IL2225 LAB -1
Logic Synthesis-I

Name:

____

Personal Number:

____

Assistant:

____

Date:

____

Logic Synthesis-I

1 Objective

The purpose of this lab is to introduce the basics of logic synthesis using the Synopsys Design Compiler. Upon completion of this exercise you should be able to discuss the role of logic synthesis within the overall design process and describe each of the basic steps in logic synthesis. As shown in the figure 1, RTL code, design constraints, and the technology library files are the inputs to the design synthesis tool. Design synthesis tool maps the design on the library cells and generates the gate level netlist as the output.

Synthesis includes the following steps:

Translation: The translation of the VHDL code into technology independent logic components, like AND/NAND etc.

Optimization: Optimization of the technology independent Netlist.

Mapping: Mapping of this technology independent Netlist on a target technology.

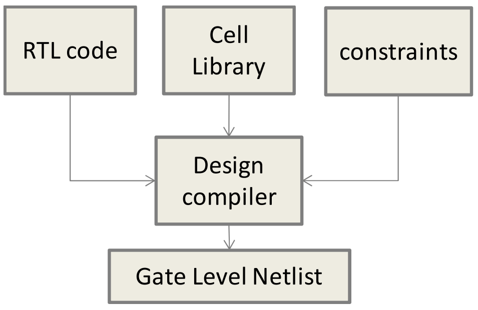


Figure 1. Logic Synthesis

In this lab, we implement a FIR filter of order N=13 in VHDL. The equation to implement a FIR filter is given as:

$$y_n = \sum_k x_{n-k} \bullet c_k$$

Where *x* is the sample vector, also known as the *delay line* since it preserves the previous *k-1* delayed samples. *c* is the impulse response of the filter, also known as coefficients.

A new x, *x(0)*, is sampled at every sample period, marked by sample clock. When a new sample arrives, the previous samples are shifted, so that the oldest sample *x(k-1)* is shifted out.

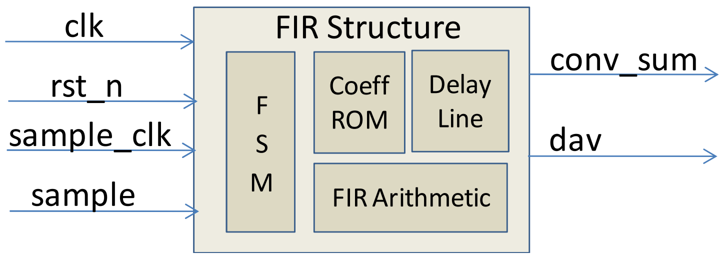


Figure 2. FIR Structure

2 Preparation Tasks

2.1 Laboration Directory Structure

For Lab-1 and Lab-2 you should create the following directory structure. These directories will have the following purposes;

IL2225_LAB: This will be the main directory which will have three more directories for LAB1 and LAB2 and global scripts. The global scripts directory will be used to store the scripts to instantiate different technology libraries.
LAB1/LAB2: These directories will store the files for the respective laborations.
synopsys_dc.setup: This file will contain the setup information for the design vision and will be discussed in the next section.
WORK: This directory will contain the working files for design analyzer.
SOURCE: Contains the design files for synthesis
SCRIPTS: You can save your scripts in this directory
MAPPED: The gate level netlist generated by the synthesis will be stored in this directory.
REPORTS: All the generated reports will be saved in this directory.

2.2 Fully Parallel FIR

For this laboratory the VHDL code for a fully parallel FIR filter is given in a folder IL2225_LAB/LAB1/ParallelFIR. The architecture of the fully parallel FIR is shown in Figure 3. To implement the delay line a shift register is used.

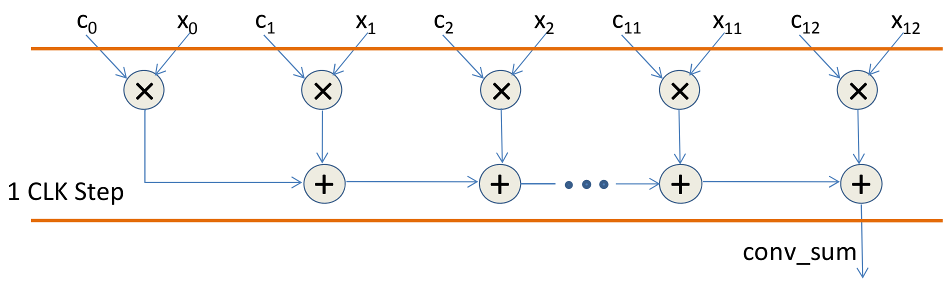


Figure 3. Fully parallel 13-tap FIR filter

1. Read the code and make a block diagram of the code on the paper so that we could know your understanding of the overall functionality of the code.
2. Do a functional simulation of the code. A fully functional test-bench is given. This test-bench gives only one sample to the shift register.

