

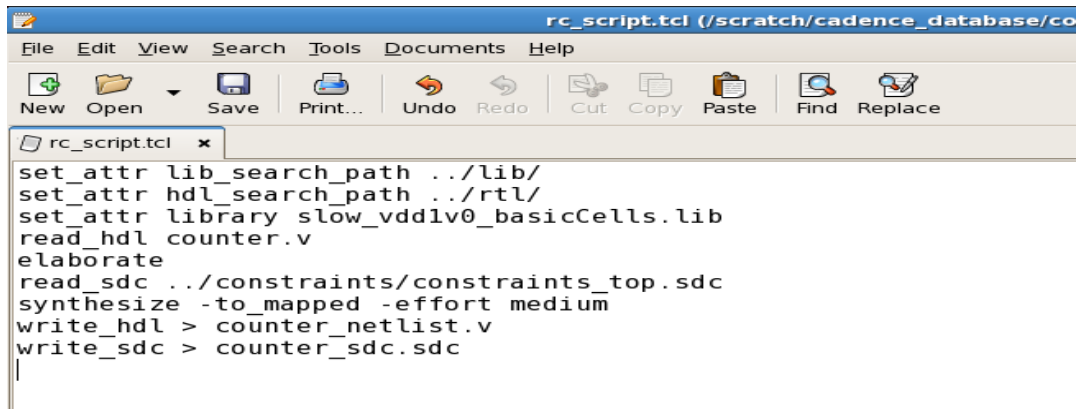
Synthesis

The tool used for synthesis (converting RTL to gate level netlist) is RTL Compiler (RC).

Running Synthesis (without DFT)

Change the directory to synthesis and write a script file for synthesis.

Below is an example of a script file for synthesis.



```
rc_script.tcl (/scratch/cadence_database/co)
File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
rc_script.tcl x
set_attr lib_search_path ../lib/
set_attr hdl_search_path ../rtl/
set_attr library slow_vddl0_basicCells.lib
read_hdl counter.v
elaborate
read_sdc ../constraints/constraints_top.sdc
synthesize -to_mapped -effort medium
write_hdl > counter_netlist.v
write_sdc > counter_sdc.sdc
```

- The necessary inputs to perform synthesis are RTL, standard cell library and constraints.
- Let us see the usage and purpose of each command.
 - **set_attribute lib_search_path <library path>**
This command will set the path for the standard cell library.
 - **set_attribute hdl_search_path <rtl path>**
This command will set the path for rtl files.
 - **set_attribute library <library name>**
This command will read the specified standard cell library from the specified library path.
 - **read_hdl <rtl design>**
This command will read the rtl design.

Note: - If the design is hierarchical or has multiple modules instantiated inside the top module, use curly braces '{ }' to mention all modules including the top design.

E.g.:- read_hdl {top.v sub1.v sub2.v}
 Here top.v is the top module and sub1.v and sub2.v are the sub modules that are instantiated inside the top module.
 - **elaborate**
The elaborate command constructs design hierarchy and connects signals.

- **read_sdc < sdc file name with path>**

This command reads in the timing constraints file. Here we have to provide the constraints file name along with the path. Explanation on constraints file is provided

- **Synthesize –to_mapped –effort medium**

This command will perform synthesis by combining the generic, mapped and incremental synthesis and – effort medium command specifies the synthesis effort. The effort can be set to ‘low’, ‘medium’ or ‘high’ depending upon the scenario.

➤ Include all the above commands in the script file.

Note: - In counter design, you can see a script file ‘rc_script.tcl’ inside the synthesis directory. Please open the script file for further understanding.

Timing Constraints or SDC file

Now let us understand the content of the Constraints or SDC file.

