THE HONG KONG POLYTECHNIC UNIVERSITY
香港理工大學

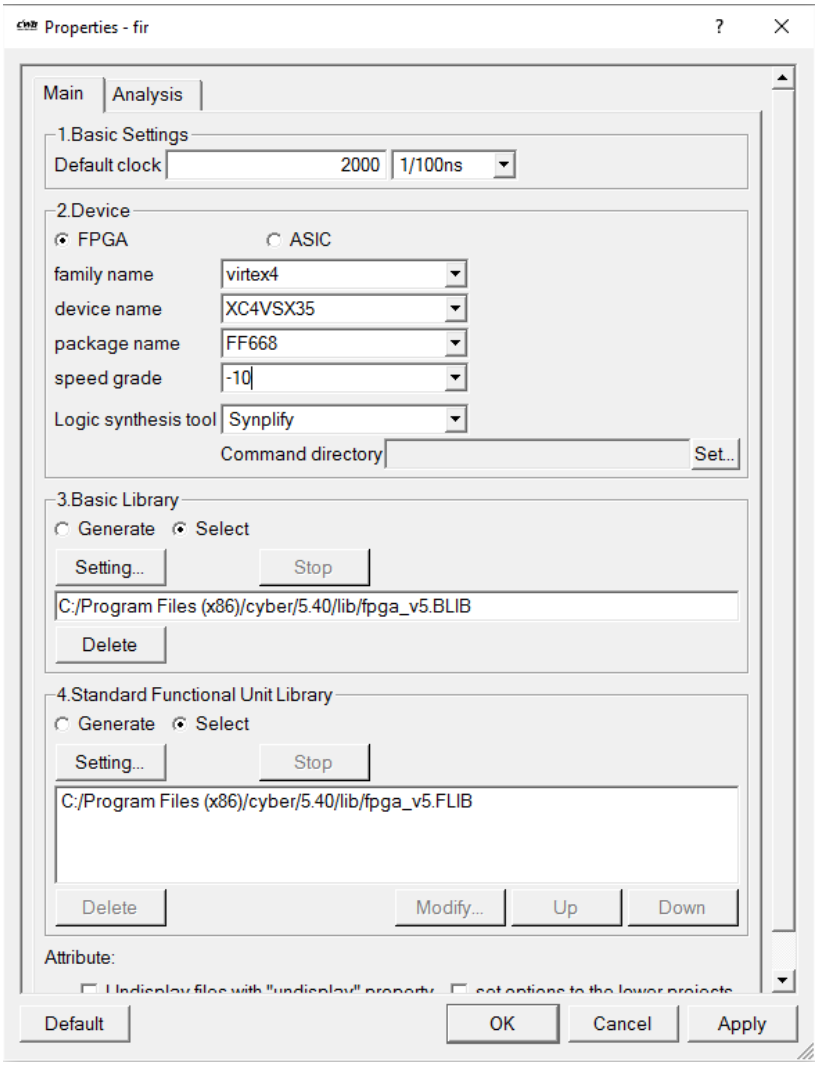
EIE4110 Introduction to VLSI and CAD
High-Level Synthesis (HLS) Laboratory

Laboratory Objectives

- Learn how to use a commercial HLS tool
- Understand how to re-write pure ANSI-C software descriptions into C or HW
- Perform the 3 main HLS steps using the commercial HLS tool:
    - Resource allocation
    - Resource Scheduling
    - Binding
- Verify that the design works as expected
    - Learn how to use the automatic testbench generator
    - Create input stimuli files
    - Simulate using the cycle-accurate model generator
    - Peform a simulation using the same input vectors as for the SW verification for the RTL sim
- Perform manual design space exploration

Creating a new empty project

1. Open CWB (either clicking on the CWB icon on the desktop or on linux %cwb &).
2. Create a new project and workspace called fir (File→New)
3. Set the project parameters as shown in the figure:



The main items that need to be specified are:

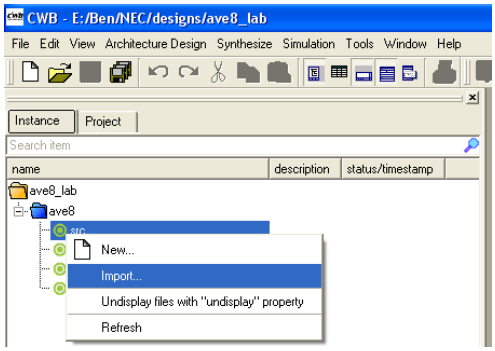
- Clock period (2000 x 1/100ns = 20ns ⇔ 50 MHz)
- Target device (ASIC or FPGA – altera or Xilinx) → In this case a Virtex 4 FPGA
- Logic Synthesis tool → ISE
- Libraries which contain the area and delay information of the basic operations
    - BLIB (Basic Library): contains the delay and area information of basic logic gates (AND, OR,etc..) and muxes of different sizes
    - FLIB (Functional Unit Library): Contains the area and delay information of Functional units of different sizes (e.g. 2-4-6-8-12 bit adder)

4. Click on Apply to generate the empty project

NOTE: If no Virtex 4 libraries are available use Virtex 5.

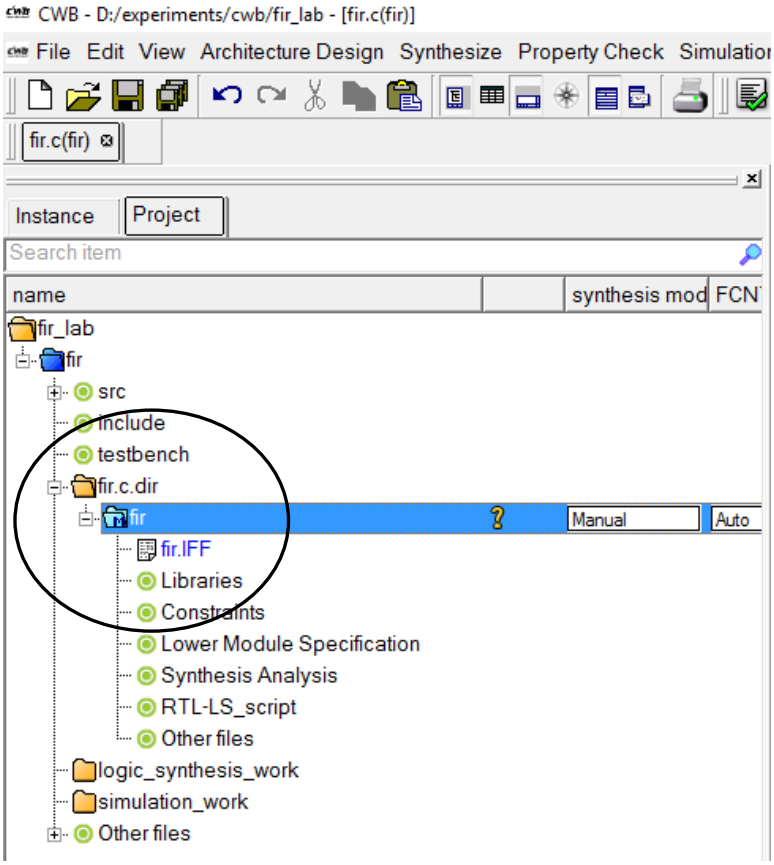
Editing or Inserting a ANSI-C description for synthesis

1. Right click on source → Import → fir.c



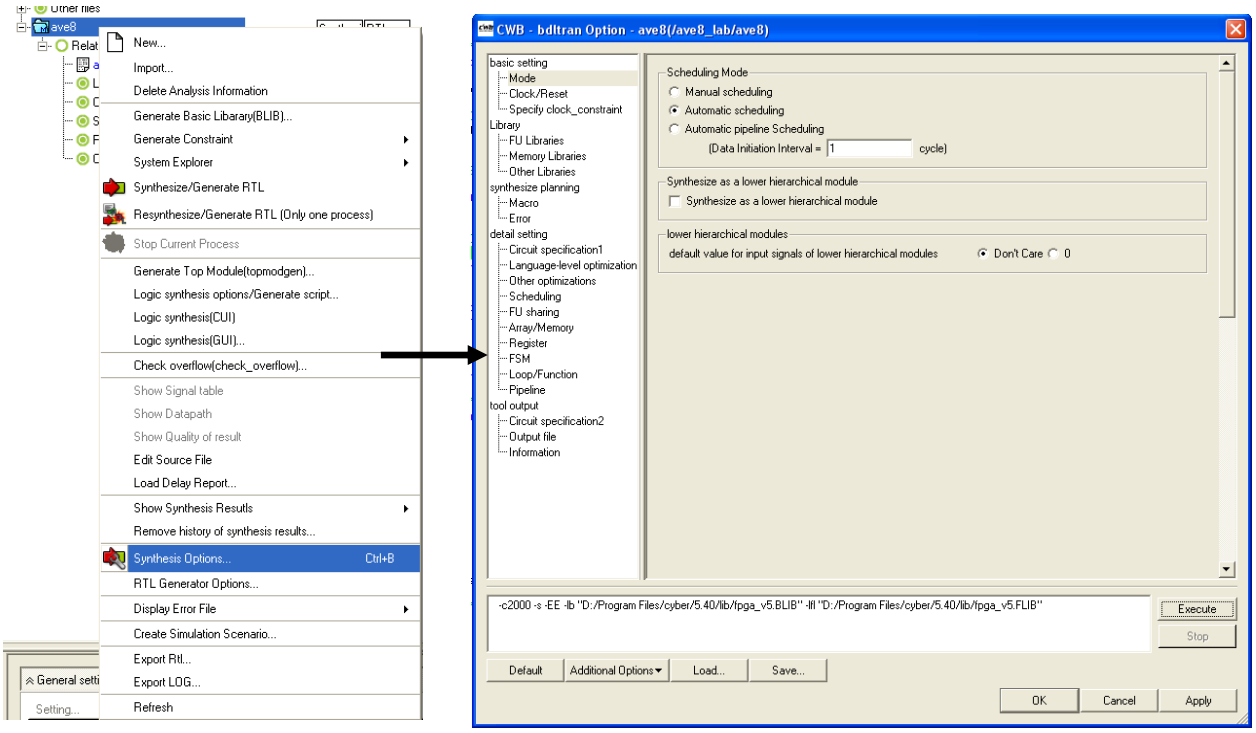
2. Modify the C code just imported to make it synthesizable. Look at the source code for clues.