Question: How many adders and multipliers are there in the critical path of the FIR filter with the architecture shown in Figure 3? ____

The filter is symmetric which means $c_i = c_{k-i}$. Therefore, the FIR equation can be written as:

$$y = (x_0 + x_{12}).c_0 + (x_1 + x_{11}).c_1 + \cdots + (x_6 + 0).c_6$$

If the FIR filter is symmetric, it is more efficient to implement it as shown in the figure 4.

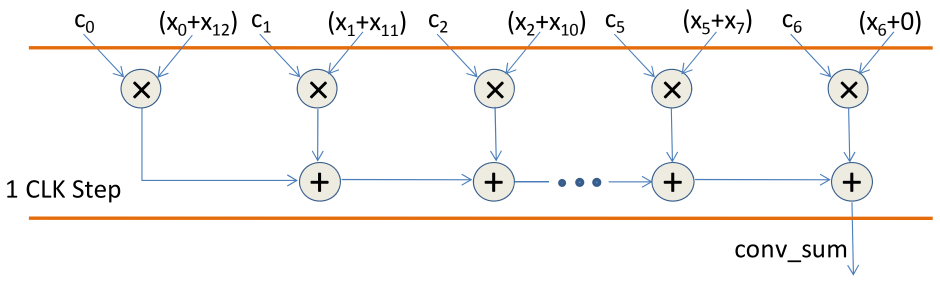


Figure 4. Symmetric fully parallel 13-tap FIR filter

Question: How many adders and multipliers are there in the critical path of the FIR filter with the architecture shown in Figure 4? Comparing the architectures shown in figure 3 and 4 which one has a shorter critical path? ____

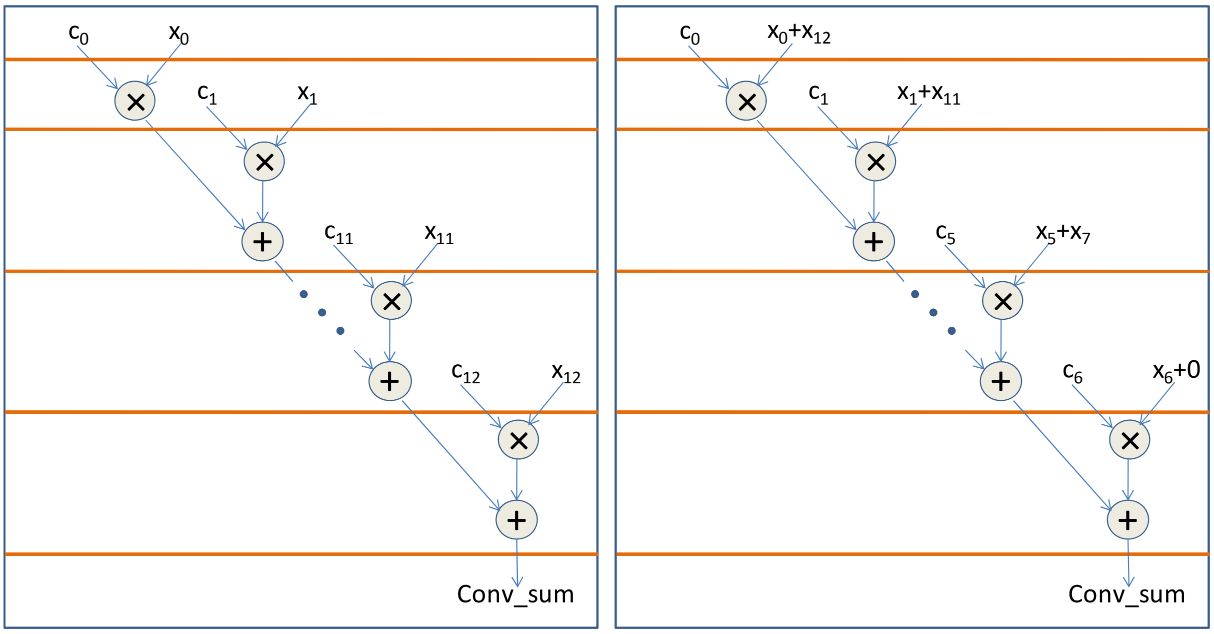
2.3 Fully Serial FIR

In the IL2225_LAB/LAB1/SerialFIR folder, the given code implements a fully serial FIR filter (Figure 5.a).

