Lab 2: Logic Synthesis & timing constraints

Objective

To study the setup of Design Compiler RTL synthesis for technology-specific libraries.

Theoretical part

Design Compiler provides the following ways to read the design files: analyze, elaborate, read. During the design process, Design Compiler uses technology-specific libraries, including target and link libraries. Target library is used to build a circuit. During mapping, Design Compiler selects functionally correct gates from the target library. It also calculates the timing of the circuit using the vendor-supplied timing data for these gates.
Design Compiler uses the link library to resolve references. For the design to be complete, it must connect to all library components and designs it references.

Practical part

In this part, we will use an opensource I2C Master design to demonstrate the synthesis process using Design Compiler. More information about the design can be found in this URL:

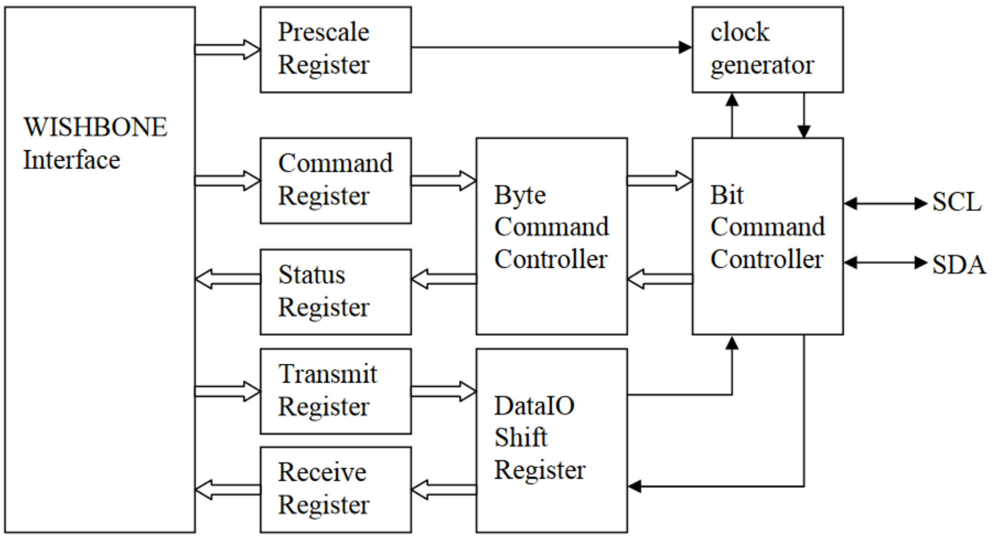


Fig. 1. I2c master controller block diagram.

1. Move to the working directory lab02/syn

```
$ cd $HOME/icdesign/m3/lab02/syn
```

2. Simulate the design

This design has both Verilog and VHDL implementation. You can simulate this design in Verilog using the following commands:

```
vcs -debug_access+all           \
     +incdir+../src/tb/verilog       \
     +incdir+../src/rtl/verilog      \
     ../src/rtl/verilog/i2c_master_bit_ctrl.v        \
     ../src/rtl/verilog/i2c_master_byte_ctrl.v       \
     ../src/rtl/verilog/i2c_master_top.v             \
     ../src/tb/i2c_slave_model.v     \
     ../src/tb/wb_master_model.v     \
     ../src/tb/tst_bench_top.v
./simv
```

3. Invoke Design Compiler

```
$ source /home/tools/synopsys/env.sh
$ dc_shell -gui
```

4. Setup target and link libraries.

In this lab, we will use Synopsys Education (non-manufacturable) 14nm technology. The standard cell libraries and related tech files are stored in /home/dkits/synposys/SAED14nm. There are 3 flavors including High Voltage Threshold (HVT), Low Voltage Threshold (LVT) and Normal Voltage Threshold (RVT). In this lab, we will use RVT standard cells only.
Take a look at the standard cell library organization:

```
stdcell_rvt
├── db_ccs
├── db_nldm
├── doc
├── gds
├── hspice
├── lef
├── lvs
├── milkyway
├── ndm
└── verilog
```

