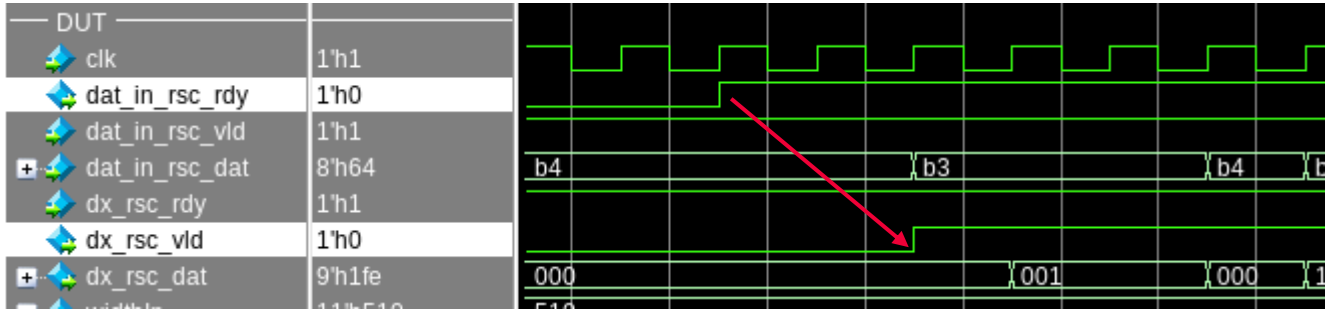
Decide the FIFO depth according to the latency of *_Ver and *_Hor

- dat_out : 2T
- dy_out : 2T
- dx_out : 2T

Ver

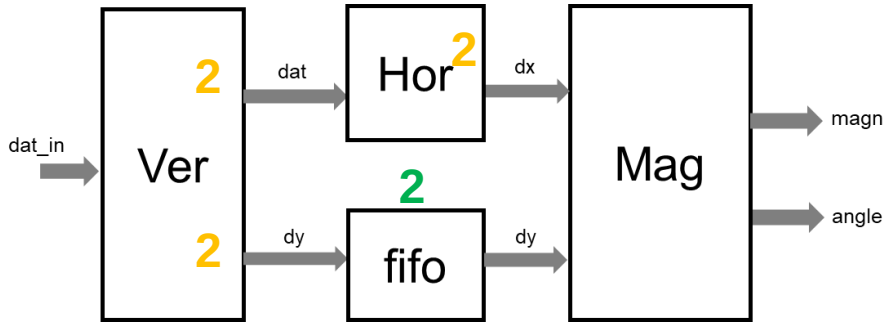


Hor



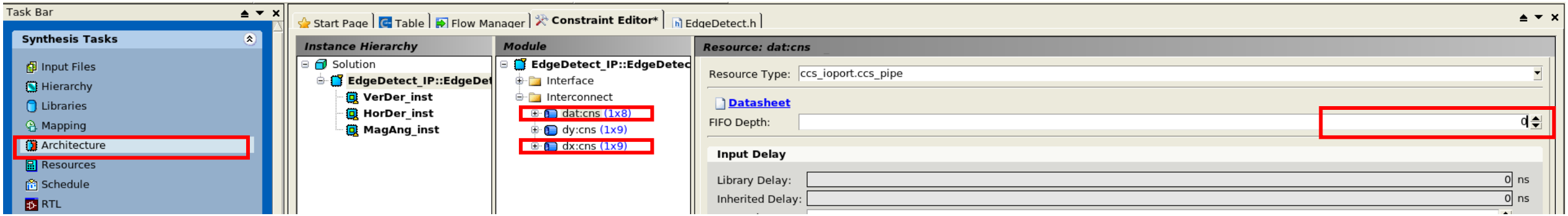
SIEMENS

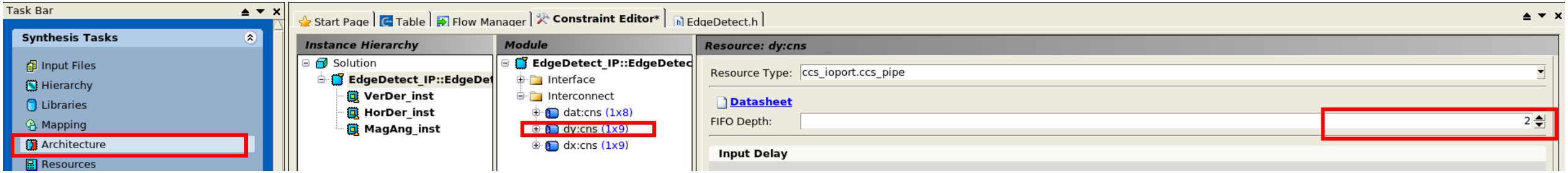
Synthesize the *_Top (Cont.)