1. Read the code and make a block diagram of the code on the paper so that we could know your understanding of the overall functionality of the code.
2. Do a functional simulation of the code. A fully functional test-bench is given. This test-bench gives only one sample to the circular buffer.

Question: How many adders and multipliers are there in the critical path of the FIR filter with the architecture shown in Figure 5.a? ____

3. The efficient way of implementing a symmetric fully serial FIR filter is shown in figure figure5.b.



(a) Fully serial FIR filter (b) Symmetric fully serial FIR filter

Figure 5. Two different architectures for fully Serial FIR filter

Question: How many adders and multipliers are there in the critical path of the FIR filter with the architecture shown in Figure 5.a? ____

Question: Compare the number of the clock cycles that each of the architectures (5.a and 5.b) takes to produce the output. Which design has more throughput? ________________________________

2.4 Partially Parallel FIR Filter

In this lab, you are supposed to implement an asymmetric partially parallel FIR filter. Save your code in IL2225_LAB/LAB1/PartiallyParallelFIR/SOURCE). The architecture of the partially parallel FIR filter is shown in figure 6.

Hint:

- You can use the code of the serial FIR filter as the base to implement the partially parallel architecture.
- In every clock cycle, two MACs are working in parallel. Therefore, in every clock cycle, the FSM should generate two addresses to read samples from the circular buffer (figure 7).
- Note that if the number of filter taps is odd, in the last cycle only one MAC works.

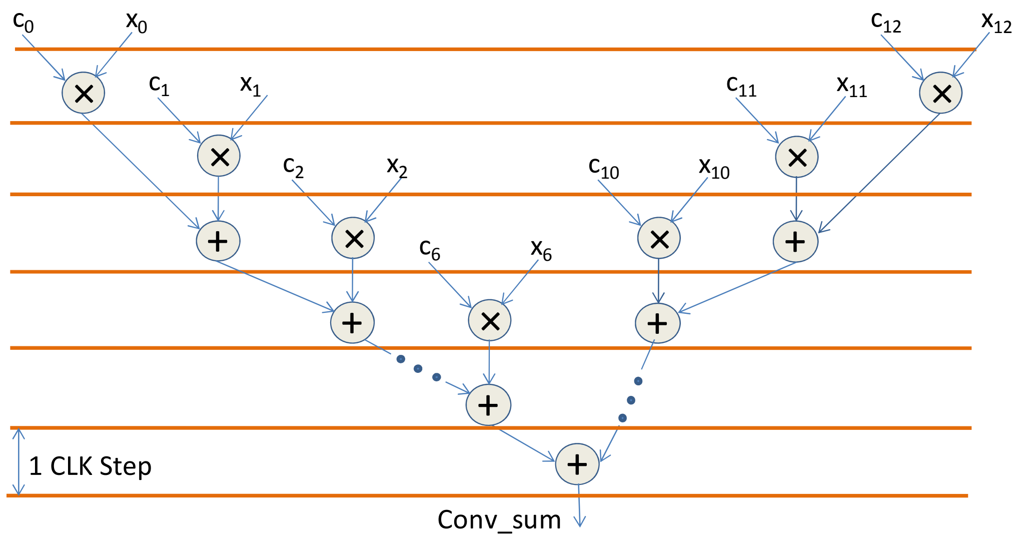


Figure 6. Partially parallel FIR filter

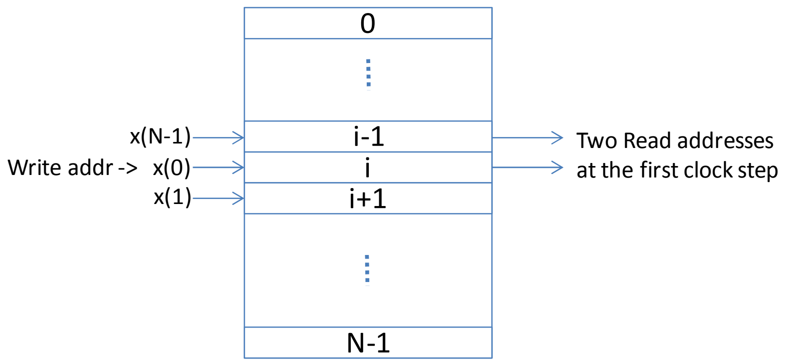


Figure 7. Circular buffer read and write addresses for a N-tap partially parallel FIR filter

Test the functionality of your code.