➤ Using SDC, we define clock period, pulse width, rise and fall time, uncertainty and also input and output delays for different signals. Below is the constraints file used in counter design.

```
create_clock -name clk -period 10 -waveform {0 5} [get_ports "clk"]
set_clock_transition -rise 0.1 [get_clocks "clk"]
set_clock_transition -fall 0.1 [get_clocks "clk"]
set_clock_uncertainty 0.1 [get_ports "clk"]
set_input_delay -max 1.0 [get_ports "rst"] -clock [get_clocks "clk"]
set_output_delay -max 1.0 [get_ports "count"] -clock [get_clocks "clk"]
```

Fig 14

➤ Let us see the usage and purpose of each command.

- **Create_clock –name –period 10 –waveform {0 5} {get_port “clk”}**

This command will define clock with period 10ns and 50% duty cycle and signal is high in the first half.

- **‘Set_clock_transition –rise/fall’** command defines the transition delay for clock.

- **‘Set_clock_uncertainty’** command will set the uncertainty due to (clock skew and jitter).

- **‘Set_input/output_delay’** command will specify the input and output delay used for timing slack calculations.

➤ Keep the constraints file inside the constraints directory.

Note: - To know more about writing timing constraints, please refer the rc_ta.pdf available inside the doc/rc_ta directory of the tool.

I.e. - <RC_tool directory>/doc/rc_ta/rc_ta.pdf

Once the script file to run synthesis and the constraints file are ready, we can initiate synthesis.

- Use the below command to invoke RTL compiler along with the script file.

rc -f <script file name with path>

'rc' is the command to invoke RTL Compiler and '-f' option is used to pass the script to RC at the time of launching the tool. RC will execute each command mentioned inside the script file one by one.

Note: - If the script file is in the current working directory (synthesis directory), we need not have to provide the path for the script.

E.g. - In case of the counter design, the command will be 'rc -f rc_script.tcl'

- While performing synthesis, always check the RC terminal whether the tool is reporting any error. Figure below shows the RC terminal after synthesis.

```

Terminal
File Edit View Terminal Tabs Help
Incremental optimization status
=====
              Group
              Tot Wrst - - DRC Totals - -
              Weighted  Max  Max
Operation      Area  Slacks  Trans  Cap
-----
init_iopt        762      0      0      0

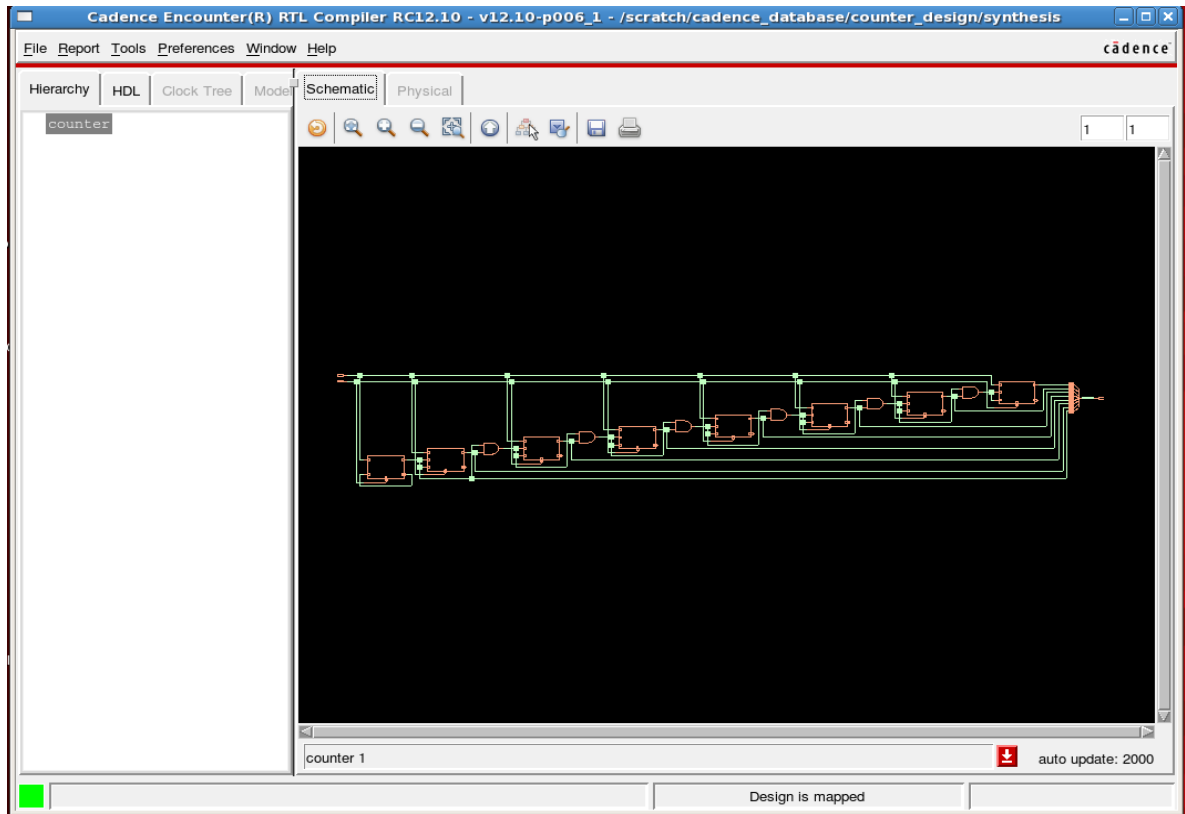
Incremental optimization status
=====
              Group
              Tot Wrst - - DRC Totals - -
              Weighted  Max  Max
Operation      Area  Slacks  Trans  Cap
-----
init_delay        762      0      0      0
init_drc          762      0      0      0
init_area         762      0      0      0
rem_inv_qb        762      0      0      0

Incremental optimization status
=====
              Group
              Tot Wrst - - DRC Totals - -
              Weighted  Max  Max
Operation      Area  Slacks  Trans  Cap
-----
init_delay        762      0      0      0
init_drc          762      0      0      0
init_area         762      0      0      0

Done mapping counter
Synthesis succeeded.
rc:/>