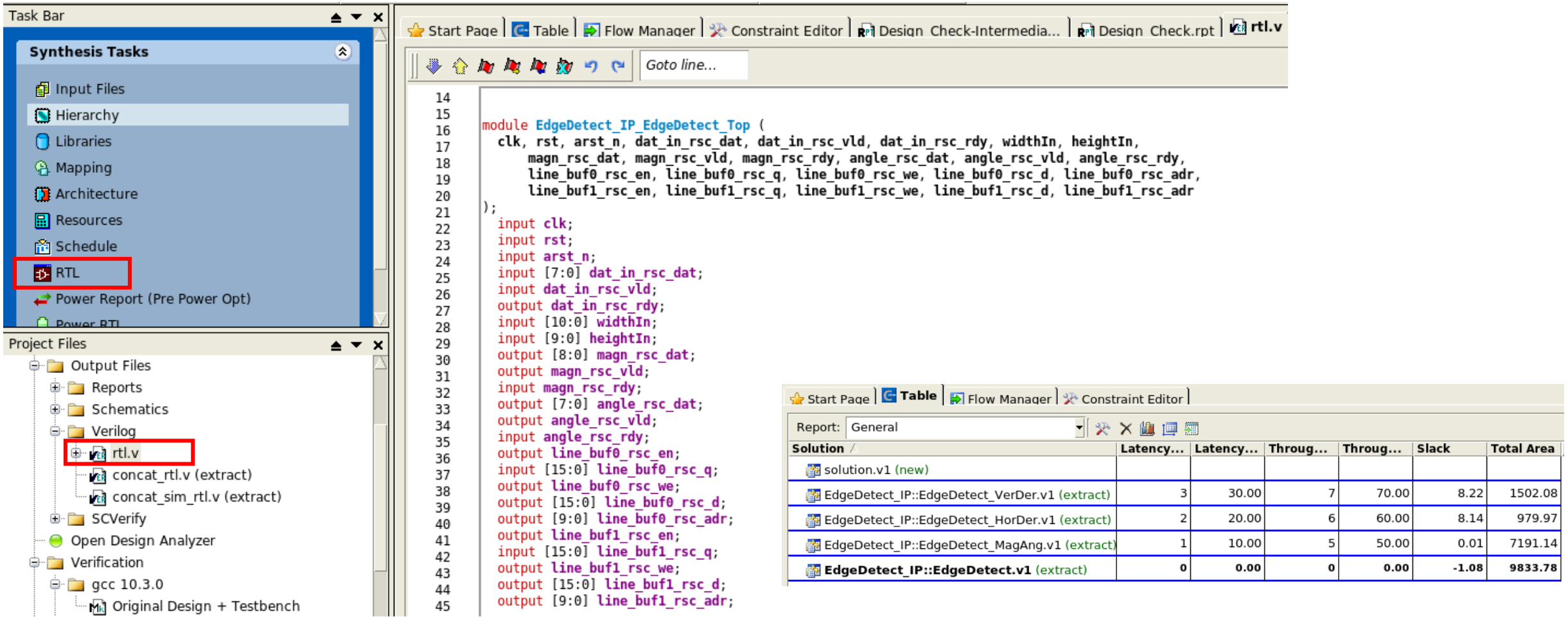
FIFO depth for each channel between blocks

- dat_chn: 0
- dx_chn: 0
- dy_chn: 2 + 2 - 2 = 2





Synthesize the *_Top (Cont.)