3. Parse source code to check if there are any syntax errors. The console window will indicate if any erros have occurred and where. A 'light-blue' folder with an .IFF file will be generated if the parsing was successful



Setting up Synthesis mode and other constraints

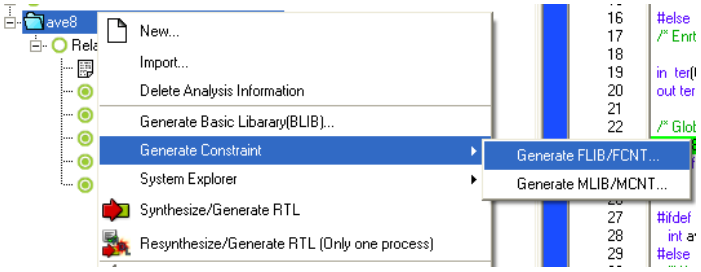
1. Right click on ‘light-blue’ folder → Synthesis options



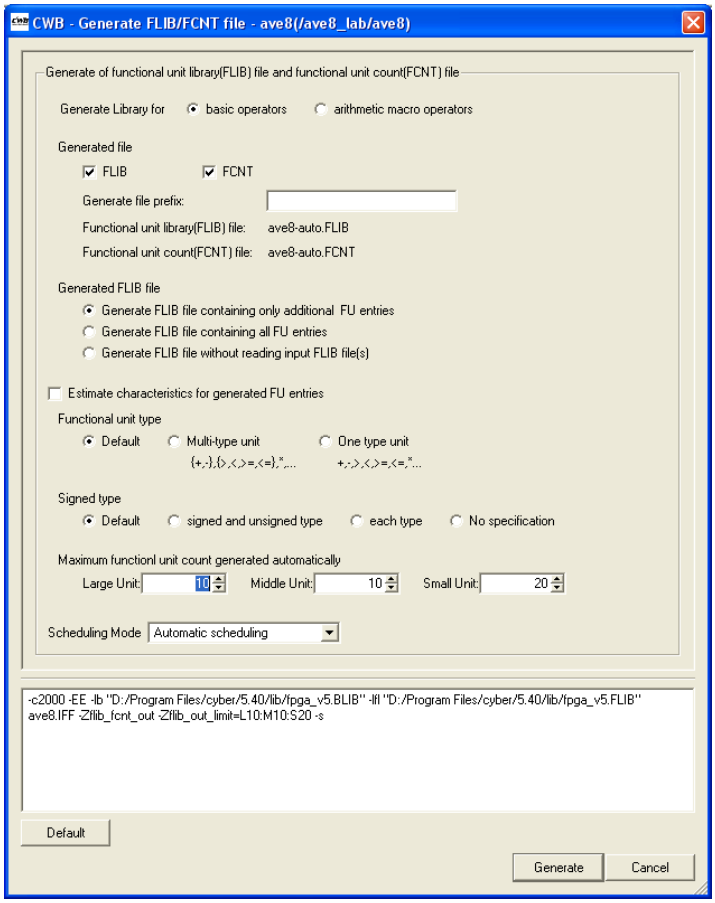
2. Set the scheduling mode to ‘Automatic scheduling’. This implies that CWB will automatically time the C description at the scheduling phase. Manual scheduling implies that the user will manually time the description using the ‘$’ sign as a clock boundary.
3. Other synthesis options that control the synthesis process can also be set here as well as the target clock frequency.
4. Click on ‘Apply’ and OK to accept the changes.

Create the FU constraint files (Resource Allocation)

1. Righ-click on ‘light-blue’ folder → Generate Constraint → Generate FLIB/FCNT to open a dialog window that allows the generation of the resource constraint file



2. Increase the ‘Large Unit’ field to 10 as shown in the dialog window.



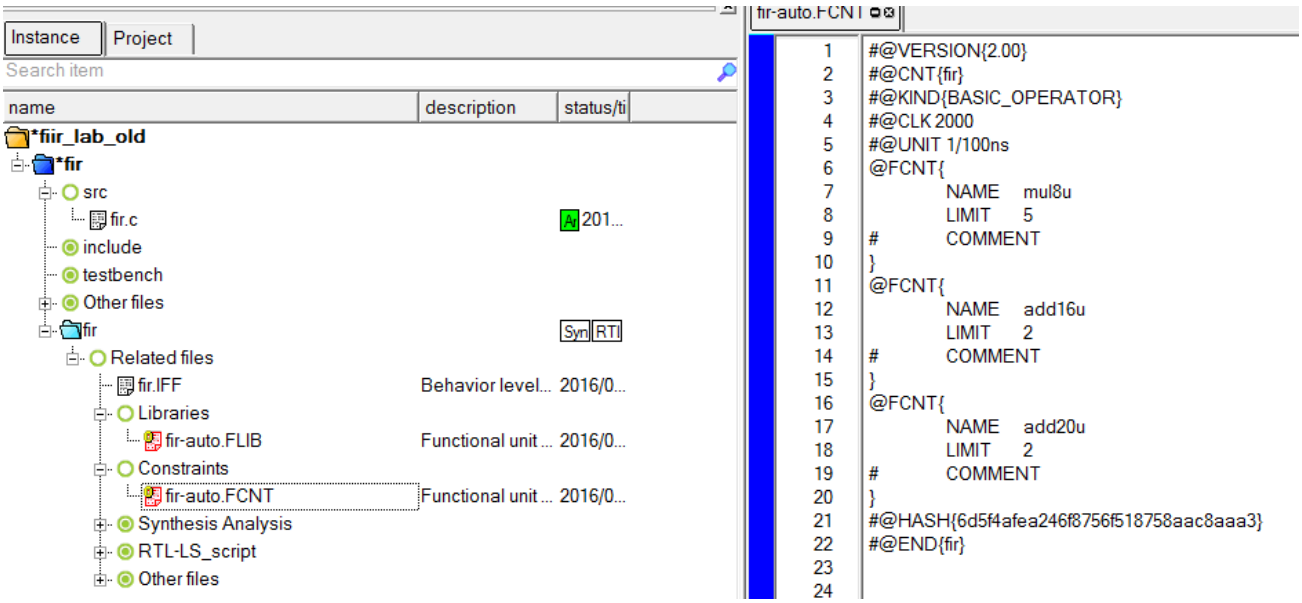
CWB considers FUs of different bws as Large (<=8 bits) , middle (8bits<=4 bits) and small (>4 bits). By default it limits the number of FUs allocated as indicated in the dialog window to 5:10:20

3. Click ‘Generate’ . Two new files will be generated:

   fir-auto.FLIB: FU Library file containing FUs area and delay of FUs not contained in the initial FLIB/BLIB files (empty in most cases like in this)

   fir-auto.FCNT: Constraint file limiting the number of FUs tha the synthesizer can instantiate in paralle.

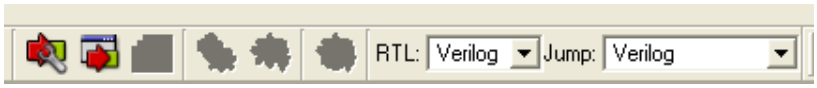
Review the contents of the files by ‘right-click → open by internal editor



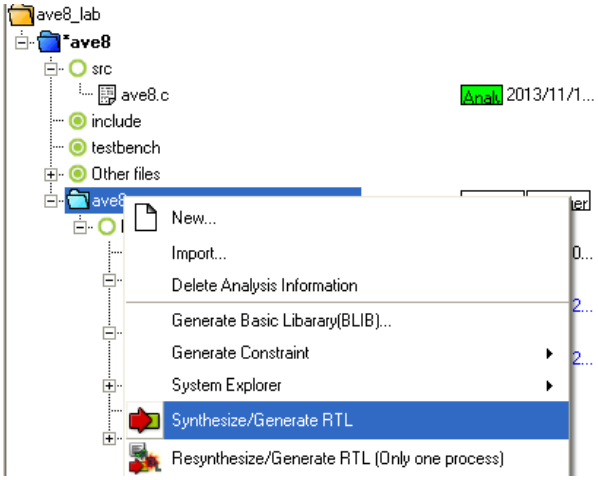
High-Level Synthesis

The conversion of the C program into RTL (Verilog or VHDL) can now take place:

1. Select Verilog at the toolbar

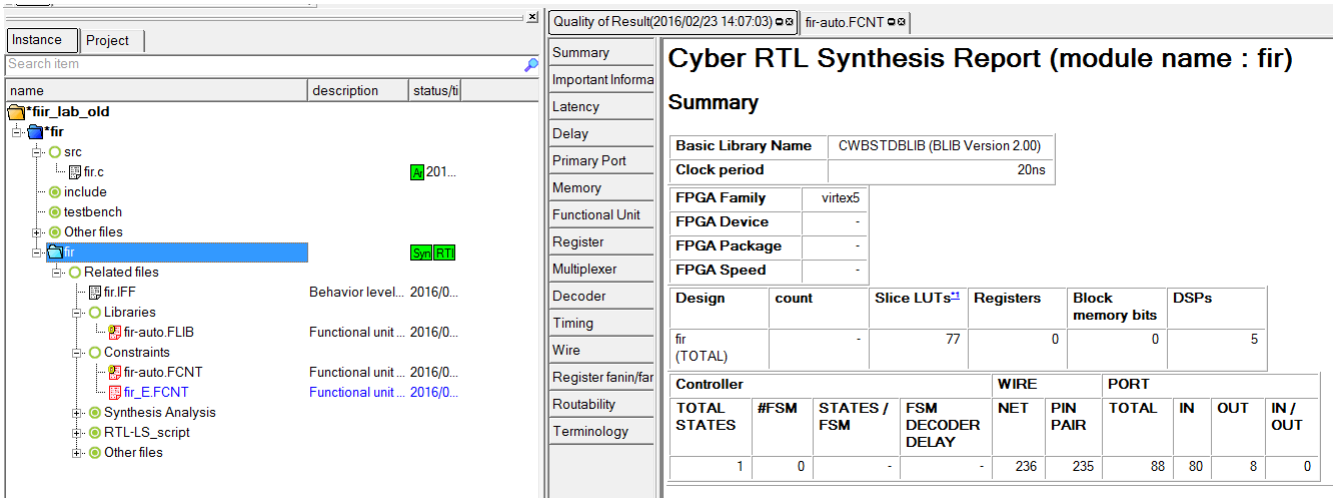


2. Righ-click on ‘light-blue’folder → Synthesize/Generate RTL (or toolbar)



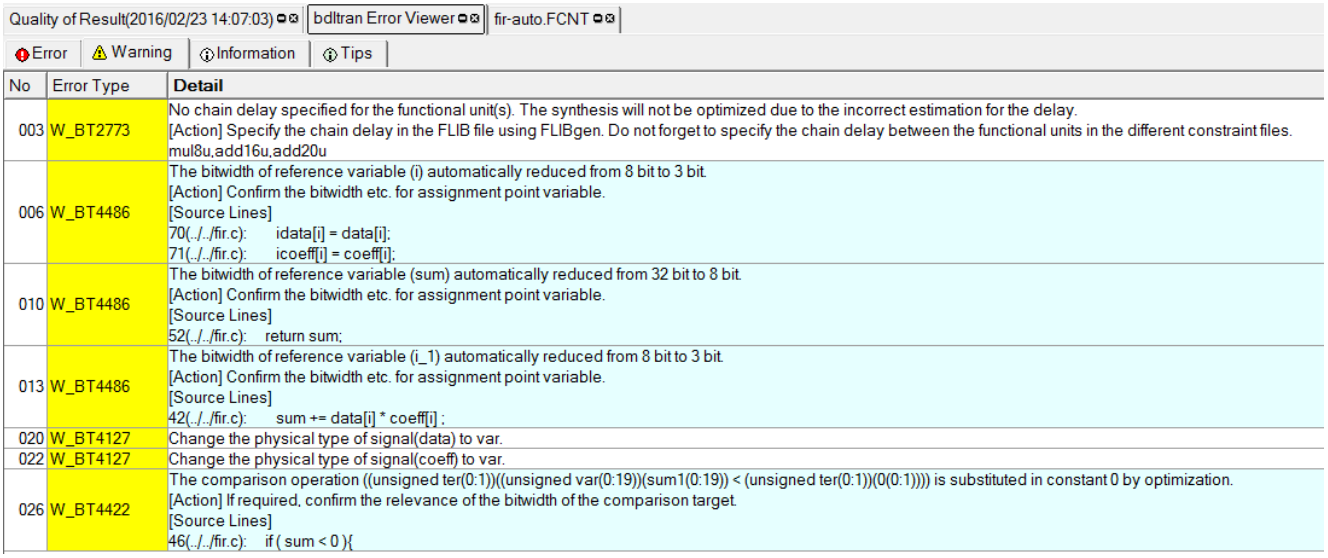
A report file (Quality of Results-QoR) will be generated and appears automatically. It contains all the synthesis information: FPGA resources used, critical path, design latency, etc…

> **NOTE:** The information reported is based on the FLIB and BLIB file provided. It is therefore important to provide the correct library files or to re-generated these if a different device is targeted.
> Also, the reported data should be confirmed performing a logic synthesis and a simulation.



Other report files and information files are also created (.err, .SUMM, .tips). The .err files contains any possible synthesis error, warnings and tips to improve the synthesis results.

3. Open fir.err file and review the warning, information and tips messages

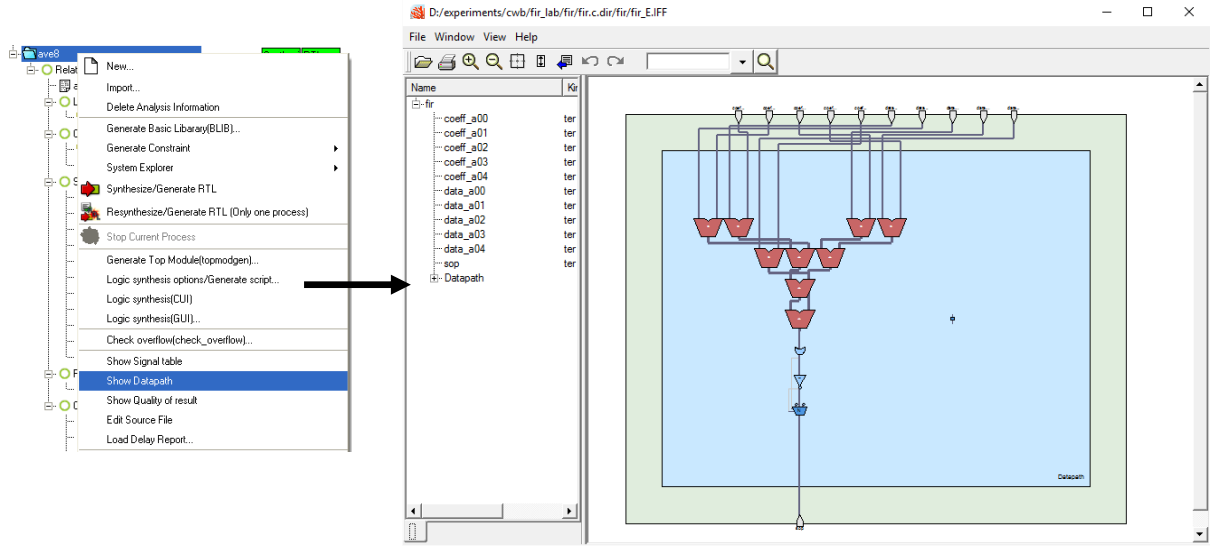


No	Error Type	Detail
003	W_BT2773	No chain delay specified for the functional unit(s). The synthesis will not be optimized due to the incorrect estimation for the delay. [Action] Specify the chain delay in the FLIB file using FLIBgen. Do not forget to specify the chain delay between the functional units in the different constraint files. mul8u,add16u,add20u
006	W_BT4486	The bitwidth of reference variable (i) automatically reduced from 8 bit to 3 bit. [Action] Confirm the bitwidth etc. for assignment point variable. [Source Lines] 70(../../fir.c): idata[i] = data[i]; 71(../../fir.c): icoeff[i] = coeff[i];
010	W_BT4486	The bitwidth of reference variable (sum) automatically reduced from 32 bit to 8 bit. [Action] Confirm the bitwidth etc. for assignment point variable. [Source Lines] 52(../../fir.c): return sum;
013	W_BT4486	The bitwidth of reference variable (i_1) automatically reduced from 8 bit to 3 bit. [Action] Confirm the bitwidth etc. for assignment point variable. [Source Lines] 42(../../fir.c): sum += data[i] * coeff[i] ;
020	W_BT4127	Change the physical type of signal(data) to var.
022	W_BT4127	Change the physical type of signal(coeff) to var.
026	W_BT4422	The comparison operation ((unsigned ter(0:1))((unsigned var(0:19))(sum1(0:19)) < (unsigned ter(0:1))(0(0:1)))) is substituted in constant 0 by optimization. [Action] If required, confirm the relevance of the bitwidth of the comparison target. [Source Lines] 46(../../fir.c): if (sum < 0){

Review of Synthesis Result

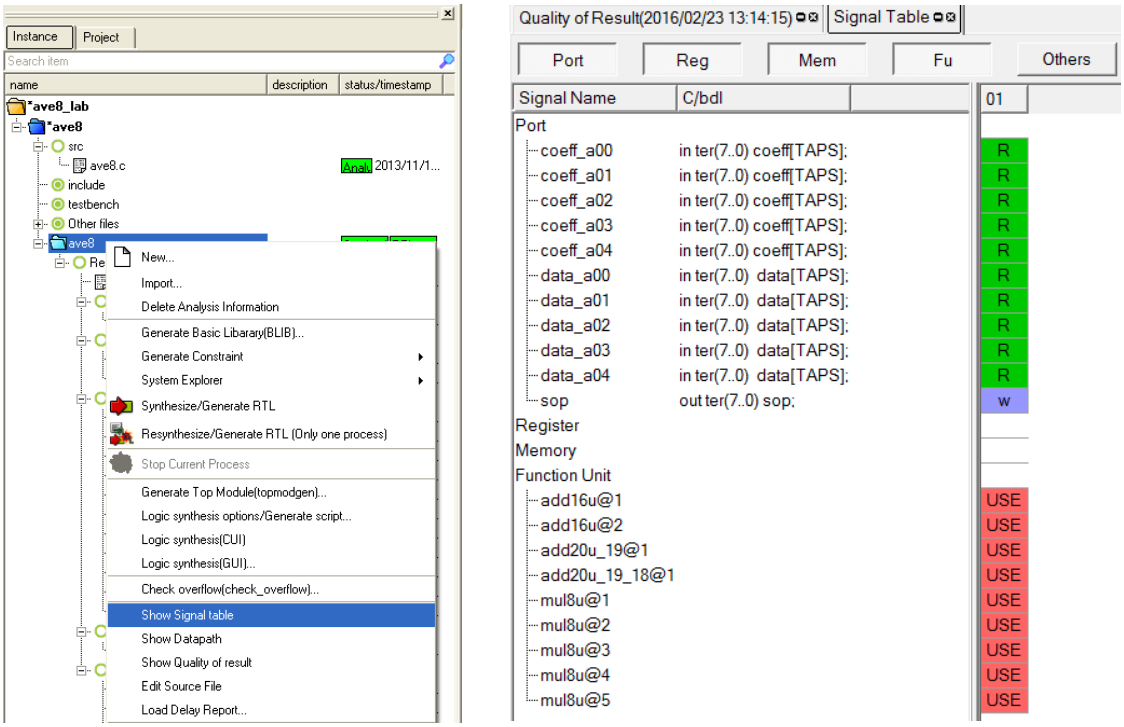
The synthesis result can be reviewed in different ways.