```

- After synthesis, to see the schematic, launch the gui using the below command.
'gui_show' in the 'rc' terminal. Use 'gui_hide' command to close the gui.



- Use 'report' command to write out the results.
 - **'report timing'** to report the timing details.
 - **'report power'** to dump out the power report.

Note: - Use just report command to know what all different reports you can dump out from RC.

- After completing synthesis, Use the below command to dump out netlist, SDF, SDC etc for next stages of the flow.
 - **write_hdl > netlist_name.v**
You can provide any name for the netlist file but the extension of the file should be '.v'
 - **write_sdc > sdc_name.sdc**
You can provide any name for the SDC file but the extension of the file should be '.sdc'
 - **write_sdf -timescale ns -nonechecks -recrem split -edges check_edge > delays.sdf**

timescale: used to mention the timeunit.

nonechecks: used to ignore the negative timing checks.

recrem: used to split out the recrem(recovery-removal) timing check to separate checks for recovery and removal.

edges: Specifies the edges values.

check_edge Keeps edge specifiers on timing check arcs but does not add edge specifiers on combinational arcs.

The above commands will generate netlist, SDF and SDC in the synthesis directory.

- Use **'exit'** command to close RC.

Note: - To know more about synthesis, please refer the rc_user.pdf available inside the doc/rc_ta directory of the tool.

I.e. - <RC_tool directory>/doc/rc_user/rc_user.pdf

After dumping out the netlist and SDC, we can proceed for Physical Design. Please close the RC tool.

Synthesis with DFT

Understanding the Flow

- Let us look at the files we are working with in this lab. Change directory to synthesis and locate the **rc_dft_script.tcl**. The main purpose of this script is to set up the variables and libraries that will be used in the lab.
- Now let us look at the content of the run script(rc_dft_script.tcl). Here is a breakdown of the script flow for clarity:

- Load all the design files and elaborate
- Read sdc (in constraints dir)
- Read in DFT setup
- Synthesize to GENERIC
- Synthesize to MAPPED
- Run DFT flow
- Synthesis to MAPPED
- Write results and database
- Write ET files and ATPG flow



Same as explained in normal synthesis flow.