- db_ccs: standard cell libraries characterized using Composite Current Source model
- db_nldm: standard cell libraries characterized Non-Linear Delay Model
- doc: standard cell library document
- gds: standard cell layout in GDSII format
- hspice: standard cell SPICE models
- lef: abstract view of the standard cells in LEF format
- lvs: spice model for Layout-Versus-Schematic check
- Milkyway: abstact view of the standard cell layouts in Milkyway libraries
- ndm: new IC compiler II library format
- verilog: standard cell verilog models

In this lab, we will use the NLDM standard cell library format to do synthesis, because it is quicker to run. However, CCS model give more accurate results. You can try with CCS libraries after complete this lab and compare the results.

Take a look at `/home/dkits/synopsys/SAED14nm/stdcell_rvt/db_nldm by running this command:`

```
ls -al /home/dkits/synopsys/SAED14nm/stdcell_rvt/db_nldm
```

How many operating corners are provided in this technology?

Open /home/dkits/synopsys/SAED14nm/stdcell_rvt/db_nldm/saed14rvt_tt0p8v25c.lib using a text editor. What is the area of the 2-input NAND with drive strength of 1 in this standard cell libraries? What is the operating voltage, temperature and the process corner?

To setup the link and target libraries, use the following command:

```
dc_shell> set search_path ". /home/dkits/synopsys/SAED14nm/stdcell_rvt/db_nldm ../src/rtl/verilog"
dc_shell> set link_library          "* saed14rvt_tt0p8v25c.db"
dc_shell> set target_library        "saed14rvt_tt0p8v25c.db"
dc_shell> set_min_library saed14rvt_ss0p6v125c.db -min_version saed14rvt_ff0p88vm40c.db
```

- `search_path` variable is used to set the search path for source files, libraries files for different command in design compiler.
- `link_library` is the list of libraries to link the design. The star (*) at the beginning of the list to tell design compiler to link with the design that has been read into it.
- `target_library` is a list of libraries that the design will be synthesized (mapped) into.
- `set_min_library` is used to declare the worst case and best case (-min_version). These libraries can be used to do timing analysis.

It is also possible to set link_library (link_path) and target_library using the GUI by selecting File >> Setup as in Fig. 2.

Fig. 2. Dialog box of setting up link and target libraries.

5. Analyze designs

The analyze command does the following:

- Reads an HDL source file
- Checks it for errors (without building generic logic for the design)
- Creates HDL library objects in an HDL-independent intermediate format
- Stores the intermediate files in a defined location

```
dc_shell> analyze -library work -format verilog "i2c_master_bit_ctrl.v \
      i2c_master_byte_ctrl.v \
      i2c_master_defines.v \
      i2c_master_top.v"
```