1. Schematic View: Right-click ‘light-blue’ folder→ Show Datapath



This opens the CWB’s datapath viewer. Clicking on any part of the schematic will also highlight the source code in CWB that corresponds to it (cross referening)

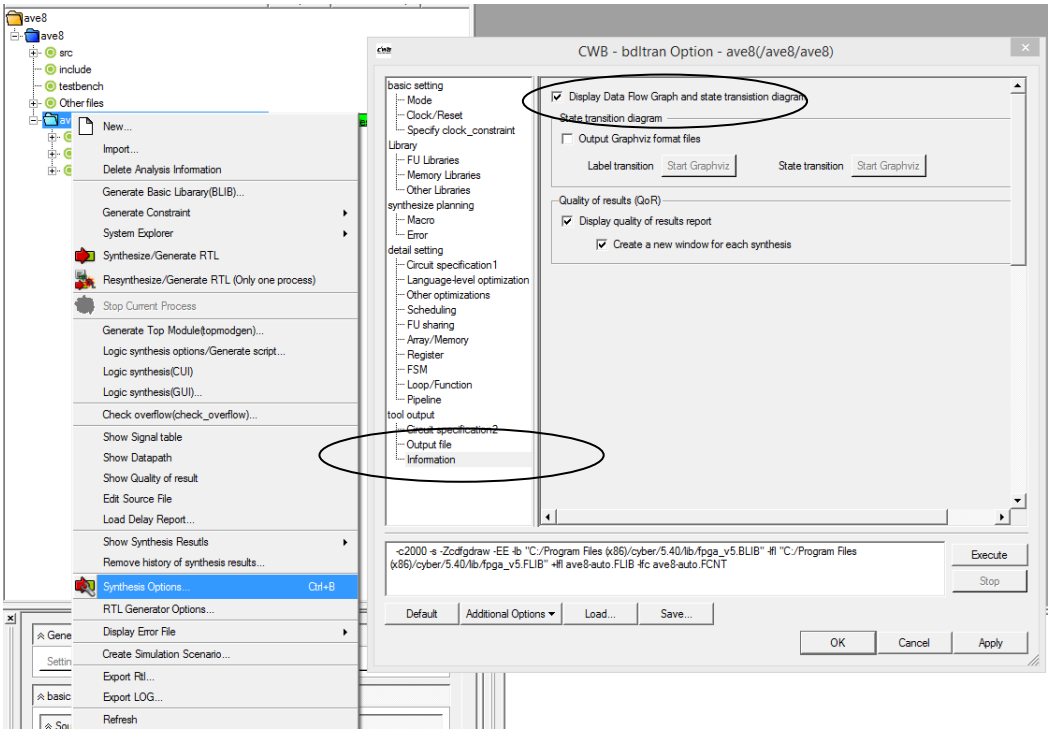
2. Signal Table: Right-click ‘light-blue’ folder→ Show Signal Table



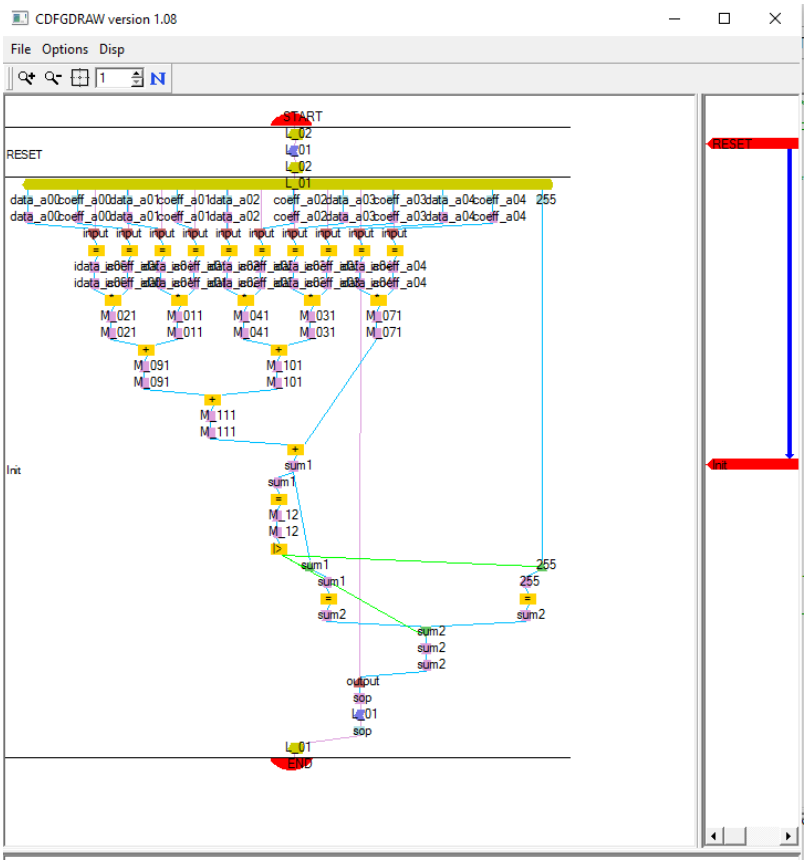
The Signal table shows the timing diagram of the synthesis. When inputs and outputs are being read or written and when registers accessed

Scheduling Result

You can review the scheduling result, by enabling a synthesis option to show the scheduled of the synthesized circuit. In Information →select Display Data Flow Graph



Re-synthesize the circuit. The CDFG will be displayed. The synthesis will continue after the window is closed.

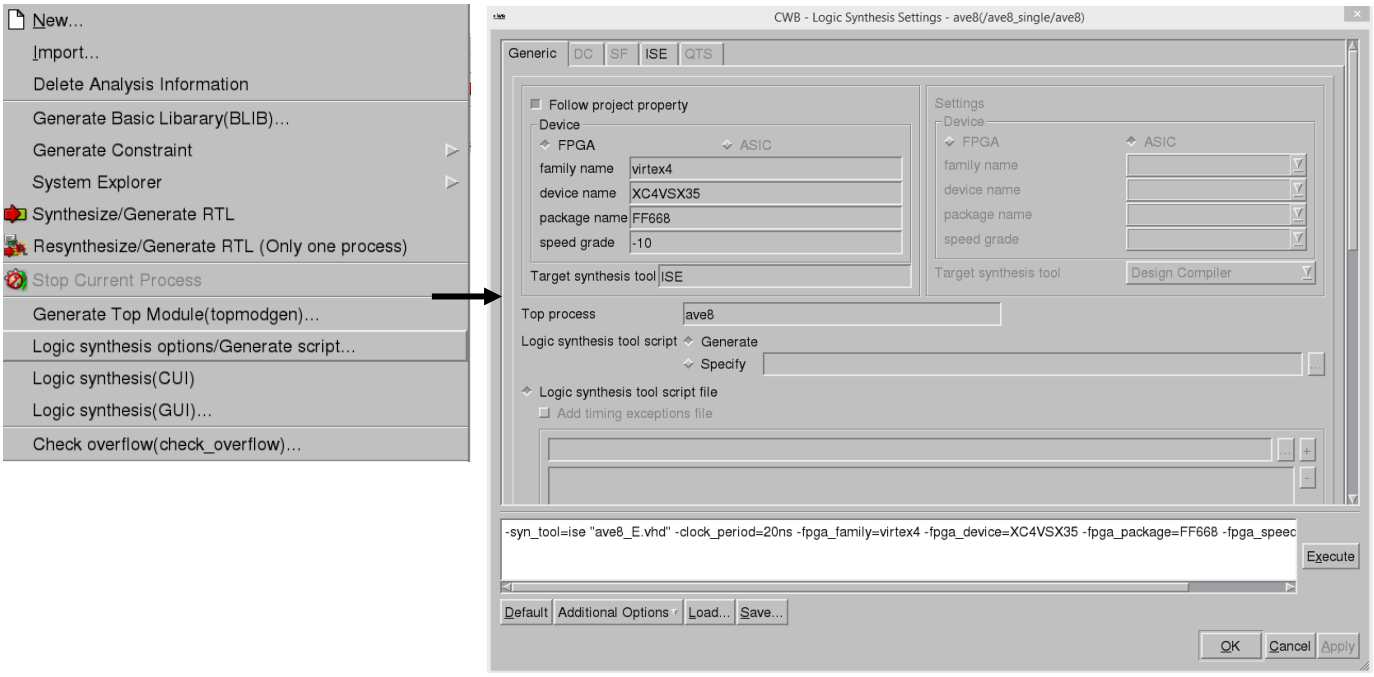


Logic Synthesis

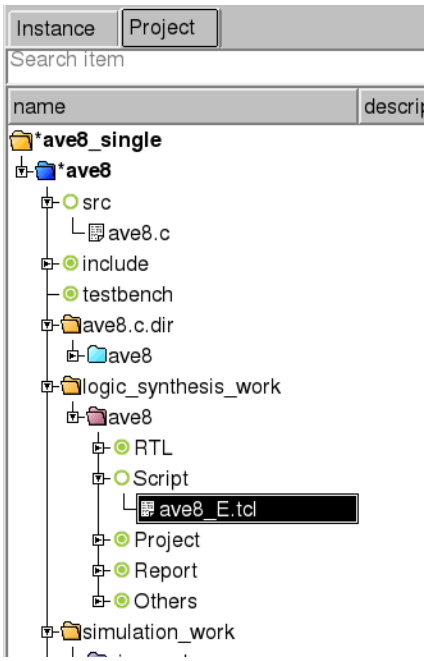
It is now possible to call Altera's or Xilinx's logic synthesizers from within CWB. For this, first we need to create a synthesis script for the target FPGA vendor. Then the logic synthesizer can be executed by either opening its IDE (GUI) or from the command line interface (CUI)

1. Create Logic synthesis script: Right click on light blue process folder → Logic synthesis options/Generate script →Execute

   This creates

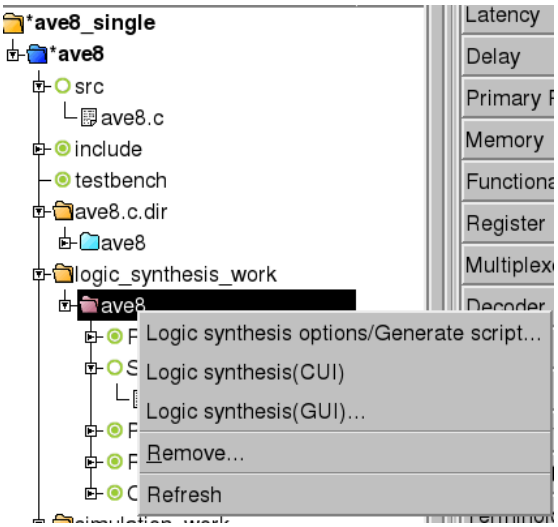


This will create a .tcl script file for Xilinx ISE. The newly generated red folder includes all the information regarding the logic synthesis



Note: You might need to set up the path for the Xilinx tools in Tools→ Options→Tool Path

2. Logic Synthesis: Right clicking on the red folder → Logic synthesis (CUI) will launch Xilinx ISE in command mode. ISE will start and the ISE outputs will be displayed on CWB's console window