Question: How many adders and multipliers are there in the critical path of the FIR filter with the architecture shown in Figure 6? ____

Question: How many clock cycles does this architecture take to produce the output. __________________

____

3 Design Vision Initialization

When using the Synopsys synthesis tool, there are startup files that are read by Synopsys. The name of this start up file is “.synopsys_dc.setup”. A “.synopsys_dc.setup” file is a script file that is executed automatically when design_vision or dc_shell is invoked. The setup file is useful for setting up variables and commands that you want to execute every time you invoke Design Vision. In this start up file four important parameters must be set before you can perform any synthesis.

search_path: The parameter search_path is used to specify to the synthesis tool all the paths that it should search in when looking for a synthesis technology library to reference during synthesis.

target_library: The file pointed to by the parameter target_library is the library that contains all the logic cells for mapping during synthesis.

symbol_library: The parameter symbol_library points to the library that contains visual information on the logic cells in the synthesis technology library. All logic cells have a symbolic representation and information about the symbols is stored in this library.

link_library: The parameter link_library points to the library that contains information on the logic gates. Design Vision reads the setup file in the root of the home directory. Here the information that is general for all designs should be stored. A typical setup file located in the root of the home directory is:

```
set designer "Tom Cruise"
set company "KTH"
[getenv "SYN"]
set search_path “${SynopsysHome}/libraries/syn”
```

Then the setup file in the current directory is read. Here design specific information should be stored, e.g. the file that specifies the target technology.

```
set target_library "tcbn90gtc.db"
set symbol_library "tcbn90g.sdb"
set link_path "$search_path"
```

The Design Compiler is invoked by typing dc_shell in the UNIX shell. The Design Vision is the graphical front end version of DC and is launched by typing **design_vision.**

3.1 Design Vision Start-Up

1. Move to directory LAB1

```
Prompt>cd IL2225_LAB/LAB1
```

2. Invoke the design vision

```
/IL2225_LAB/LAB1/ParallelFIR> design_vision
```

3. Setup the path of the libraries by running the synopsys_dc.setup script.

Design_vision> source synopsys_dc.setup

4. Whenever you need help, you can access online documentation by clicking Help/online help.

5. The path of the library is

/afs/it.kth.se/pkg/synopsys/extra_libraries/standard_cell/TSMC/tcbn90g_110a/Front_End/timing_power/tcbn90g_110a/.

At this path you can find all the required libraries. So whenever you need to find a specific library, look into this path.

6. If any of the commands given in this lab is not working, try to press tab while entering the command in the command window in design vision. You will see a quick help telling you the exact syntax of the command.

7. You must know the reasons for the results you will be entering in the tables in rest of the lab. These reasons will help us judge if you have enough understanding to pass this lab.

4 Logic Synthesis Steps

We have already mentioned the process of synthesis which is described as translation plus optimization plus mapping. In terms of the Synopsys tools, translation is performed by the *analyze* plus *elaborate* command sequence. Optimization and mapping are performed by the compile command. To run each of the synthesis steps, both GUI and scripts can be used. To automate the synthesis process, you can create a script containing all of the required commands.

4.1 Design Entry

Before synthesis the design must be entered into DC in a standard format (VHDL, Verilog, EDIF, db etc). DC provides the following two methods of design entry: (a) *read* command (b) *analyze+elaborate* commands.

Read or Analyze

- The *read* command reads in files that are in another format than HDL, such as dcc and pla. Although it supports HDL format, it does not do the checking accomplished by analyze and elaborate.
- The *analyze* command reads a VHDL or Verilog file, checks for proper syntax and synthesizable logic, and stores the design in an intermediate format. Remember that you should read the files in order: 1- Packages, 2- All VHDL modules except top module, 3- Top level module (TOP_PARALLEL_FIR in the ParallelFIR design).

- analyze -format vhdl -lib WORK {"./SOURCE/filename.vhd"}

Elaborate

- The *elaborate* command creates a design from the intermediate format produced by analyze. The elaborate command replaces the HDL operators in the design with synthetic operators and determines the correct bus sizes. The design is now expressed by generic technology independent components (GTECHlibrary).
  - elaborate TOP_PARALLEL_FIR -architecture STRUCTURAL -library DEFAULT

4.2 Setting constraints

In this section we will learn to set constraints on our design. The constraints can be of following kinds:

Selecting Wire Load Models

The wireload model contains information that DC utilizes to estimate the interconnect delays during the pre-layout phase of the design. Several models appropriate to the different size of the logic are included in the technology library. These models define the capacitance, resistance, area factors. You can check all the available models by checking the tcbn90gtc.lib.