Below snapshot shows rc_dft_script.tcl

```

set_attr lib_search_path ../lib/
set_attr hdl_search_path ../rtl/
set_attr library slow_vddl0_basicCells.lib
read_hdl counter.v
elaborate
read_sdc ../constraints/constraints_top.sdc
synthesize -to_mapped
set_attr dft_scan_style muxed_scan
set_attr dft_prefix dft_
define_dft shift_enable -name SE -active high -create_port SE
synthesize -to_mapped
check_dft_rules
set_attr dft_min_number_of_scan_chains 1 /designs/counter
define_dft scan_chain -name top_chain -sdi scan_in -sdo scan_out -create_ports
synthesize -to_mapped
connect_scan_chains -auto_create_chains
synthesize -to_mapped
report_dft_chains
write_sdf -nonegchecks -edges check_edge -timescale ns -recrem split > delays.sdf
write_et_atpg -library ../lib/slow_vddl0_basicCells.v
write_hdl > counter_netlist_dft.v
write_sdc > counter_sdc.sdc

```

Let us understand the DFT specific commands in rc_dft_script.tcl file.

- The DFT scan FF style for scan replacement using the below command.
set_attr dft_scan_style muxed_scan
- Prefix is added to name of DFT logic that is inserted using the below command.
set_attribute dft_prefix DFT_ /
- Define the test signals (define_dft shift_enable) using the below command.
define_dft shift_enable -name {scan_en} -active {high} -create_port {scan_en}
- It is recommended you check DFT rules multiple times during a DFT flow using the below command.
check_dft_rules

```

Summary of check_dft_rules
*****
Number of usable scan cells: 48
Clock Rule Violations:
-----
    Internally driven clock net: 0
    Tied constant clock net: 0
    Undriven clock net: 0
    Conflicting async & clock net: 0
    Misc. clock net: 0

Async. set/reset Rule Violations:
-----
    Internally driven async net: 0
    Tied active async net: 0
    Undriven async net: 0
    Misc. async net: 0

Total number of DFT violations: 0

Total number of Test Clock Domains: 1
Number of user specified non-Scan registers: 0
Number of registers that fail DFT rules: 0
Number of registers that pass DFT rules: 8
Percentage of total registers that are scannable: 100%

```

As you can see that there are no registers that fail DFT rules, which means that all of 8 registers are eligible for scan connection.

- Specify the number of scan chains required to connect all FF's using the below command. Here we have used 1 scan chain.
set_attr dft_min_number_of_scan_chains 1 /designs/counter
- Specify the scan in and scan out ports of the scan chain using the command
define_dft scan_chain -name top_chain -sdi scan_in -sdo scan_out -create_ports
- Once scan ports has been created, perform the technology depended synthesis using the below command.
synthesis -to_mapped.
- Now connect the Scan chains using connect_scan_chains RC command. This will include all original FF's that were mapped to scan flops.
connect_scan_chains -auto_create_chains
- You can view the dft chains using the below command as shown.
report dft_chains

```

Reporting 1 scan chain (muxed_scan)

Chain 1: top_chain
scan_in:      scan_in
scan_out:     scan_out
shift_enable: SE (active high)
clock_domain: clk (edge: rise)
length: 8
bit 1 count_reg[0] <clk (rise)>
bit 2 count_reg[1] <clk (rise)>
bit 3 count_reg[2] <clk (rise)>
bit 4 count_reg[3] <clk (rise)>
bit 5 count_reg[4] <clk (rise)>
bit 6 count_reg[5] <clk (rise)>
bit 7 count_reg[6] <clk (rise)>
bit 8 count_reg[7] <clk (rise)>
-----

```

- We will now run the final ATPG analysis and vector generation. This step will take the final scan chains and run through the ET flow for basic ATPG. This flow is implemented by the command

```
write_et_atpg -library ../Lib/slow_vdd1v0_basiccells.v
```