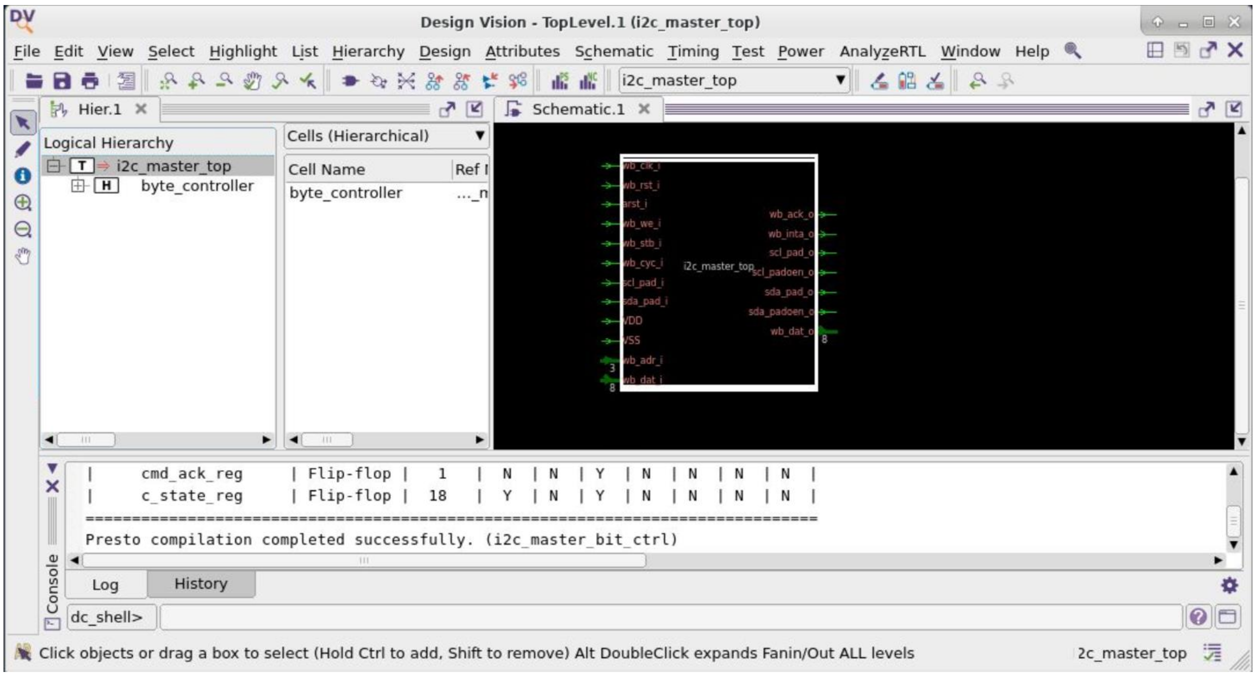
6. Elaborate designs.

The elaborate command does the following:

- Translates the design into a technology-independent design (GTECH) from the intermediate files produced during analysis
- Allows changing of parameter values defined in the source code
- Allows VHDL architecture selection
- Replaces the HDL arithmetic operators in the code with DesignWare components
- Automatically executes the link command, which resolves design references

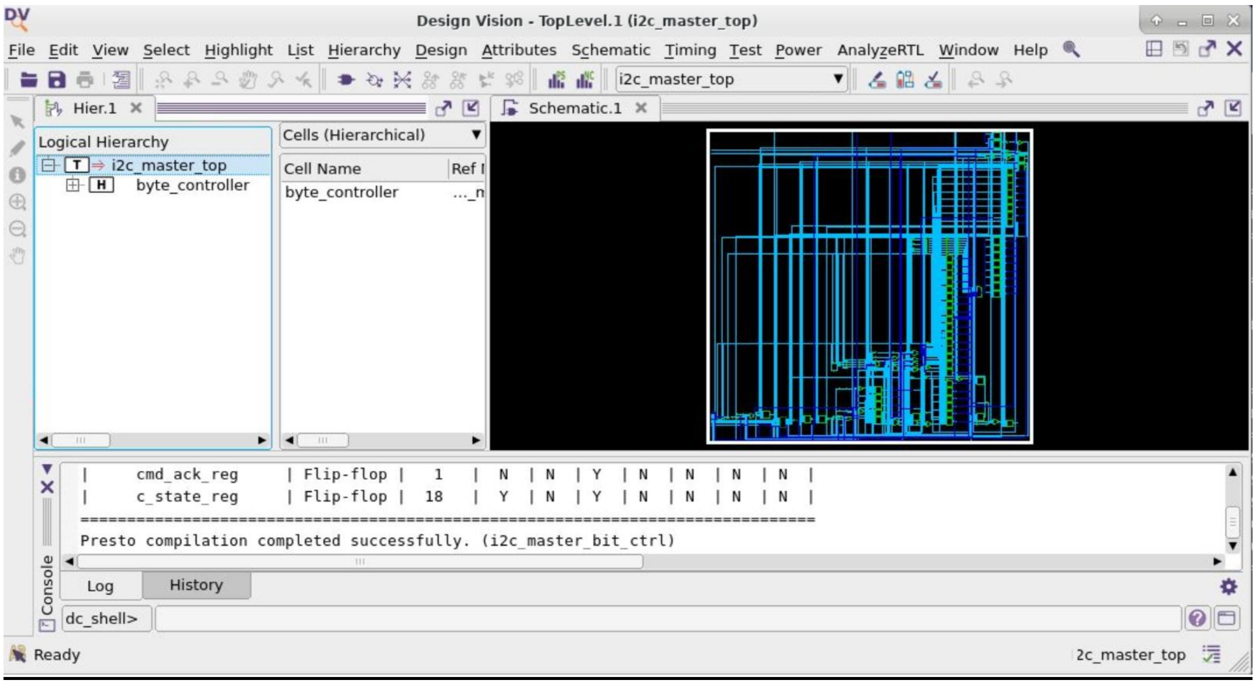
```
dc_shell> elaborate i2c_master_top -architecture verilog
```