Question: Open this file in an editor and find the available wire_load models. List four of them in the following table.

Table: Wireload model

Wire-load-model	Resistance	Capacitance	Slope

Select wireload mode by:

- set_wire_load_mode (top/enclosed/segmented)

Select one of the wireload models using this command:

- set_wire_load_model -name “name of wireload model"

Question: What do you understand by wireload mode? ____

____

Setting Operating Conditions

Set of operating conditions defined in the library specify the process, temperature, voltage and RC tree values. These values are used during synthesis and timing analysis of the circuit.

Question: What is the default operating condition in tcbn90gtc.lib? ________________________________

Explore the target directory for tcbn90gtc.lib and open another lib file. Read the operating conditions in that file and fill in the following table:

Table: Operating conditions

Library	Condition	Process	Temperature	Voltage	tree_type
tcbn90gtc.lib	NCCOM	1	25 C	1.0 V	balanced_tree

Set the operating condition of the design using the following command.

- set_operating_conditions " OPERATINGCONDITION"

Modeling Clock

The design can be constrainted by a clock. A clock can be created by the following command.

- create_clock -name "clk" -period 5 -waveform { 0 2.5 } { clk }

False Path

The reset pin doesn't contribute to the signal path and can produce false timing analysis results. It's important to ignore the timing path originating from reset pins. For that reason we need to use the following command.

- set_false_path -from [find / -port rst_n]

4.3 Synthesis and Analysis

Mapping

The technology independent design is mapped to technology dependent library cells using the command :

- compile -map_effort medium

Save Design

For saving the design you can use this command:

- write -hierarchy -format ddc -output $HOME/IL2225_LABS/LAB1/Parallel_FIR.ddc

After saving a design, you can load it using *Read* command.

For saving the design in the Verilog netlist format use this command:

- write -hierarchy -format verilog -output $HOME/IL2225_LABS/LAB1/Parallel_FIR.v

Generate Reports

Now that the design has been mapped to standard cells, it is the time to generate reports. You need to generate the following reports. The “>>” operator appends all the reports in one file.

1. Constraints
   - report_constraints > ./REPORTS/constraint_Parallel_FIR.rep
2. Cells
   - report_cell > ./REPORTS/cell_Parallel_FIR.rep
3. Area
   - report_area > ./REPORTS/ area_Parallel_FIR.rep
4. Timing
   - report_timing > ./REPORTS/ timing_Parallel_FIR.rep
5. Power
   - report_power -analysis_effort low > ./REPORTS/ power_Parallel_FIR.rep

5 Experiments:

In this part, you have to synthesize all three discussed FIR filter architectures.

You can create a synthesis script for each design by writing all commands in a file *myScript.tcl* to automate the synthesis process and execute it using: *File -> Execute Scripts*

5.1 Fully Parallel FIR Filter:

Do the following experiments for all 3 designs located in 1. /IL2225_LABS/LAB1/ParallelFIR, 2. /SerialFIR, and 3./PartiallyParallelFIR.

Start-up the *Design Vision* form each of the 1- 3 design folders.

Analyze all of the files located in the SOURCE folder and *Elaborate* the top level module.

Question: How many memory elements (latches and flip-flops) are there in the design?

Architecture	Parallel FIR	Partially Parallel FIR	Serial FIR
Memory elements			

In the logical hierarchal window of design vision GUI, right click on FIR_Toplevel, then click schematic view to see the schematic generated by the design vision.

Question: Does your block diagram match the schematic generated by design vision? _______________

Set the design constraints as explained in the previous section and compile the design with the clock period of 5ns. Then save the design and generates the reports.

To display critical paths in the schematic view:

1. Choose, *Schematic* > *New Design Schematic View.*Make sure the schematic view is the active view.
2. Select the critical paths by choosing, *Select* > *Paths From/Through/To*, and then clicking OK in the Select Paths dialog box. The default options are set to select the path or paths with the worst negative slack in the design.
3. Highlight the selected paths by choosing, *Highlight* > *Selected.*
4. Use the magnification tools and the pointer to identify the startpoints and endpoints of the critical path.

Question: Does the critical path cross the boundaries of the modules? Write the start point and end point of the critical path in the table.

