

## HOW TO SYNTHESIZE VERILOG CODE USING RTL COMPILER

This tutorial explains how to synthesize a verilog code using RTL Compiler. In order to do so, let's consider the verilog codes below.

CNT\_16 Module:

16 bit up counter with asynchronous active-low reset

```
`timescale 1ns/1ps

module CNT_16(CLK, RSTB, OUT);

input CLK, RSTB;
output [15:0] OUT;

reg [15:0] OUT;

always@(posedge CLK or negedge RSTB) begin
    if(!RSTB) begin
        OUT <= 0;
    end
    else begin
        OUT <= OUT + 1;
    end
end

endmodule
```

CMP\_16 Module:

Comparator which compares the 16 bit input with 50.

```
`timescale 1ns/1ps

module CMP_16(IN, OUT);

input [15:0] IN;
output OUT;

reg OUT;

always@(IN) begin
    if(IN <= 50) begin
        OUT = 0;
    end
end
```

```
        else begin
            OUT = 1;
        end
    end
end

endmodule
```

TOP Module:

The top module which connects both. The resulting design makes OUT=1 after 50 clock cycles.

```
`timescale 1ns/1ps

module TOP(CLK, RSTB, OUT);

    input CLK, RSTB;
    output OUT;

    wire [15:0] OUT_16;

    CMP_16 U1(OUT_16, OUT);
    CNT_16 U2(CLK, RSTB, OUT_16);

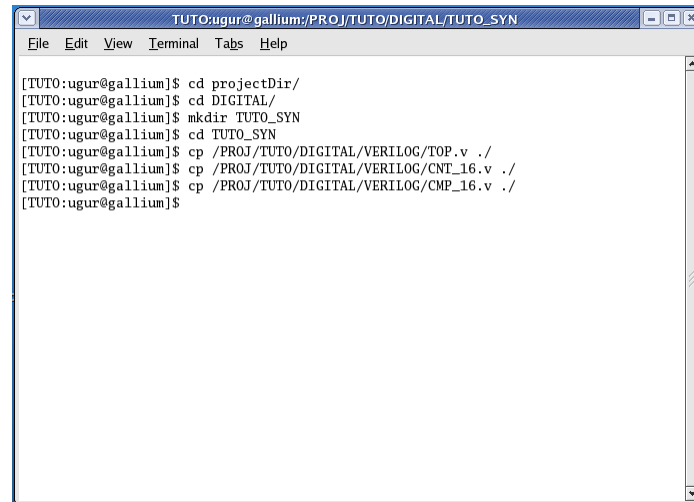
endmodule
```

To start synthesis, let's make a new folder.

A terminal window titled 'TUTO:ugur@gallium:/PROJ/TUTO/DIGITAL/TUTO\_SYN' with a menu bar (File, Edit, View, Terminal, Tabs, Help). The terminal shows the following commands and output:

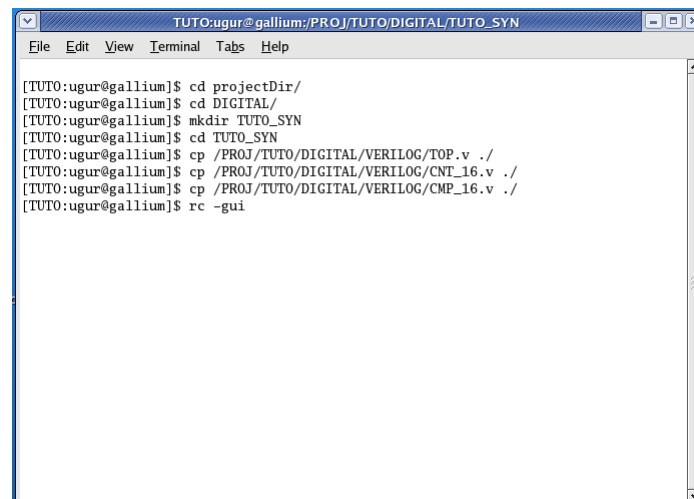
```
[TUTO:ugur@gallium]$ cd projectDir/
[TUTO:ugur@gallium]$ cd DIGITAL/
[TUTO:ugur@gallium]$ mkdir TUTO_SYN
[TUTO:ugur@gallium]$ cd TUTO_SYN
[TUTO:ugur@gallium]$
```

Then let's copy all Verilog files to newly created folder.

A terminal window titled 'TUTO:ugur@gallium:/PROJ/TUTO/DIGITAL/TUTO\_SYN' with a menu bar (File, Edit, View, Terminal, Tabs, Help). The terminal shows a series of commands to set up a project directory and copy Verilog files.

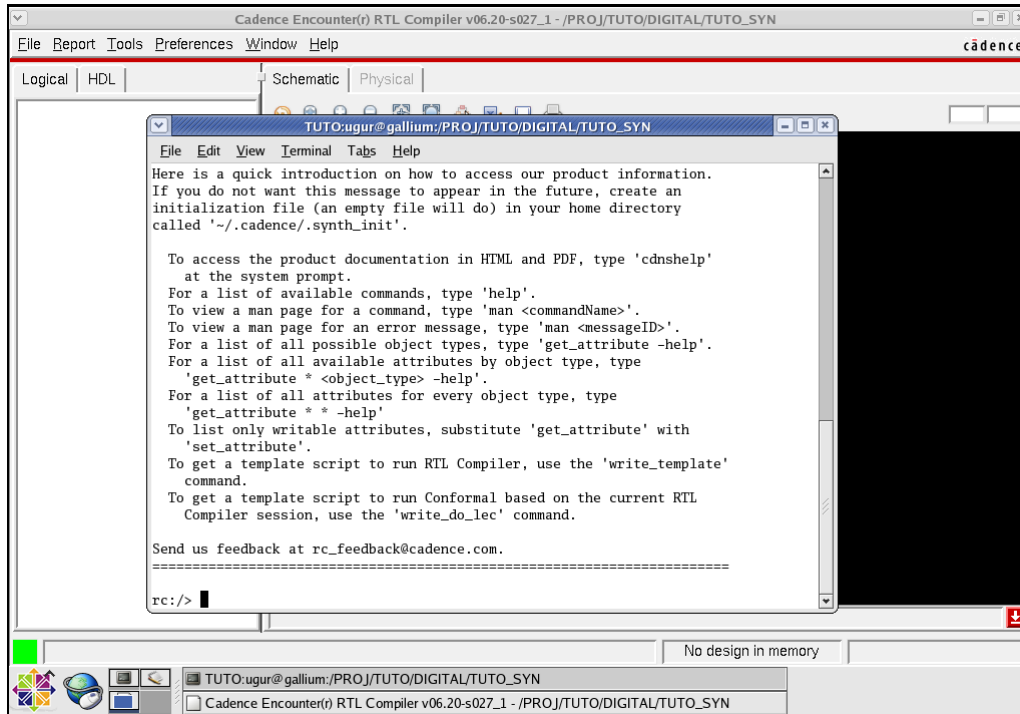
```
[TUTO:ugur@gallium]$ cd projectDir/  
[TUTO:ugur@gallium]$ cd DIGITAL/  
[TUTO:ugur@gallium]$ mkdir TUTO_SYN  
[TUTO:ugur@gallium]$ cd TUTO_SYN  
[TUTO:ugur@gallium]$ cp /PROJ/TUTO/DIGITAL/VERILOG/TOP.v ./  
[TUTO:ugur@gallium]$ cp /PROJ/TUTO/DIGITAL/VERILOG/CNT_16.v ./  
[TUTO:ugur@gallium]$ cp /PROJ/TUTO/DIGITAL/VERILOG/CMP_16.v ./  
[TUTO:ugur@gallium]$
```