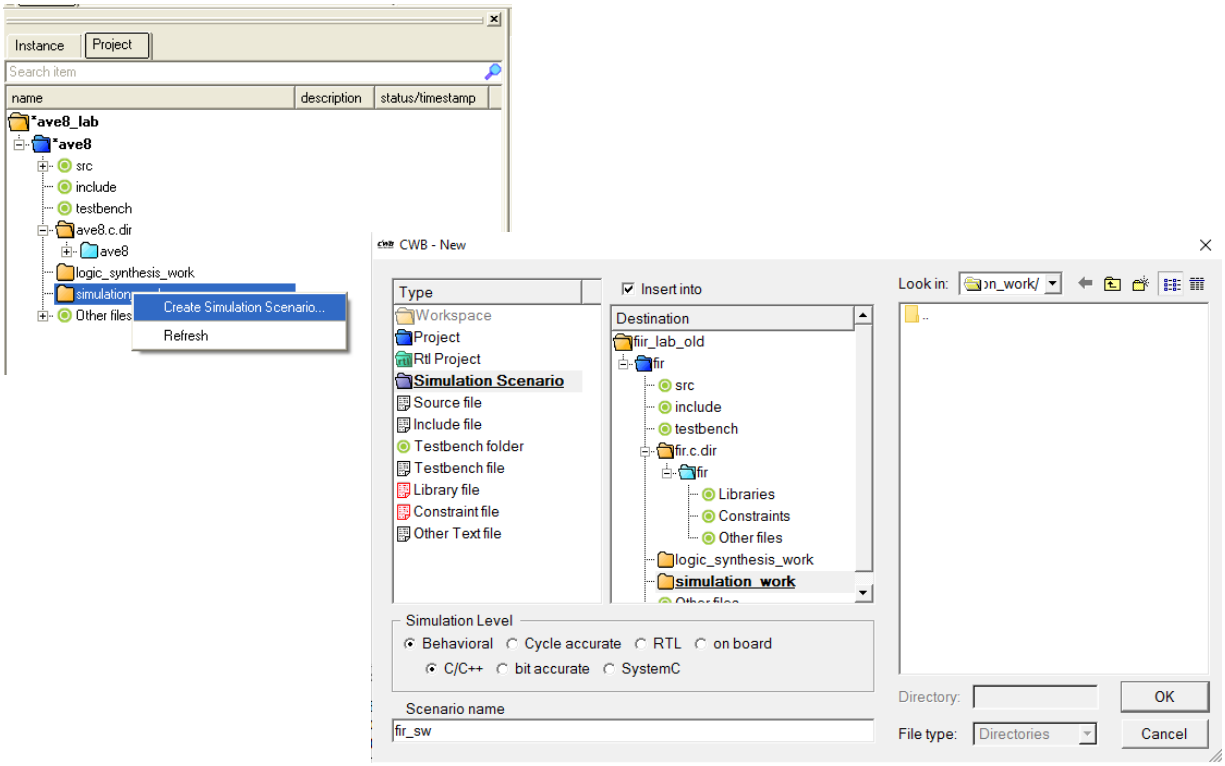
Once ISE finishes, the logic synthesis results can be observed at the report folder.

Design Verification: Software Simulation

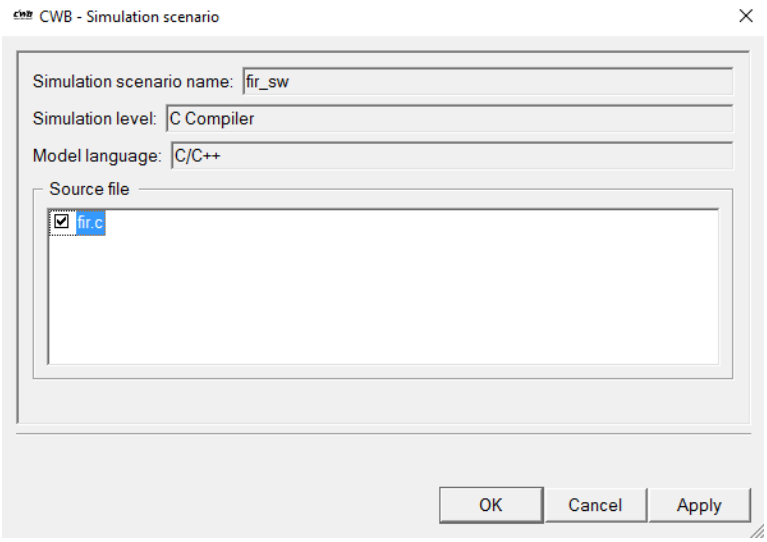
There are different levels of simulations that can be used to verify the design:

- Pure SW simulation: normal SW simulation
- Behavioral simulation: untimed simulation (like SW), but considering the HW data types (ter, var, reg)
- Cycle-accurate simulation : timed simulation of the scheduled synthesis results
- RTL simulation

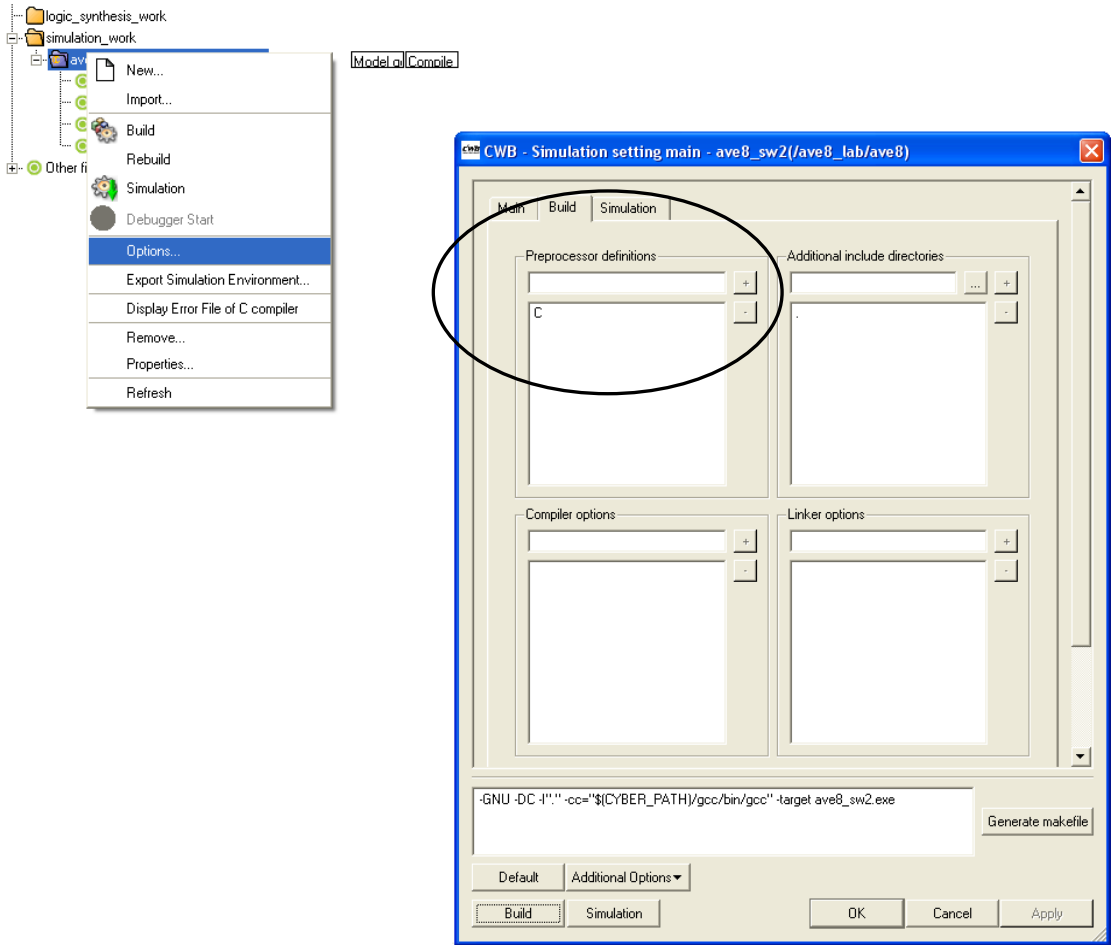
1. The first step is to do a normal SW simulation from within CWB. Chane to the 'Project' tab → Create Simulation Scenario.



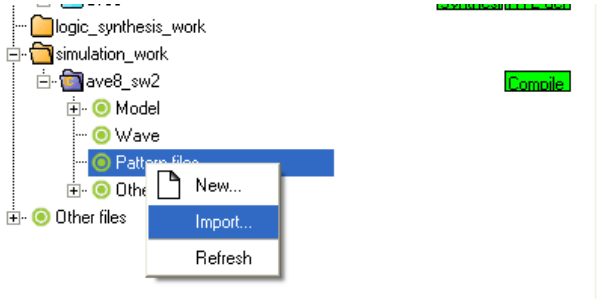
2. Select "Behavioral" and C/C++ and call the scenario name 'fir_sw'
3. Select the top process (check the check box)



4. Right-click on ‘fir_sw” folder → options→ Build tab -> Add C to the preprocessor definitions



5. Add the indata.txt input stimuli file to the simulation enviornment



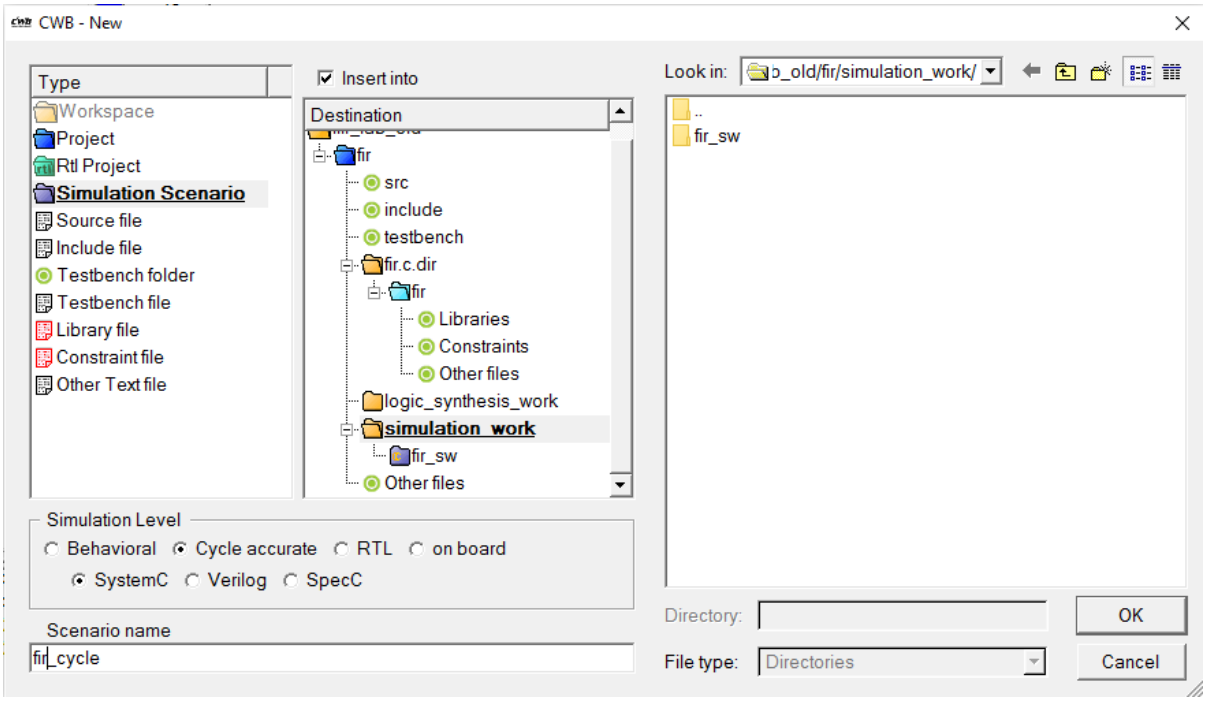
6. Build the binary (.exe file) and simulate it

The sop.txt file with the SW simulation values will be generated. This will be our ‘golden’ output against which the simulation results throughout the different synthesis stages will be compared against.