Architecture	Parallel FIR	Partially Parallel FIR	Serial FIR
Start point			
End point			

Question: Does the start and the end points of the critical path match with what is reported in the timing report? ____

Click on the ArithUnit on the schematic to see inside of the module. Highlight the critical path of this module.

Question: How many MACs are there in the critical path of the ArithUnit? Justify the observation___

____

Architecture	Parallel FIR	Partially Parallel FIR	Serial FIR
Number of MACs in the critical path			

Question: Which designs have the shortest and the longest critical paths? __________________________

Question: Which architecture can be sampled with the highest frequency? Will this design result in the highest throughput amongst the four architectures? Discuss your answer ___________________________

Reports:

Clock period = 5ns

Architecture	Parallel FIR	Partially Parallel FIR	Serial FIR
Area			
Timing Slack			

Question: Is the performance requirement met in all designs? Which design is the smallest and which one is the largest? Why? ____

Change the clock period and recompile the designs.

Clock period = 2.5 ns

Architecture	Parallel FIR	Partially Parallel FIR	Serial FIR
Area			
Timing Slack			

Question: Is the timing requirement met or violated for each of the designs? ________________________

Question: Compare the area of every design with the case of *CLK period=5 ns*. Why the area is increased by decreasing the clock period?____

6 Gate Level Simulation

In this section of the lab, your task is to do gate-level simulation using modelsim for your **partially parallel FIR filter**. The purpose of the gate-level simulation is to verify the functionality of the gate-level netlist. You should show your working gate-level simulations to the lab assistant. For gate level simulation,

Create a folder containing the Verilog netlist of your design ($home/IL2225_LAB /PartiallyParallelFIR/db/main.v), misc.vhdl and myPackage.vhdl packages, testbench, and tcbn90g.v library file. You can find this library file in the

/afs/it.kth.se/pkg/synopsys/extra_libraries/standard_cell/TSMC/tcbn90g_110a/Front_End/verilog/tcbn90g_110a$ directory.

For gate-level simulation you need to consider that:

After/during synthesis, check for the name of your top level design. Synthesis tool changes the name according to generics used. For instance, the name of top level in our design may be changed to TopFIR_5. You have to use this name in your test bench for proper simulation.

Check if the output is correct. You must show your gate level simulation to the assistant to pass this lab.

Good luck.

Appendix A: Sample Synthesis Script

```
####### Set Global Libraries #########
source synopsys_dc.setup
####### Set Directary #########
set LIB typical
set SYNDIR /home/ASICLABS/IL2225_LAB/LAB1/ParallelFIR
set SRCDIR /home/ASICLABS/IL2225_LAB/LAB1/ParallelFIR/SOURCE
set SCRDIR /home/ASICLABS/IL2225_LAB/LAB1/ParallelFIR/SCRIPTS
set RPTDIR /home/ASICLABS/IL2225_LAB/LAB1/ParallelFIR/REPORTS
set SYNDB  /home/ASICLABS/IL2225_LAB/LAB1/ParallelFIR/db
######Enviroment############
define_design_lib WORK -path $SYNDIR/WORK
######Read Design###########
analyze -library WORK -format vhdl {$SRCDIR/misc.vhd}
analyze -library WORK -format vhdl {$SRCDIR/myPackage.vhd}
analyze -library WORK -format vhdl {$SRCDIR/FSM.vhd}
analyze -library WORK -format vhdl {$SRCDIR/ShiftRegister.vhd}
analyze -library WORK -format vhdl {$SRCDIR/coefRom.vhd}
analyze -library WORK -format vhdl {$SRCDIR/MAC.vhd}
analyze -library WORK -format vhdl {$SRCDIR/ArithUnit.vhd}
analyze -library WORK -format vhdl {$SRCDIR/TopFIR.vhd}
######Elaborate Design###########
elaborate Top_Parallel_FIR -lib WORK
########Set Constraints#############
set_wire_load_mode top
set_wire_load_selection_group WireAreaLowkCon
set_operating_conditions -library tcbn90gtc NCCOM
create_clock -name "clk" -period 5 -waveform { 0 2.5 }  { clk  }
set_false_path -setup -reset_path  -from { rst_n }
set_false_path -hold -reset_path  -from { rst_n }
#######Compile Option############
compile -map_effort medium
#######Report####################
report_timing > $RPTDIR/report_timing.rpt
report_area > $RPTDIR/report_area.rpt
#######Save Design####################
write -hierarchy -format ddc -output $SYNDB/main.ddc
write -format verilog -hier -o $SYNDB/main.v
write_sdf -version 2.1 $SYNDB/main.sdf
```