Let's start our synthesizer with **rc -gui** command.

A terminal window titled 'TUTO:ugur@gallium:/PROJ/TUTO/DIGITAL/TUTO\_SYN' with a menu bar (File, Edit, View, Terminal, Tabs, Help). The terminal shows the same setup commands as the previous window, followed by the 'rc -gui' command to start the synthesizer.

```
[TUTO:ugur@gallium]$ cd projectDir/  
[TUTO:ugur@gallium]$ cd DIGITAL/  
[TUTO:ugur@gallium]$ mkdir TUTO_SYN  
[TUTO:ugur@gallium]$ cd TUTO_SYN  
[TUTO:ugur@gallium]$ cp /PROJ/TUTO/DIGITAL/VERILOG/TOP.v ./  
[TUTO:ugur@gallium]$ cp /PROJ/TUTO/DIGITAL/VERILOG/CNT_16.v ./  
[TUTO:ugur@gallium]$ cp /PROJ/TUTO/DIGITAL/VERILOG/CMP_16.v ./  
[TUTO:ugur@gallium]$ rc -gui
```

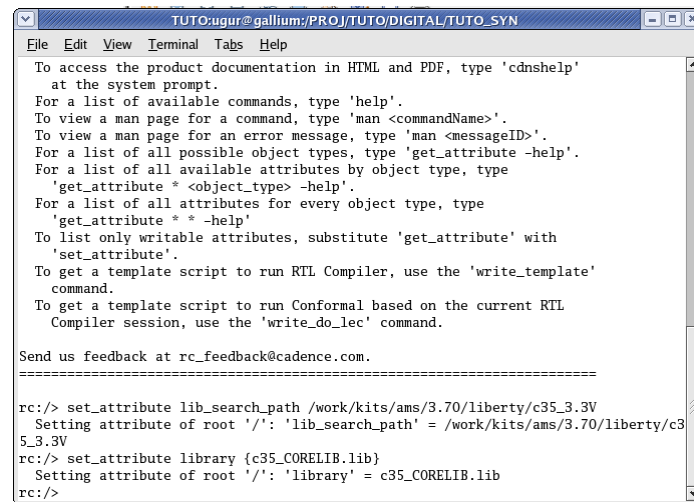
We can see that the synthesizer has two windows: one for command line input and one for graphical user interface. We will use the command line to enter all the commands and use the graphical user interface only to see the results.



In order to specify the path of the technology library, we must enter  
**set\_attribute lib\_search\_path /work/kits/ams/3.70/liberty/c35\_3.3V**



Next we must specify the name of the library. In order to do so, let's enter **set\_attribute library {c35\_CORELIB.lib}**

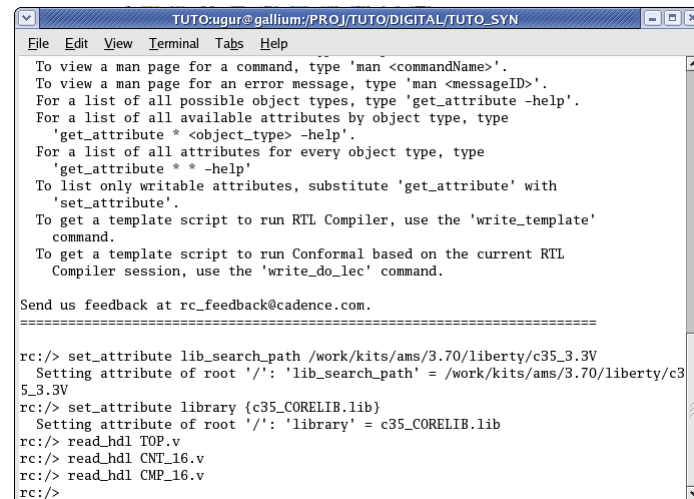


```
TUTO:ugur@ gallium:/PROJ/TUTO/DIGITAL/TUTO_SYN
File Edit View Terminal Tabs Help

To access the product documentation in HTML and PDF, type 'cdnshelp'
at the system prompt.
For a list of available commands, type 'help'.
To view a man page for a command, type 'man <commandName>'.
To view a man page for an error message, type 'man <messageID>'.
For a list of all possible object types, type 'get_attribute -help'.
For a list of all available attributes by object type, type
'get_attribute * <object_type> -help'.
For a list of all attributes for every object type, type
'get_attribute * * -help'.
To list only writable attributes, substitute 'get_attribute' with
'set_attribute'.
To get a template script to run RTL Compiler, use the 'write_template'
command.
To get a template script to run Conformal based on the current RTL
Compiler session, use the 'write_do_lec' command.

Send us feedback at rc_feedback@cadence.com.
=====
rc:/> set_attribute lib_search_path /work/kits/ams/3.70/liberty/c35_3.3V
Setting attribute of root '/': 'lib_search_path' = /work/kits/ams/3.70/liberty/c3
5_3.3V
rc:/> set_attribute library {c35_CORELIB.lib}
Setting attribute of root '/': 'library' = c35_CORELIB.lib
rc:/>
```