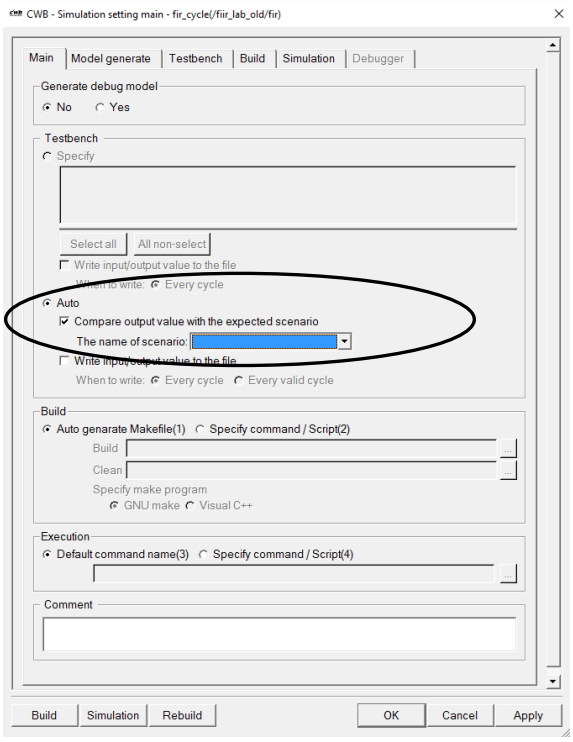
Cycle-Accurate simulation

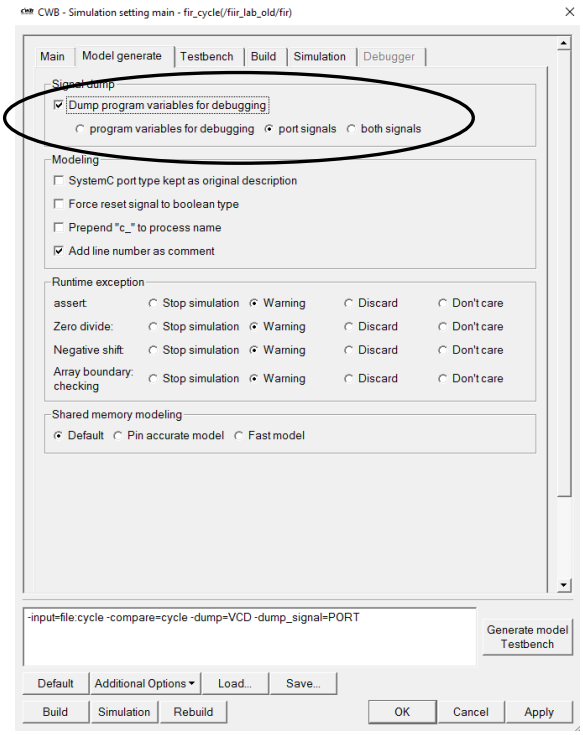
In order to verify the corretnes of the timing and also the accurate performance of the synthesized designs two options are available: (1) simulate the generated RTL code or (2) created a cycle-accurate model. This second options is normally prefrerred as the simulation is faster than RTL.

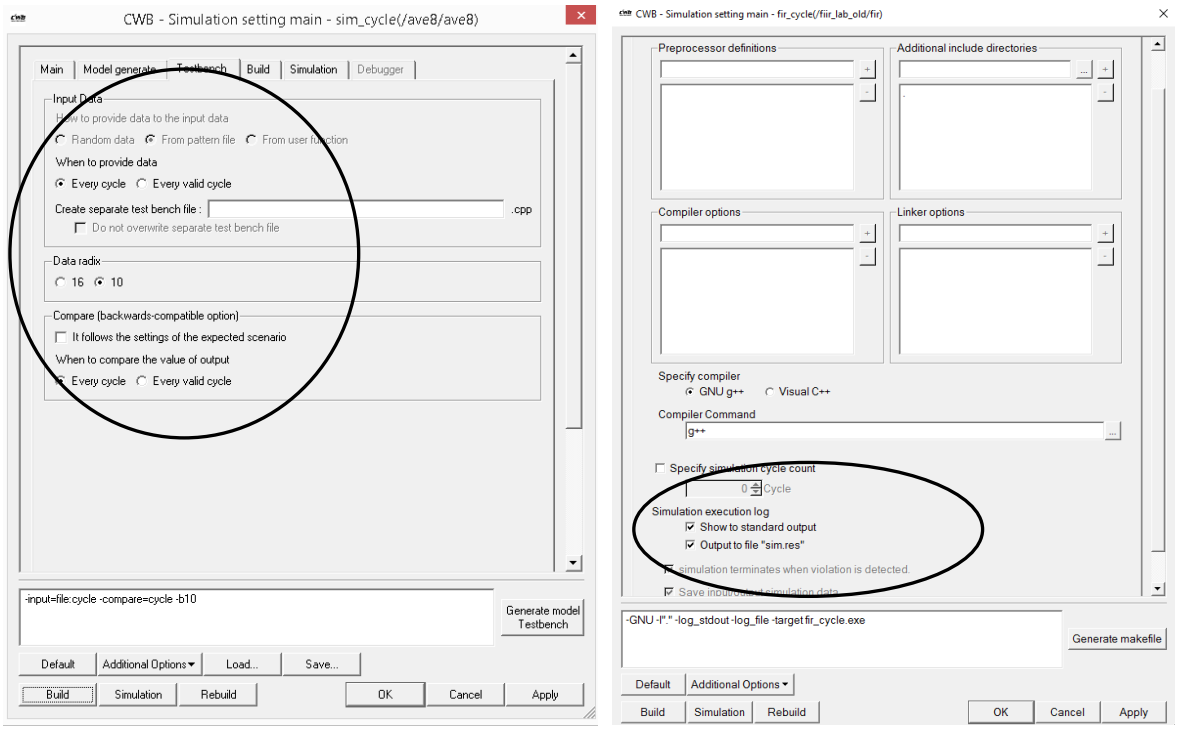
1. Create a new simulation scenario (right-click on the ‘simulation work’ folder. Select ‘Cycle accurate’ options and enter scenario name (fir_cycle)