After successfull elaboration, you can view the schematic by right-click on i2c_master_top in the Logical Hierachy windows, select Schematic View. After that, in the Schematic View, double click on i2c_master_top to view the schematic as shown in Fig. 4.



```
|      cmd_ack_reg      | Flip-flop |   1   |  N  | N  | Y  | N  | N  | N  | N  |
|      c_state_reg      | Flip-flop |  18   |  Y  | N  | Y  | N  | N  | N  | N  |
================================================================================
Presto compilation completed successfully. (i2c_master_bit_ctrl)
```

Click objects or drag a box to select (Hold Ctrl to add, Shift to remove) Alt DoubleClick expands Fanin/Out ALL levels



```
|      cmd_ack_reg      | Flip-flop |   1   |  N  | N  | Y  | N  | N  | N  | N  |
|      c_state_reg      | Flip-flop |  18   |  Y  | N  | Y  | N  | N  | N  | N  |
===============================================================================
Presto compilation completed successfully. (i2c_master_bit_ctrl)
```

Fig. 4. Design view from Design Compiler

7. Design linking & checking

```
dc_shell> current_design i2c_master_top
dc_shell> link
dc_shell> check_design
```

- `current_design` command set the design that will be synthesized in case there are many different designs in the Design Compiler's memory.
- `link` command will check if there are missing (unresolved) design
- `check_design` command will perform design check before performing additional operation

8. Set the best case and work case operating condition for timing analysis

```
dc_shell> set_operating_conditions -analysis_type bc_wc \
          -min ff0p88vm40c -min_library saed14rvt_ff0p88vm40c.db:saed14rvt_ff0p88vm40c \
          -max ss0p6v125c -max_library saed14rvt_ss0p6v125c.db:saed14rvt_ss0p6v125c
```

- set_operating_conditions command sets the analysis type. In this case, we will use the best case/worst case analysis

9. Set the wireload models

```
dc_shell> set_wire_load_mode enclosed
```

- set_wire_load_mode configures the Wire Load Model

10. Design constraint

Create the design constraint for the design with the specification as follows:

- Operating frequency: 500MHz
- Clock signal: wb_clk_i
- Clock uncertanty: 0.3ns
- Clock transition: 0.5ns
- Set input delays and output delays and also other attributes if possible