Now let's tell the synthesizer which verilog files to use. For that purpose we must enter **read\_hdl CNT\_16.v**  
**read\_hdl CMP\_16.v**  
**read\_hdl TOP.v**



```
TUTO:ugur@ gallium:/PROJ/TUTO/DIGITAL/TUTO_SYN
File Edit View Terminal Tabs Help

To view a man page for a command, type 'man <commandName>'.
To view a man page for an error message, type 'man <messageID>'.
For a list of all possible object types, type 'get_attribute -help'.
For a list of all available attributes by object type, type
'get_attribute * <object_type> -help'.
For a list of all attributes for every object type, type
'get_attribute * * -help'.
To list only writable attributes, substitute 'get_attribute' with
'set_attribute'.
To get a template script to run RTL Compiler, use the 'write_template'
command.
To get a template script to run Conformal based on the current RTL
Compiler session, use the 'write_do_lec' command.

Send us feedback at rc_feedback@cadence.com.
=====
rc:/> set_attribute lib_search_path /work/kits/ams/3.70/liberty/c35_3.3V
Setting attribute of root '/': 'lib_search_path' = /work/kits/ams/3.70/liberty/c3
5_3.3V
rc:/> set_attribute library {c35_CORELIB.lib}
Setting attribute of root '/': 'library' = c35_CORELIB.lib
rc:/> read_hdl TOP.v
rc:/> read_hdl CNT_16.v
rc:/> read_hdl CMP_16.v
rc:/>
```

Next let's specify the name of the top level by entering  
**elaborate TOP**

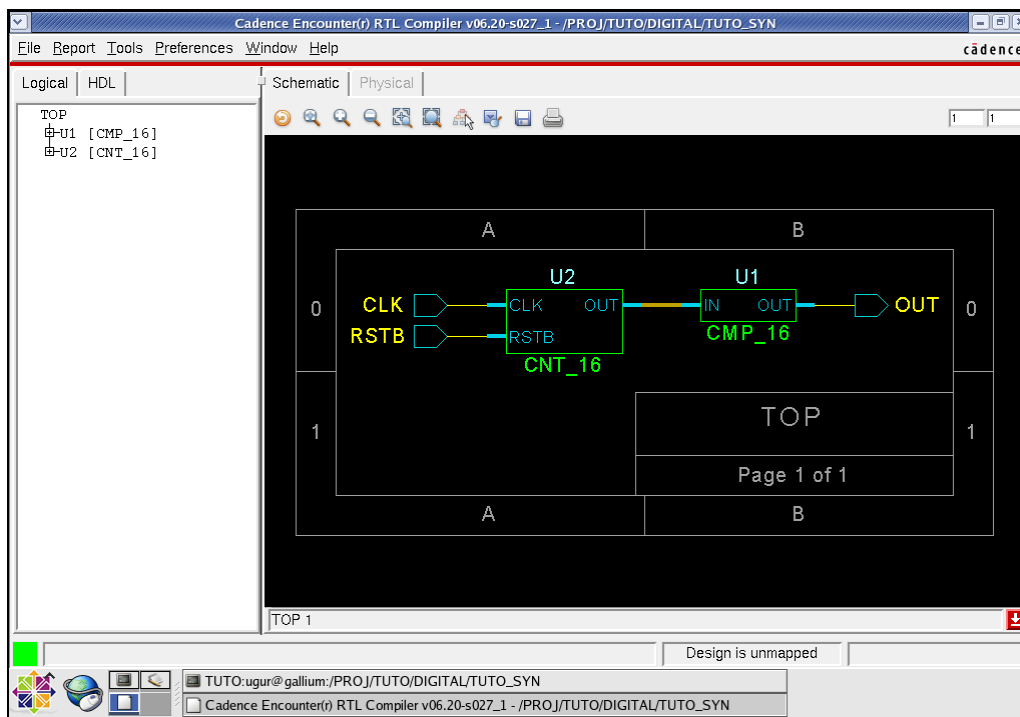
```
TUTO:ugur@gallium:/PROJ/TUTO/DIGITAL/TUTO_SYN
File Edit View Terminal Tabs Help

'get_attribute * <object_type> -help'.
For a list of all attributes for every object type, type
'get_attribute * * -help'
To list only writable attributes, substitute 'get_attribute' with
'set_attribute'.
To get a template script to run RTL Compiler, use the 'write_template'
command.
To get a template script to run Conformal based on the current RTL
Compiler session, use the 'write_do_lec' command.

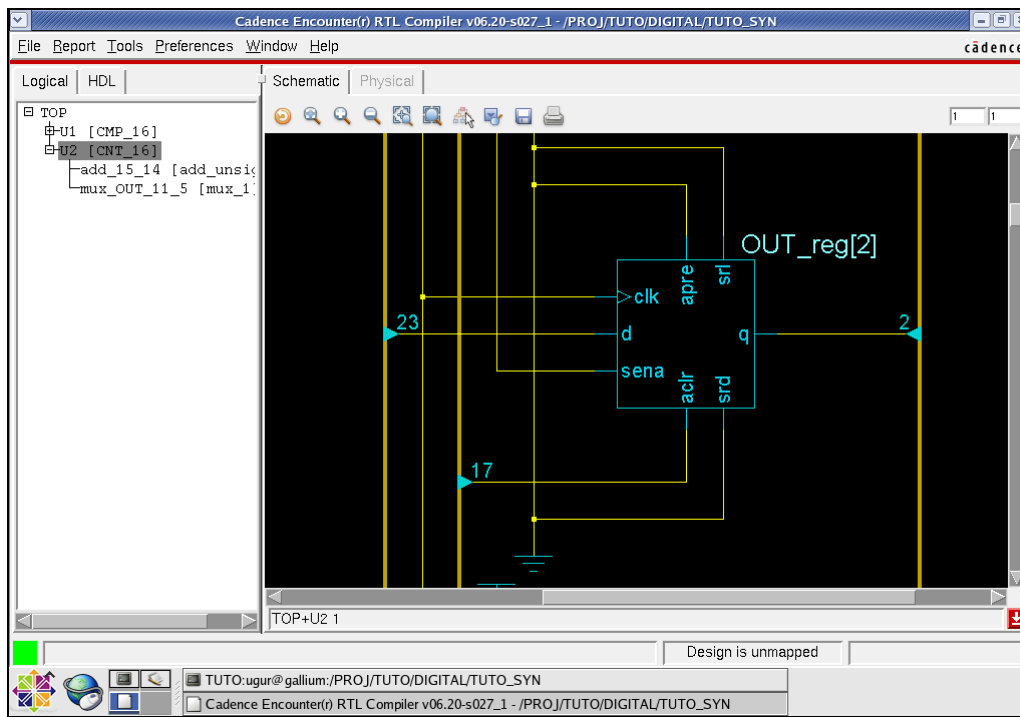
Send us feedback at rc_feedback@cadence.com.
=====

rc:/> set_attribute lib_search_path /work/kits/ams/3.70/liberty/c35_3.3V
Setting attribute of root '/': 'lib_search_path' = /work/kits/ams/3.70/liberty/c3
5_3.3V
rc:/> set_attribute library {c35_CORELIB.lib}
Setting attribute of root '/': 'library' = c35_CORELIB.lib
rc:/> read_hdl TOP.v
rc:/> read_hdl CNT_16.v
rc:/> read_hdl CMP_16.v
rc:/> elaborate TOP
Library has 147 usable logic and 88 usable sequential lib-cells.
Elaborating top-level block 'TOP' from file 'TOP.v'.
Done elaborating 'TOP'.
rc:/>
```