2. Spefy testbench options (right click→ options)



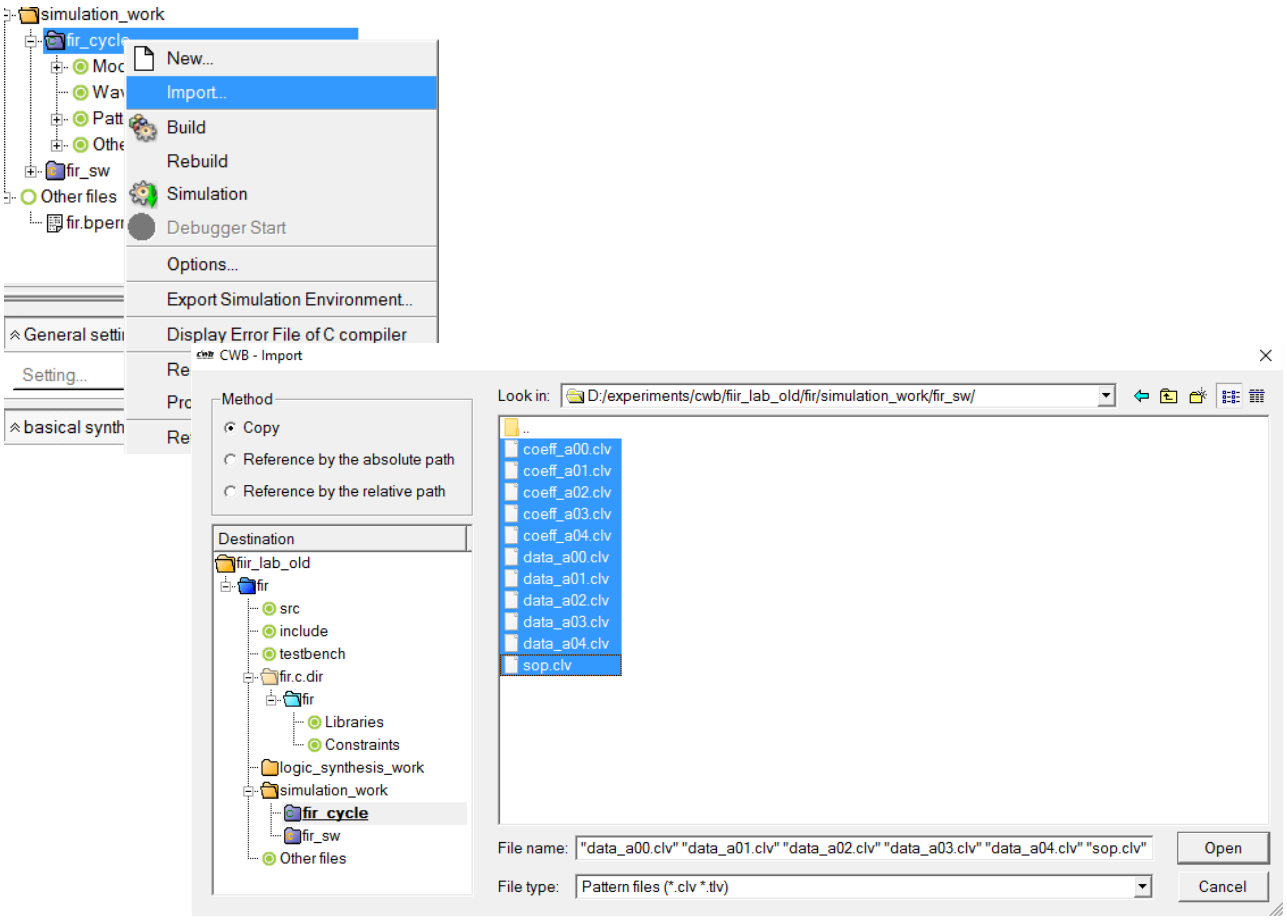




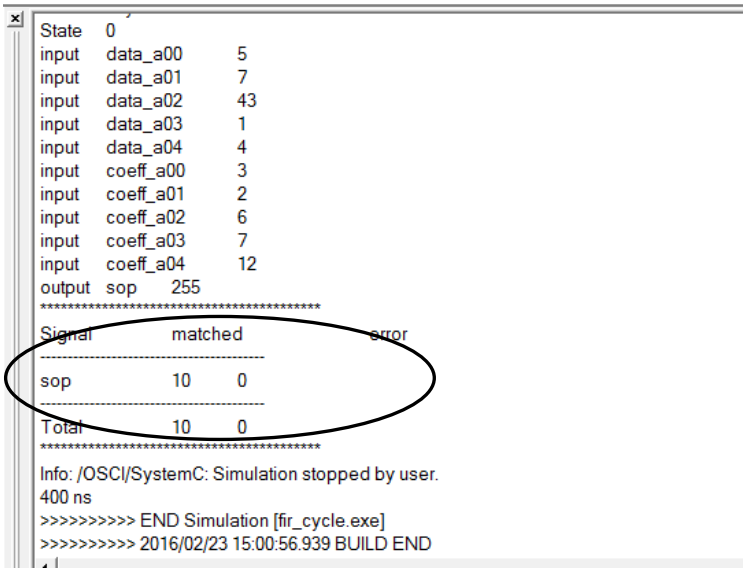
3. The input test vector files and golden output file generated in the pure software simulation needs to be renamed and manually edited. In CWB each input port needs its own file with its name and with the extenstion .clv. Moreover, because the input ports are declared as arrays, these are expanded automatically, the indata.txt and coeff.txt need to be expanded as follows:

File name in Software Simulation	File name in Cycle-accurate Simulation
indata.txt	indata_a00.clv indata_a01.clv indata_a02.clv indata_a03.clv indata_a04.clv
coeff.txt	coeff_a00.clv coeff_a01.clv coeff_a02.clv coeff_a03.clv coeff_a04.clv
sop.txt	sop.clv

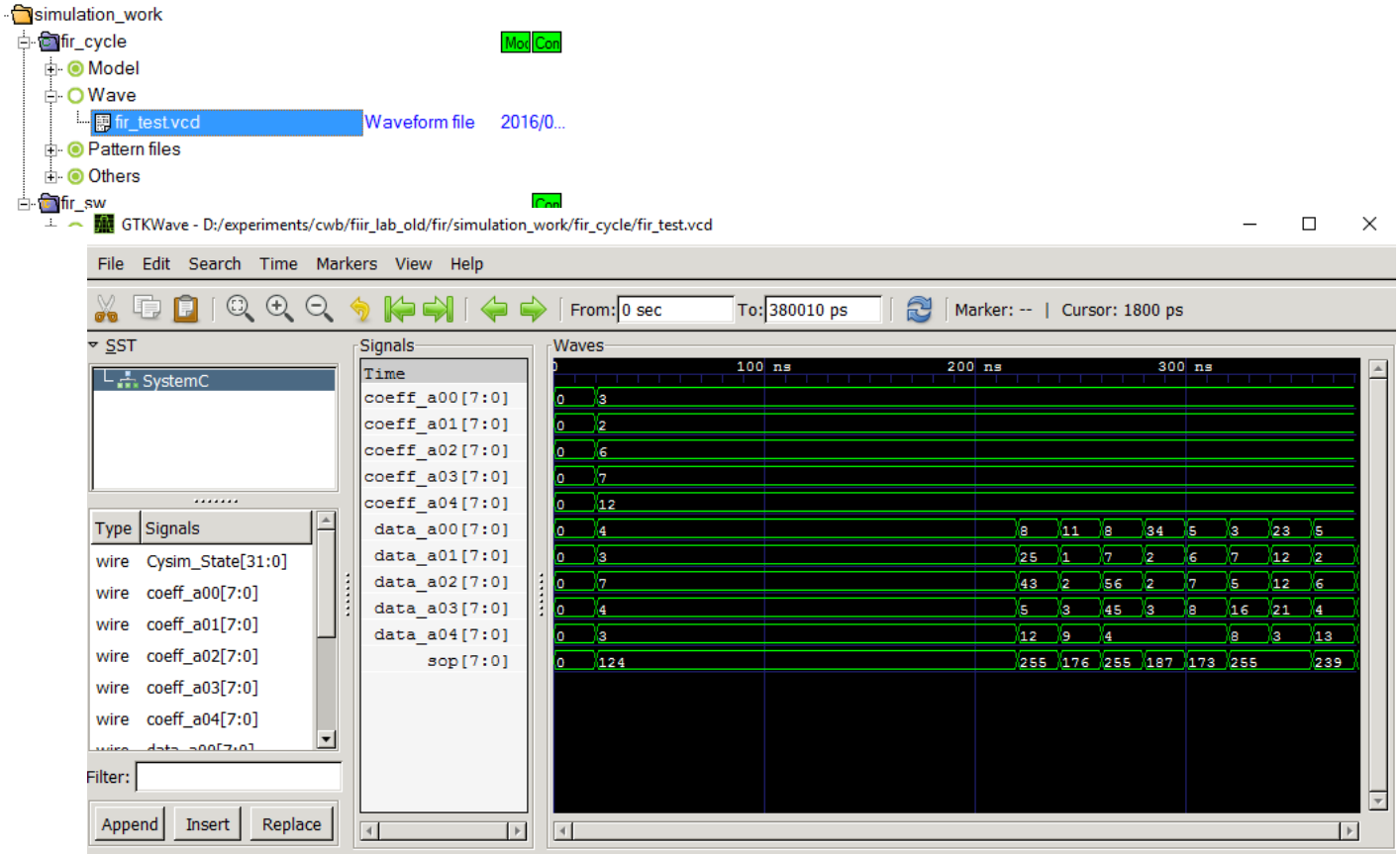
4. The SW testbench was generated in such a way that these 11 files were automatically generated. Thus import these to the fir_cycle folder as follows. Right click on fir_cycle folder → Import → Select .clv file generated during the pure software simulation in the fir_sw folder



5. Build and simulate the model. The console window will display if the outputs from the cycle-accurate simulation match the outputs from the SW simulation



6. A .vcd file which can be opened by a waveform viewer (e.g. GTKwave) is generated. Double click on it to open it.

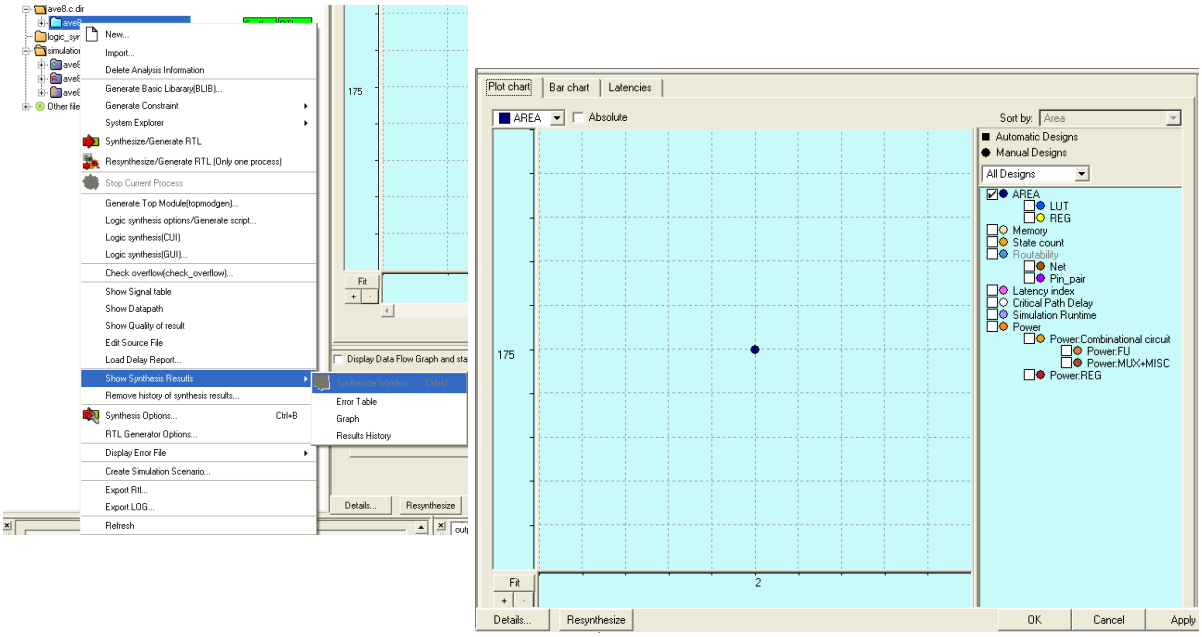


The simulation output should match the result of the SW simulation.