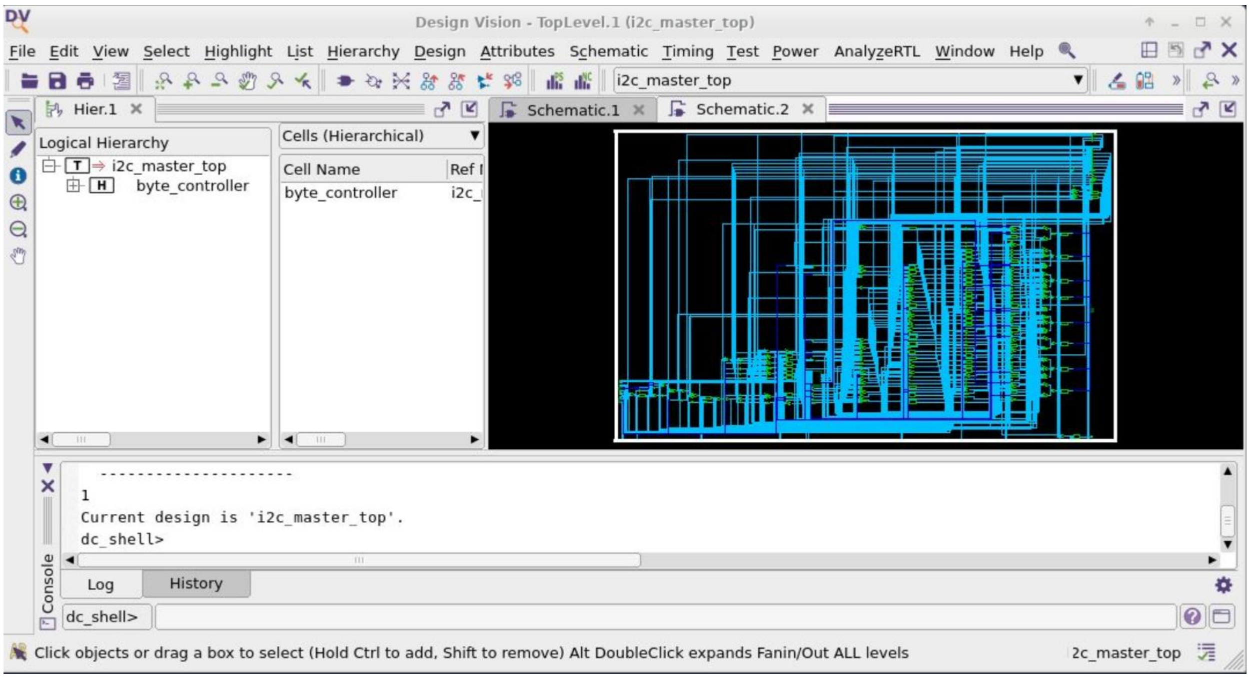
Write the constraints into scripts/i2c_master_top.sdc. You can load the file into Design Compiler by the following commands:

```
dc_shell> source -echo -verbose scripts/i2c_master_top.sdc
```

11. Design mapping

To perform design mapping, **compile** command is used. The mapped design view from Design Compiler is given in Fig. 5.

```
dc_shell> compile
```


Design Vision - TopLevel.1 (i2c_master_top)
File Edit View Select Highlight List Hierarchy Design Attributes Schematic Timing Test Power AnalyzeRTL Window Help
i2c_master_top
Hier.1
Schematic.1
Schematic.2
Logical Hierarchy
i2c_master_top
byte_controller
Cells (Hierarchical)
Cell Name
Ref I
byte_controller
i2c_
Log
History
Console
Click objects or drag a box to select (Hold Ctrl to add, Shift to remove) Alt DoubleClick expands Fanin/Out ALL levels
2c_master_top

```
--------------------
1
Current design is 'i2c_master_top'.
dc_shell>
```

12. Generating reports

```
dc_shell> file mkdir reports
dc_shell> report_timing > reports/timing.rpt
dc_shell> report_qor > reports/qor.rpt
dc_shell> report_area -hierarchy > reports/area.rpt
dc_shell> report_power -hierarchy > reports/power.rpt
```

Does the design meet the timing requirement?
What is the area occupied by the design?

13. Save the results

```
dc_shell> file mkdir results
dc_shell> change_names -rule verilog
dc_shell> write_file -format verilog -hierarchy -output results/i2c_master_top_mapped.v
dc_shell> write_file -format ddc -hierarchy -output results/i2c_master_top_mapped.ddc
```

14. Resynthesize the design with different options

You can try to change the timing constraint, or compile options and report the change in terms of area and timing

Modify the timing constraints that also consider the input transition, load capacitance and so on

Try to compile the design using compile_ultra instead of compile and compare the area and timing of the design in these case. The design has smaller area and better timing closure?