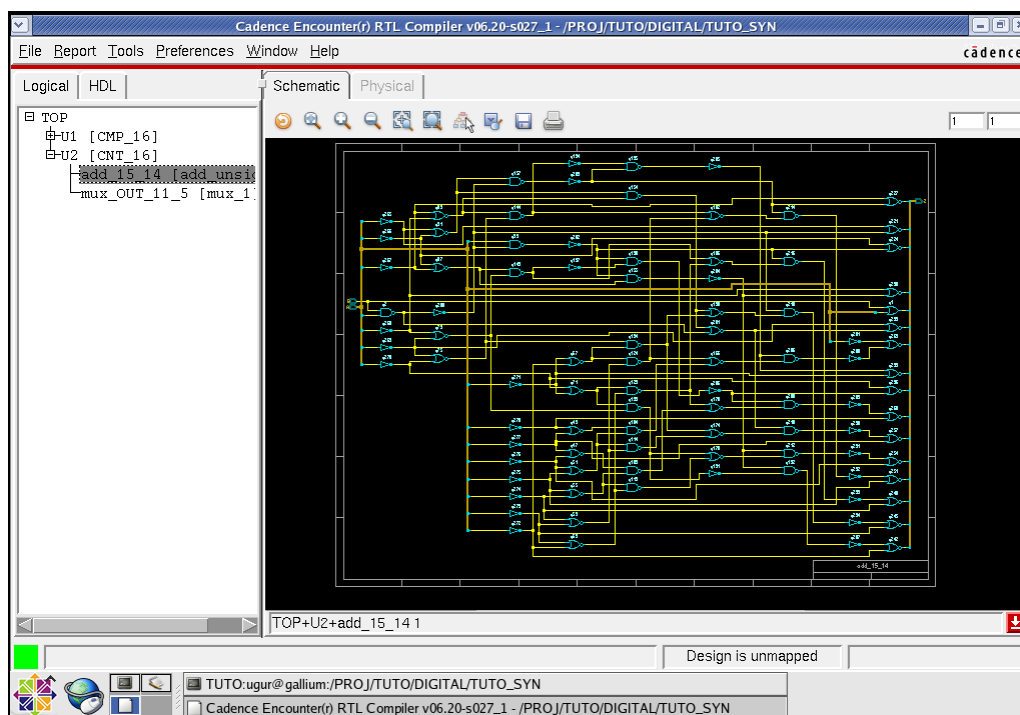
Now we can see that the design is elaborated to its most generic form.



Let's investigate the design. If we enter to CNT\_16 module, we can see that the D type flip flops are synthesized in their most general form, which means that the technological cells are not yet implemented. So, this design is technology independent for now.



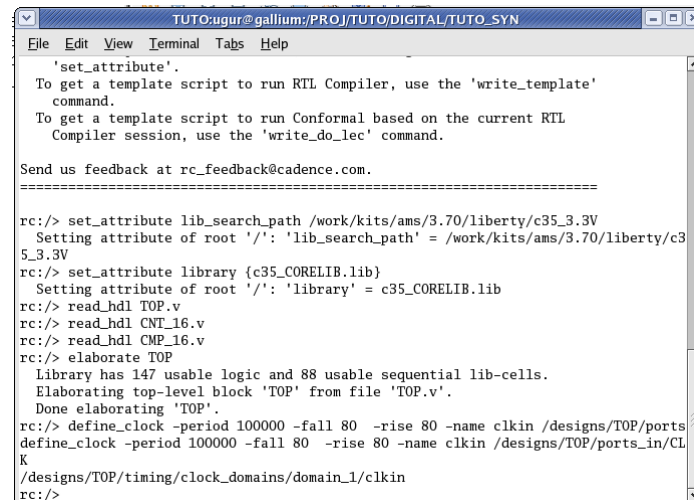
Let's investigate add\_15\_14 submodule. We see once again that the elaboration synthesized 16 bit adder in its most generic form, using only basic logical blocks (AND, OR, NOR etc.).



It is worth mentioning that RTL Compiler uses a somewhat interesting way to store design hierarchy. Every port, bus, clock, subdesign and constraints are stored as virtual folders accessible from RTL Compiler command window. For example, our TOP design is stored in

**/designs/TOP** folder and CNT\_16 is stored in **/designs/TOP/subdesigns/CNT\_16**. Also the top level input ports are stored in **/designs/TOP/ports\_in**. Now let's specify a 10MHz (100000ps) clock named **clkin** with 80ps rise and fall time which will be connected to CLK input of TOP module by entering:

**define\_clock -period 100000 -fall 80 -rise 80 -name clkin /designs/TOP/ports\_in/CLK**



```
TUTO:ugur@ gallium:/PROJ/TUTO/DIGITAL/TUTO_SYN
File Edit View Terminal Tabs Help

'set_attribute'.
To get a template script to run RTL Compiler, use the 'write_template'
command.
To get a template script to run Conformal based on the current RTL
Compiler session, use the 'write_do_lec' command.

Send us feedback at rc_feedback@cadence.com.
=====

rc:/> set_attribute lib_search_path /work/kits/ams/3.70/liberty/c35_3.3V
Setting attribute of root '/': 'lib_search_path' = /work/kits/ams/3.70/liberty/c3
5_3.3V
rc:/> set_attribute library {c35_CORELIB.lib}
Setting attribute of root '/': 'library' = c35_CORELIB.lib
rc:/> read_hdl TOP.v
rc:/> read_hdl CNT_16.v
rc:/> read_hdl CMP_16.v
rc:/> elaborate TOP
Library has 147 usable logic and 88 usable sequential lib-cells.
Elaborating top-level block 'TOP' from file 'TOP.v'.
Done elaborating 'TOP'.
rc:/> define_clock -period 100000 -fall 80 -rise 80 -name clkin /designs/TOP/ports
define_clock -period 100000 -fall 80 -rise 80 -name clkin /designs/TOP/ports_in/CL
K
/designs/TOP/timing/clock_domains/domain_1/clkin
rc:/>
```

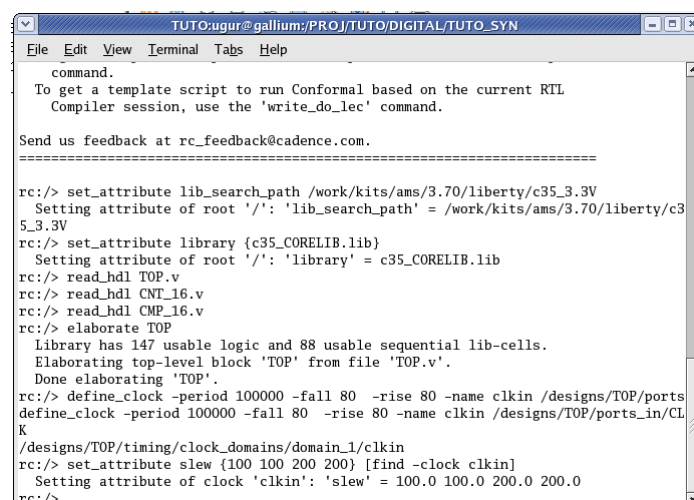
We can see that our new clock is defined under **/designs/TOP/timing/clock\_domains**. Now let's enter a slew attribute to clock named **clkin** with minimum rise time of 100ps, minimum fall time of 100ps, maximum rise time of 200ps and maximum fall time of 200ps by entering

**set\_attribute slew {100 100 200 200} [find -clock clkin]**

which is equivalent to

**set\_attribute slew {100 100 200 200} /designs/TOP/timing/clock\_domains/domain\_1/clkin**

The argument **[find -clock clkin]** simplifies our work by simply telling the program to find a clock named **clkin**, rather than entering the full path of the clock.



```
TUTO:ugur@ gallium:/PROJ/TUTO/DIGITAL/TUTO_SYN
File Edit View Terminal Tabs Help

command.
To get a template script to run Conformal based on the current RTL
Compiler session, use the 'write_do_lec' command.

Send us feedback at rc_feedback@cadence.com.
=====

rc:/> set_attribute lib_search_path /work/kits/ams/3.70/liberty/c35_3.3V
Setting attribute of root '/': 'lib_search_path' = /work/kits/ams/3.70/liberty/c3
5_3.3V
rc:/> set_attribute library {c35_CORELIB.lib}
Setting attribute of root '/': 'library' = c35_CORELIB.lib
rc:/> read_hdl TOP.v
rc:/> read_hdl CNT_16.v
rc:/> read_hdl CMP_16.v
rc:/> elaborate TOP
Library has 147 usable logic and 88 usable sequential lib-cells.
Elaborating top-level block 'TOP' from file 'TOP.v'.
Done elaborating 'TOP'.
rc:/> define_clock -period 100000 -fall 80 -rise 80 -name clkin /designs/TOP/ports
define_clock -period 100000 -fall 80 -rise 80 -name clkin /designs/TOP/ports_in/CL
K
/designs/TOP/timing/clock_domains/domain_1/clkin
rc:/> set_attribute slew {100 100 200 200} [find -clock clkin]
Setting attribute of clock 'clkin': 'slew' = 100.0 100.0 200.0 200.0
rc:/>
```



Now we must tell the program when the input will change after the rising edge of **clk** so that there are would be no hold violations. Let's tell the program that all inputs will change 1ns (1000ps) after the rising edge of **clk** by entering

**external\_delay -input 1000 -clock [find -clock clk] -edge\_rise [find /des\* -port ports\_in/\*]**

which is equivalent to

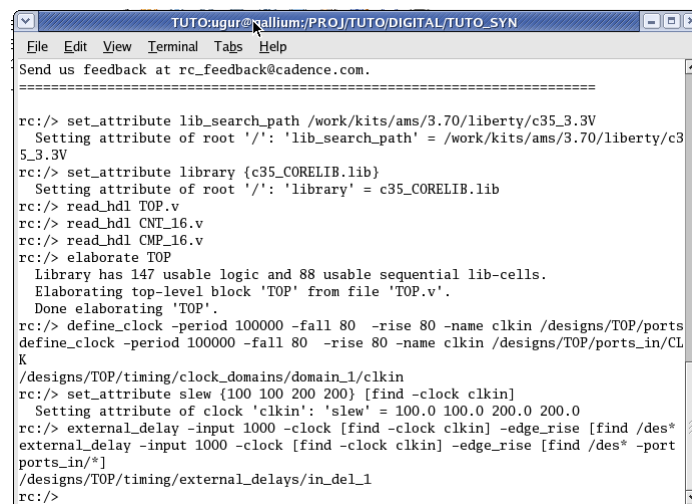
**external\_delay -input 1000 -clock [find -clock clk] -edge\_rise /designs/TOP/ports\_in/\***

Alternatively we can say only RSTB will change 1ns after the rising edge of **clk** by entering