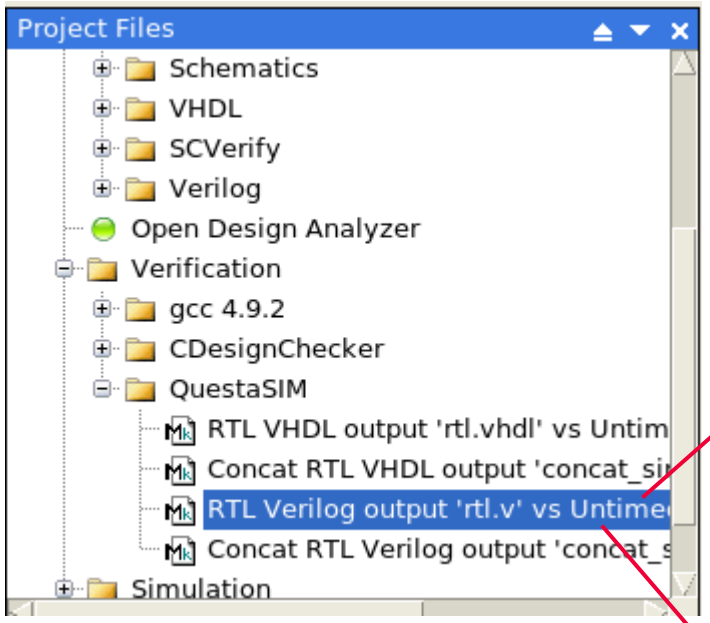
Start Page | **Table** | Flow Manager | Constraint Editor