Design Space Exploration

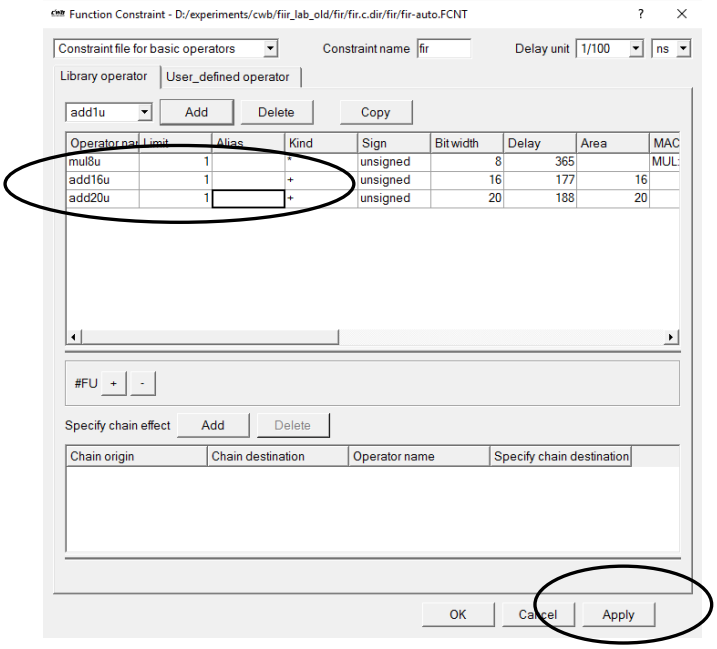
C-Based design allows the generation of different architectures with differen area vs. performance constraints without having to modify the actual C code. This is mainly done by modifying the synthesis constraints. E.g. FCNT constraint file or synthesis options.

1. Open the synthesis window to observe the synthesis result of the current desing in the design space exploration window. Right-click on ‘light-blue’ folder→ Show synthesis results→ Synthesis window

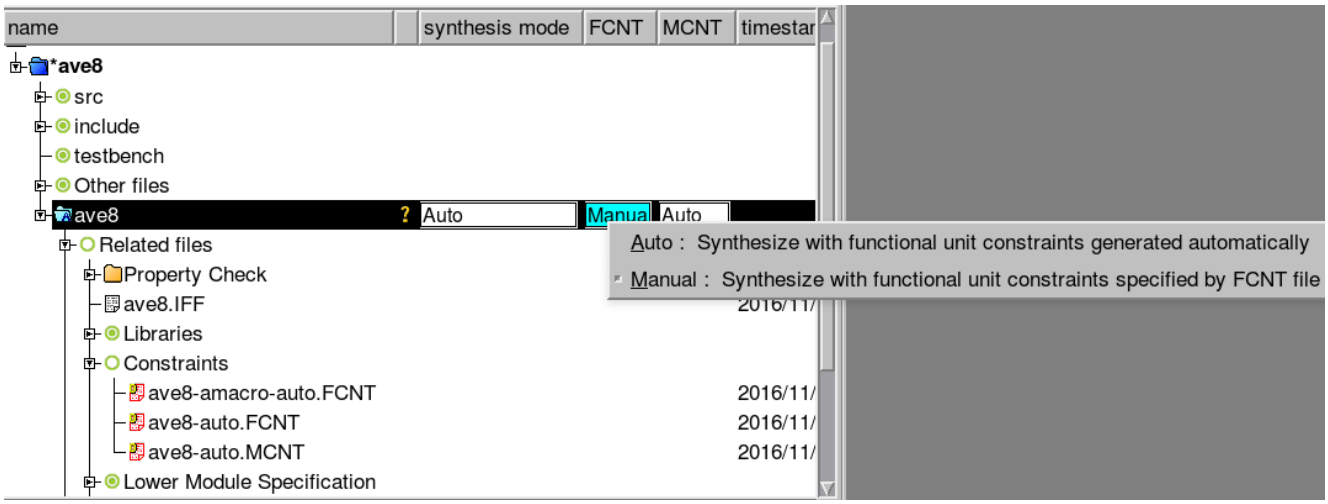


The Y-axis represents the area of the design, while the y-axis can be modified to represent different things. Modify to ‘Latency index’

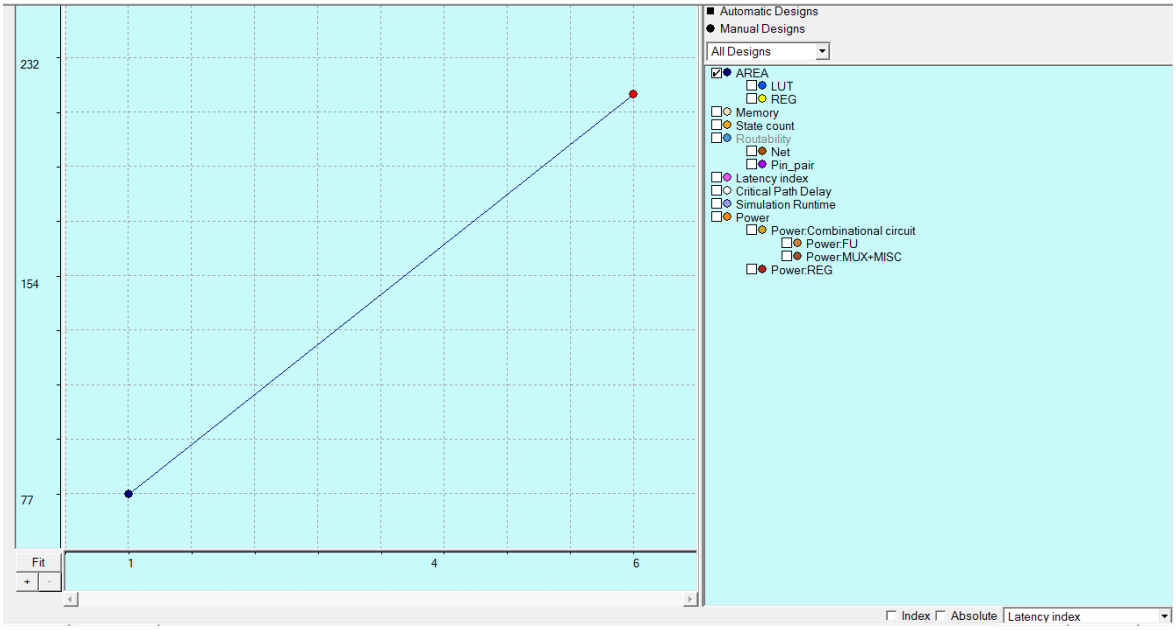
2. Open the fir-auto.FCNT resource constraint file and reduce the number of FUs to 1



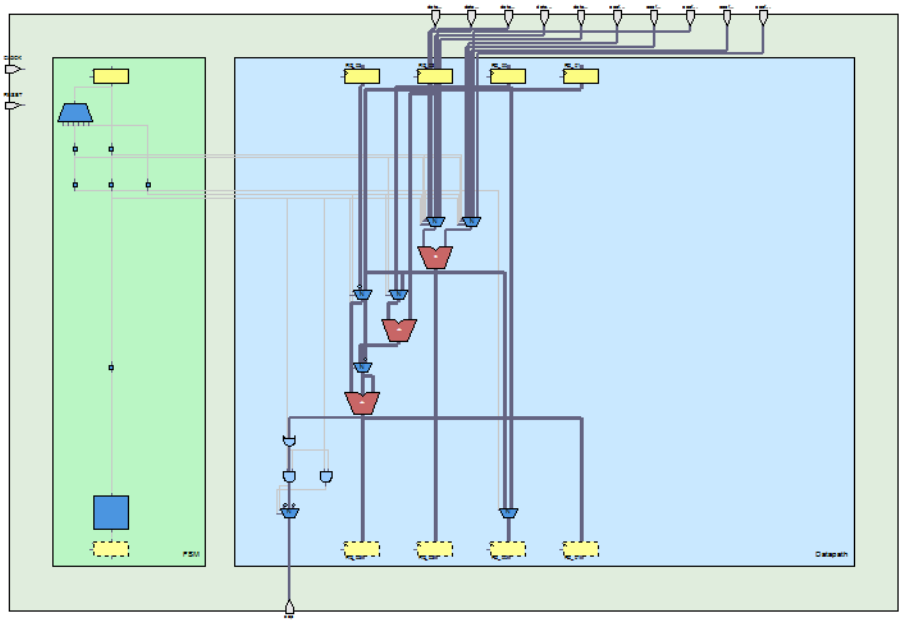
3. Apply the changes
4. Set manual FCNT generation so that CWB does not generate a new FCNT file when re-synthesizing



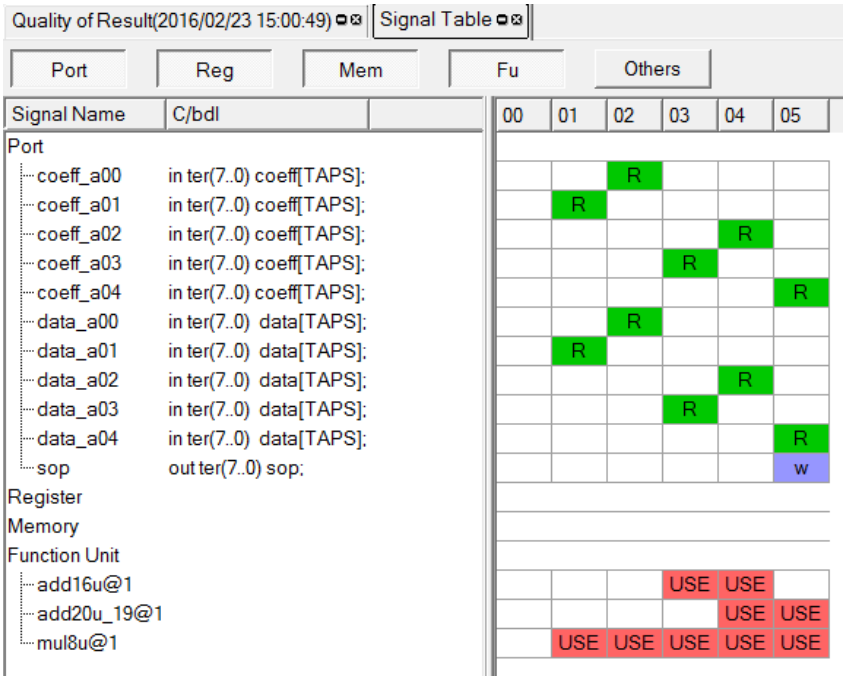
5. Re-synthesize the design. A new design with different Area and latency characteristics is generated and plotted in the synthesis window



Opening the schematic viewer confirms that only 3 FUs have been instantiated. The synthesized results maximizes resource sharing.



Opening the signal table shows that the latency has now increased from 1 cycle to 4



Re-verify the new design. No new simulation scenarios are needed. CWB will automatically re-generate the RTL, testbenches and cycle-accurate simulation models.

[END]