**external\_delay -input 1000 -clock [find -clock clk] -edge\_rise [find -port RSTB]**

which is equivalent to

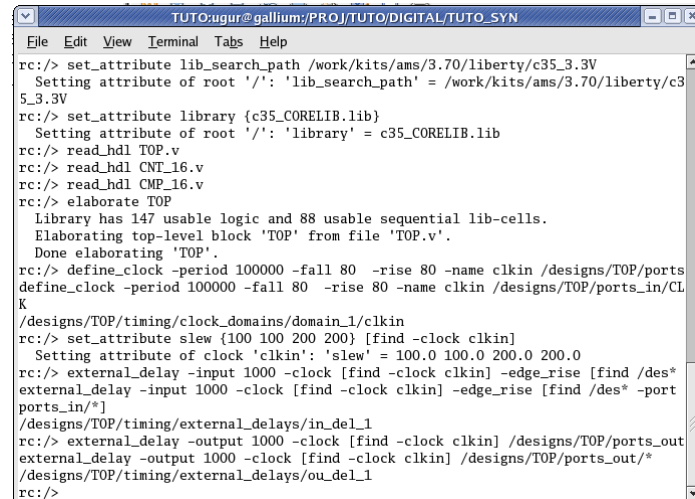
**external\_delay -input 1000 -clock [find -clock clk] -edge\_rise /designs/TOP/ports\_in/RSTB**



```
TUTO:ugur@xallium:/PROJ/TUTO/DIGITAL/TUTO_SYN
File Edit View Terminal Tabs Help
Send us feedback at rc_feedback@cadence.com.
=====
rc:/> set_attribute lib_search_path /work/kits/ams/3.70/liberty/c35_3.3V
Setting attribute of root '/': 'lib_search_path' = /work/kits/ams/3.70/liberty/c35_3.3V
rc:/> set_attribute library {c35_CORELIB.lib}
Setting attribute of root '/': 'library' = c35_CORELIB.lib
rc:/> read_hdl TOP.v
rc:/> read_hdl CNT_16.v
rc:/> read_hdl CMP_16.v
rc:/> elaborate TOP
Library has 147 usable logic and 88 usable sequential lib-cells.
Elaborating top-level block 'TOP' from file 'TOP.v'.
Done elaborating 'TOP'.
rc:/> define_clock -period 100000 -fall 80 -rise 80 -name clk /designs/TOP/ports_in/CLK
define_clock -period 100000 -fall 80 -rise 80 -name clk /designs/TOP/ports_in/CLK
/designs/TOP/timing/clock_domains/domain_1/clk
rc:/> set_attribute slew {100 100 200 200} [find -clock clk]
Setting attribute of clock 'clk': 'slew' = 100.0 100.0 200.0 200.0
rc:/> external_delay -input 1000 -clock [find -clock clk] -edge_rise [find /des*
external_delay -input 1000 -clock [find -clock clk] -edge_rise [find /des* -port
ports_in/*]
/designs/TOP/timing/external_delays/in_del_1
rc:/>
```

Now we must tell the program when the output data should be ready before the rising edge of **clk<sub>in</sub>**, so that there would be no setup violations. Let's tell the program that all outputs should be ready 1000ps before the rising edge of clk<sub>in</sub>.

**external\_delay -output 1000 -clock [find -clock clk<sub>in</sub>] -edge\_rise /designs/TOP/ports\_out/\***



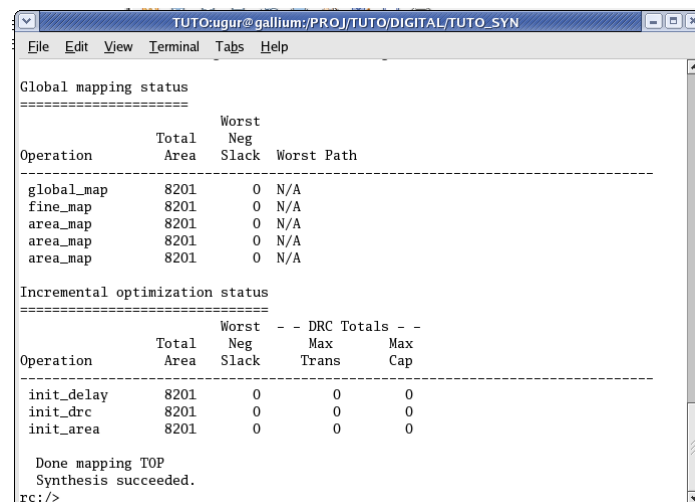
```

TUTO:ugur@ gallium:/PROJ/TUTO/DIGITAL/TUTO_SYN
File Edit View Terminal Tabs Help
rc:/> set_attribute lib_search_path /work/kits/ams/3.70/liberty/c35_3.3V
Setting attribute of root '/': 'lib_search_path' = /work/kits/ams/3.70/liberty/c35_3.3V
rc:/> set_attribute library {c35_CORELIB.lib}
Setting attribute of root '/': 'library' = c35_CORELIB.lib
rc:/> read_hdl TOP.v
rc:/> read_hdl CNT_16.v
rc:/> read_hdl CMP_16.v
rc:/> elaborate TOP
Library has 147 usable logic and 88 usable sequential lib-cells.
Elaborating top-level block 'TOP' from file 'TOP.v'.
Done elaborating 'TOP'.
rc:/> define_clock -period 100000 -fall 80 -rise 80 -name clkin /designs/TOP/ports
define_clock -period 100000 -fall 80 -rise 80 -name clkin /designs/TOP/ports_in/CLK
/designs/TOP/timing/clock_domains/domain_1/clkin
rc:/> set_attribute slew {100 100 200 200} [find -clock clkin]
Setting attribute of clock 'clkin': 'slew' = 100.0 100.0 200.0 200.0
rc:/> external_delay -input 1000 -clock [find -clock clkin] -edge_rise [find /des*
external_delay -input 1000 -clock [find -clock clkin] -edge_rise [find /des* -port
ports_in/*]
/designs/TOP/timing/external_delays/in_del_1
rc:/> external_delay -output 1000 -clock [find -clock clkin] /designs/TOP/ports_out
external_delay -output 1000 -clock [find -clock clkin] /designs/TOP/ports_out/*
/designs/TOP/timing/external_delays/ou_del_1
rc:/>

```

Now for the first time, we can start technology specific synthesis in which we will “map” the design to the technology. In order to do so, let's enter