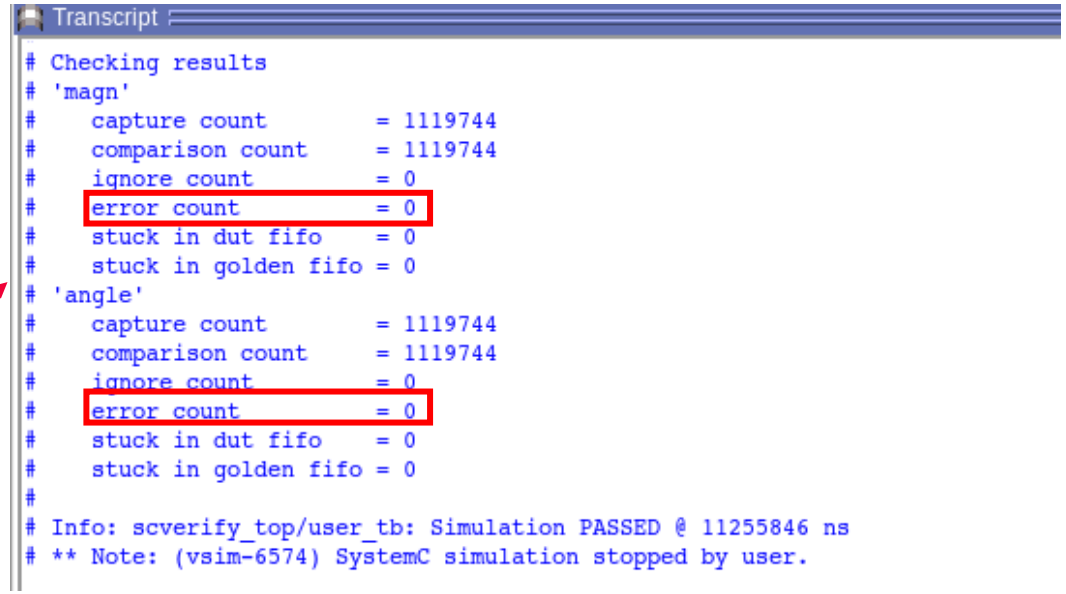
Report: General

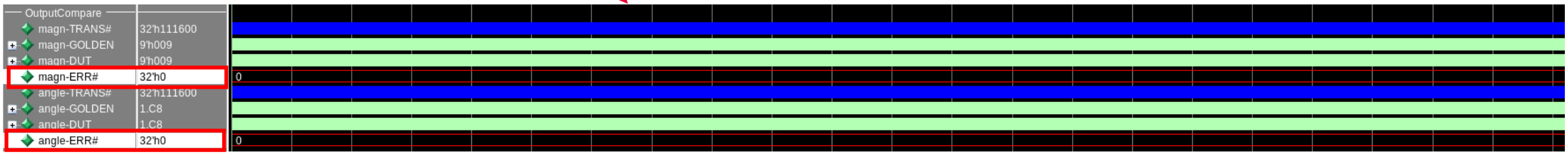
Solution	Latency...	Latency...	Throug...	Throug...	Slack	Total Area
solution.v1 (new)						
EdgeDetect_IP::EdgeDetect_VerDer.v1 (extract)	3	30.00	7	70.00	8.22	1502.08
EdgeDetect_IP::EdgeDetect_HorDer.v1 (extract)	2	20.00	6	60.00	8.14	979.97
EdgeDetect_IP::EdgeDetect_MagAng.v1 (extract)	1	10.00	5	50.00	0.01	7191.14
EdgeDetect_IP::EdgeDetect.v1 (extract)	**0**	**0.00**	**0**	**0.00**	**-1.08**	**9833.78**

SIEMENS

RTL Verification for *_Top





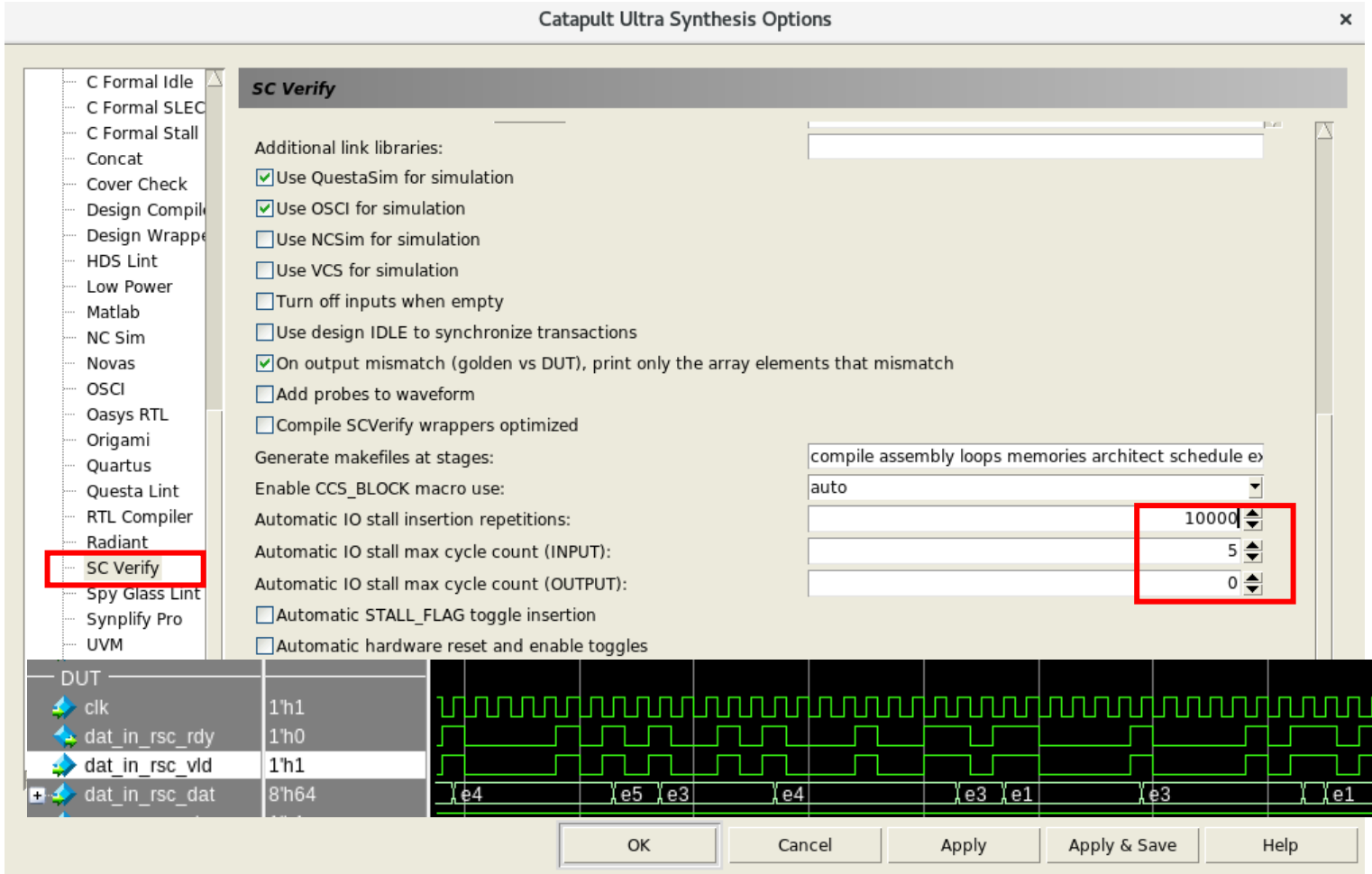


SIEMENS

RTL Verification for *_Top (Cont.)