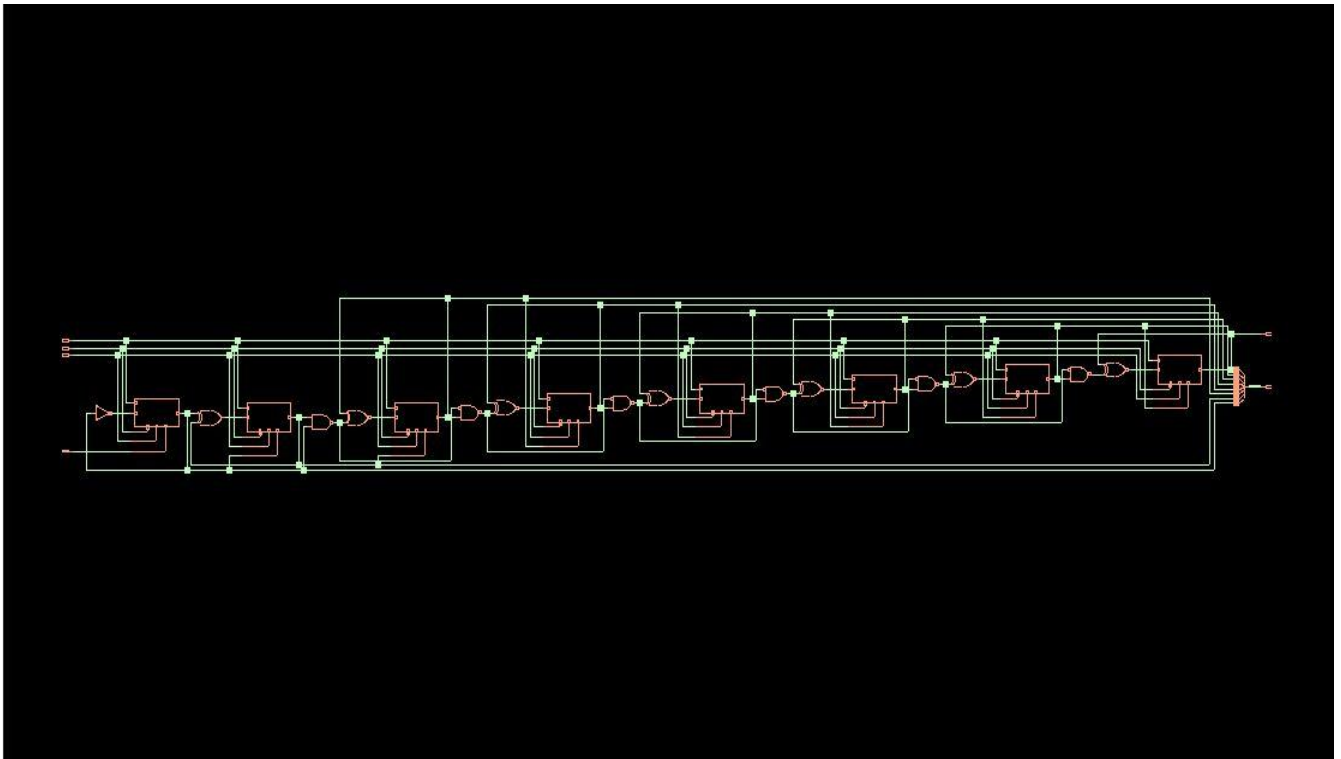
It will generate a directory '**et_scripts**' in current working location.

- Write out the final netlist, SDF & constraints using the below commands.

```
Write_hdl > counter_netlist.v
```

```
write_sdf -timescale ns -nonegchecks -recrem split -edges check_edge > delays.sdf
```

```
Write_sdc > count_sdc.sdc
```



Change directory to et_scripts to see the files that are generated by RC.

- counter.et_netlist.v (completed verilog netlist used for ET)
- runet.atpg (ET ATPG run script)
- counter.FULLSCAN.pinassign (ET file specifying IO test behavior)
- et_check.sh (self error checking file used by runet.atpg)
- run_fullscan_sim (NC-sim script to verify ATPG patterns)