**synthesize -to\_mapped**



```

TUTO:ugur@ gallium:/PROJ/TUTO/DIGITAL/TUTO_SYN
File Edit View Terminal Tabs Help

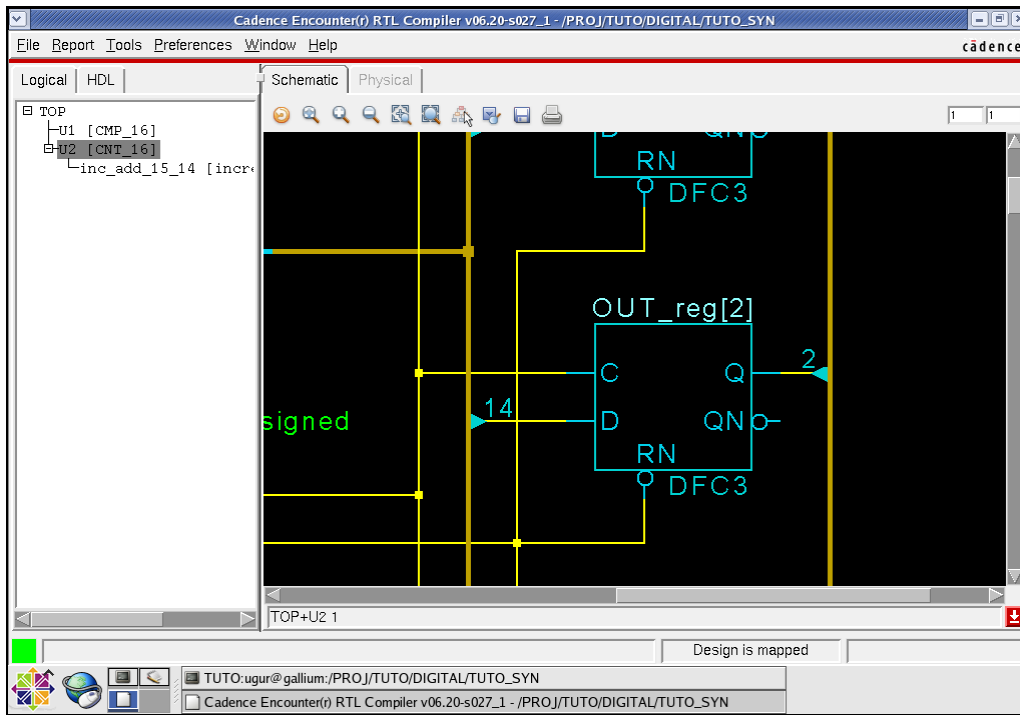
Global mapping status
=====
Operation      Total Area  Worst Neg Slack  Worst Path
-----
global_map     8201       0    N/A
fine_map       8201       0    N/A
area_map       8201       0    N/A
area_map       8201       0    N/A
area_map       8201       0    N/A

Incremental optimization status
=====
Operation      Total Area  Worst Neg Slack  DRC Totals  Max Trans  Max Cap
-----
init_delay     8201       0       0       0
init_drc       8201       0       0       0
init_area      8201       0       0       0

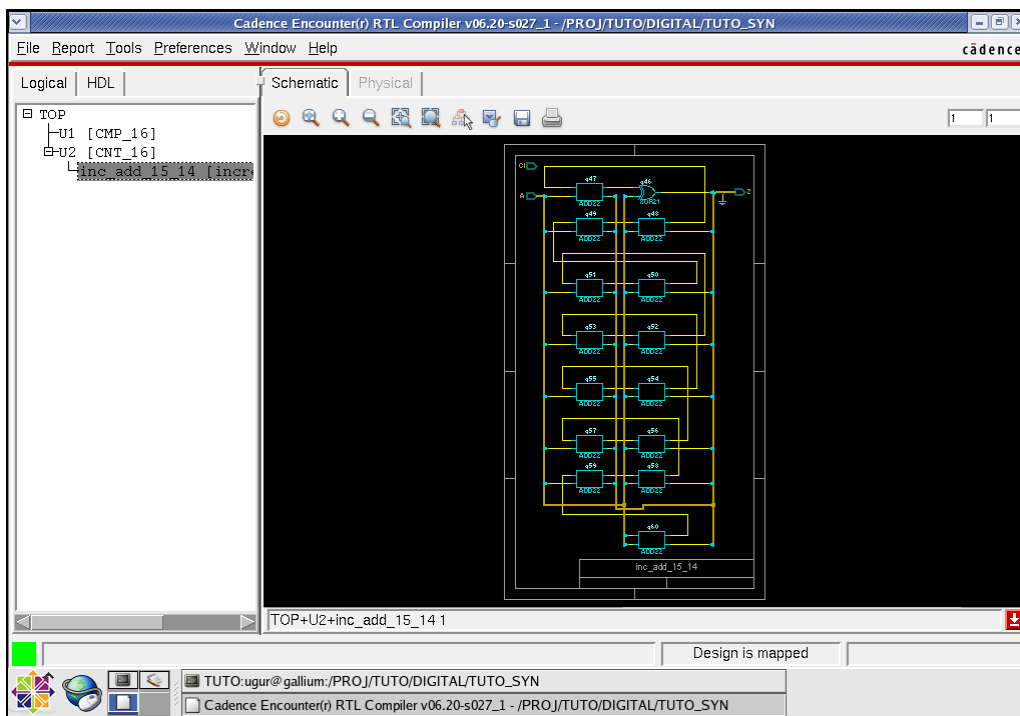
Done mapping TOP
Synthesis succeeded.
rc:/>

```

As we can see, synthesis succeeded. Now let's investigate what happened to our design from the graphical user interface. For example, let's see our D type flip flops. As we can see, generic D type flip flops are replaced by DFC3, a flip flop of AMS c35 library.



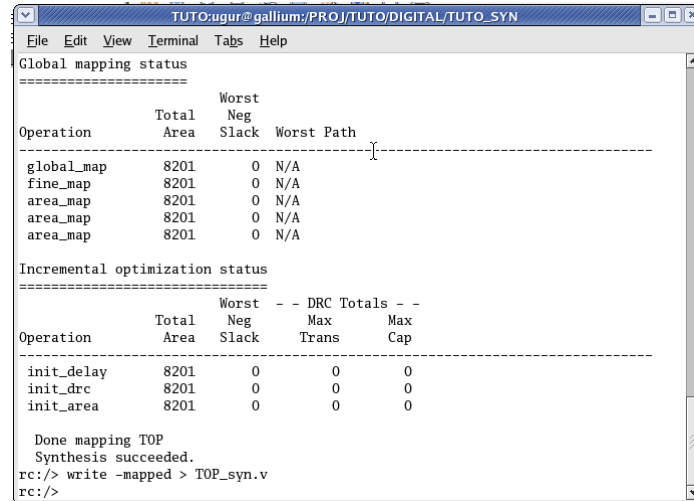
Now let's investigate our adder. We can see that the mapper found full adder blocks in the technology library and decided that they are more effective than basic gates of the elaborated design, so it replaced the basic gates with full adders.