Controlling stalling behavior in the testbench

- Stall dat_in, totally 10000 samples, for random cycles between 0 and 5



SIEMENS

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]$
```


SIEMENS

EdgeDetect IP for FSIC Project

02_edgedetect_fsic

SIEMENS

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
arst_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

SIEMENS

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

```
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]$
```

Build c++ source

- make

Run c++ simulation

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

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

SIEMENS

RTL Integration and Verification

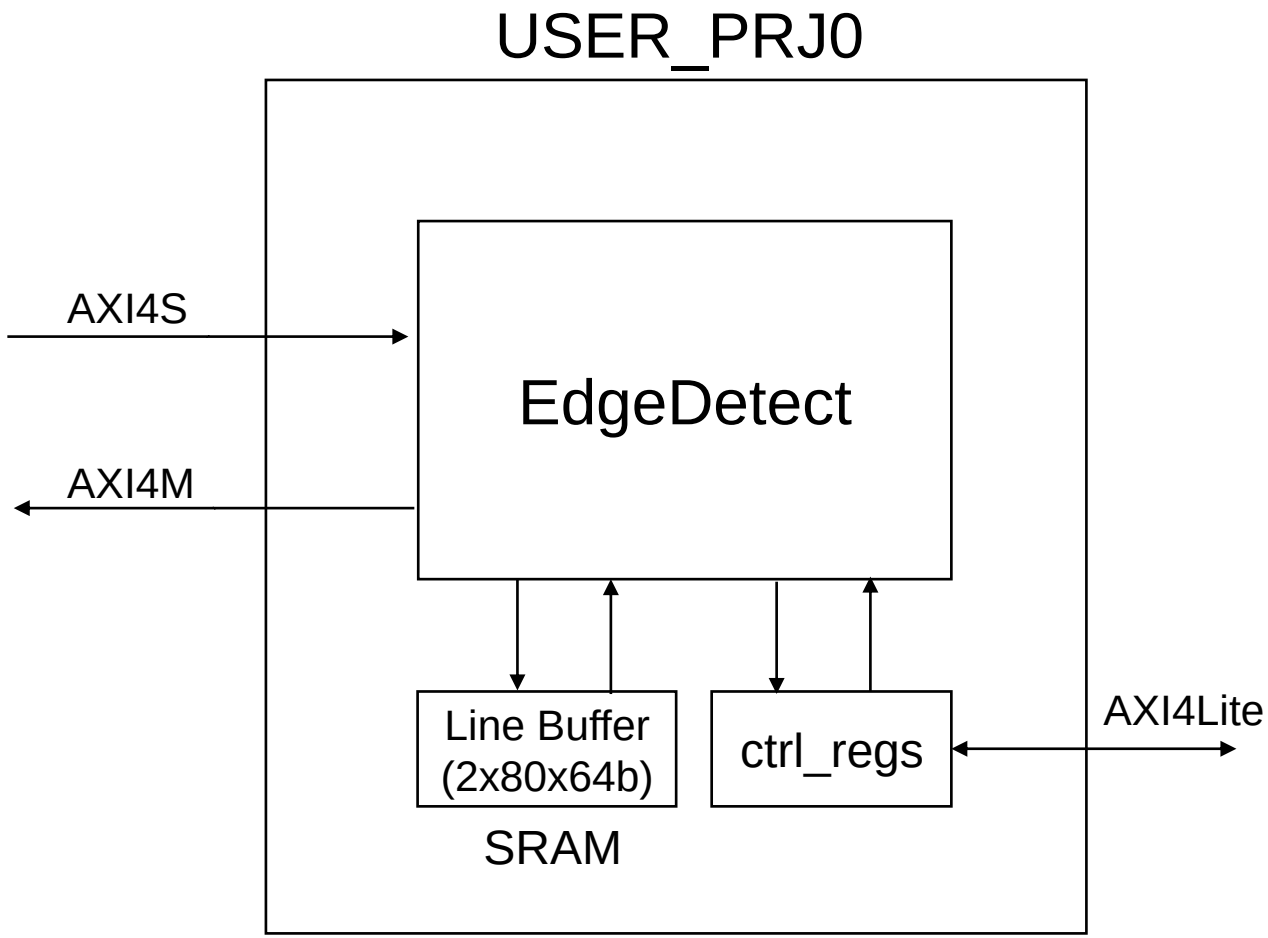
03_fsic_prj

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



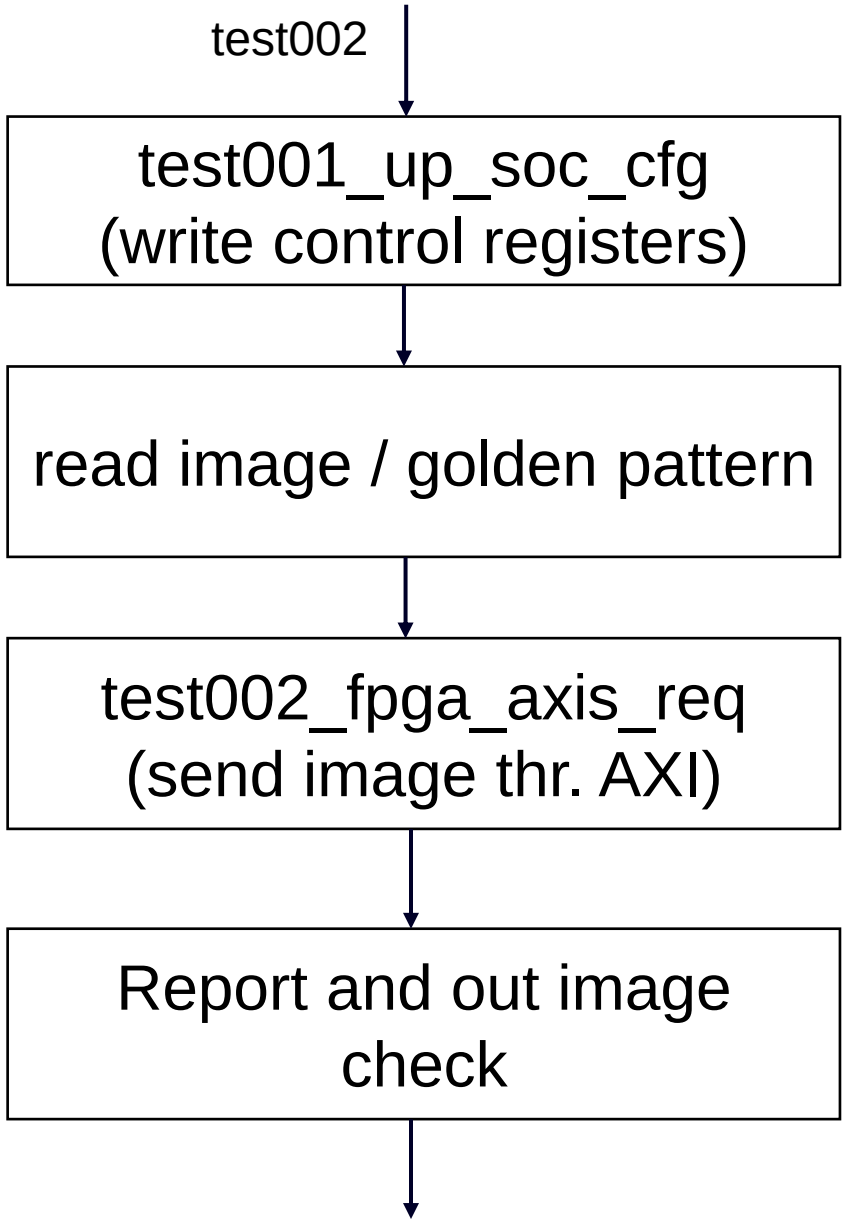
FSIC Testbench

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 &

SIEMENS