In order to save the mapped design as an RTL level verilog file, let's enter

**write -mapped > TOP\_syn.v**

which will save the synthesized and mapped design as TOP\_syn.v



Global mapping status

Operation	Total Area	Worst Neg Slack	Worst Path
global_map	8201	0	N/A
fine_map	8201	0	N/A
area_map	8201	0	N/A
area_map	8201	0	N/A
area_map	8201	0	N/A

Incremental optimization status

Operation	Total Area	Worst Neg Slack	-- DRC Totals --	Max Trans	Max Cap
init_delay	8201	0	0	0	0
init_drc	8201	0	0	0	0
init_area	8201	0	0	0	0

Done mapping TOP  
Synthesis succeeded.  
rc:/> write -mapped > TOP\_syn.v  
rc:/>

If we open TOP\_syn.v file, we will see RTL level verilog code of top module and all submodules, such as:

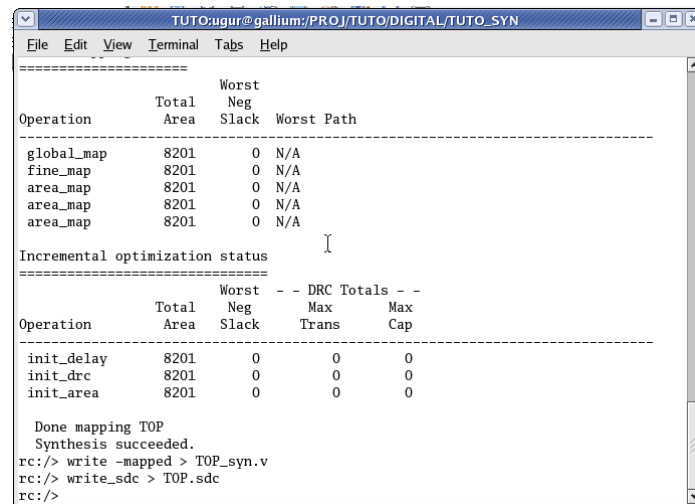
```
module increment_unsigned(A, CI, Z);
    input [15:0] A;
    input CI;
    output [16:0] Z;
    wire [15:0] A;
    wire CI;
    wire [16:0] Z;
    wire n_0, n_2, n_4, n_6, n_8, n_10, n_12, n_14;
    wire n_16, n_18, n_20, n_22, n_24, n_26;
    assign Z[0] = 1'b0;
    assign Z[16] = 1'b0;
    XOR21 g46(.A (n_26), .B (A[15]), .Q (Z[15]));
    ADD22 g47(.A (n_24), .B (A[14]), .CO (n_26), .S (Z[14]));
    ADD22 g48(.A (n_22), .B (A[13]), .CO (n_24), .S (Z[13]));
    ADD22 g49(.A (n_20), .B (A[12]), .CO (n_22), .S (Z[12]));
    ADD22 g50(.A (n_18), .B (A[11]), .CO (n_20), .S (Z[11]));
    ADD22 g51(.A (n_16), .B (A[10]), .CO (n_18), .S (Z[10]));
    ADD22 g52(.A (n_14), .B (A[9]), .CO (n_16), .S (Z[9]));
    ADD22 g53(.A (n_12), .B (A[8]), .CO (n_14), .S (Z[8]));
    ADD22 g54(.A (n_10), .B (A[7]), .CO (n_12), .S (Z[7]));
    ADD22 g55(.A (n_8), .B (A[6]), .CO (n_10), .S (Z[6]));
    ADD22 g56(.A (n_6), .B (A[5]), .CO (n_8), .S (Z[5]));
    ADD22 g57(.A (n_4), .B (A[4]), .CO (n_6), .S (Z[4]));
    ADD22 g58(.A (n_2), .B (A[3]), .CO (n_4), .S (Z[3]));
    ADD22 g59(.A (n_0), .B (A[2]), .CO (n_2), .S (Z[2]));
    ADD22 g60(.A (A[1]), .B (A[0]), .CO (n_0), .S (Z[1]));
```

endmodule

Now let's save our clock constraints as an SDC file to use later in the place and route process.

In order to do so, let's enter

**write\_sdc > TOP.sdc**



The written SDC file is as follows:

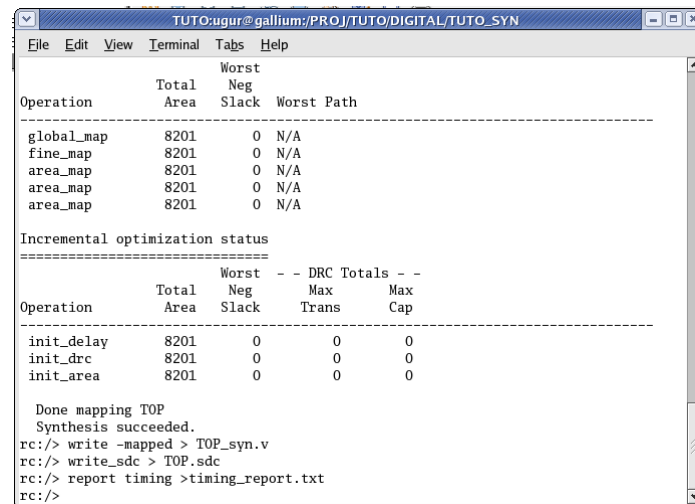
```
set sdc_version 1.5
```

```
# Set the current design
current_design TOP
```

```
create_clock -name "clkin" -add -period 100.0 -waveform {80.0
80.0} [get_ports CLK]
set_clock_transition -min 0.1 [get_clocks clkin]
set_clock_transition -max 0.2 [get_clocks clkin]
set_wire_load_mode "enclosed"
set_wire_load_selection_group "sub_micron" -library
"c35_CORELIB"
set_dont_use [get_lib_cells c35_CORELIB/BUSHD]
set_dont_use [get_lib_cells c35_CORELIB/TIE0]
set_dont_use [get_lib_cells c35_CORELIB/TIE1]
```

which is like a summary of the clock constraints we entered earlier.

Now let's see our timing reports. The timing report shows us if there is a timing violation and it also shows us the slack value, which is the excess time of our design considering our clock constraints. To write the timing report to the timing\_report.txt file, let's enter **report\_timing >timing\_report.txt**



If there were a timing violation, the terminal would say so, but let's check the timing report anyway:

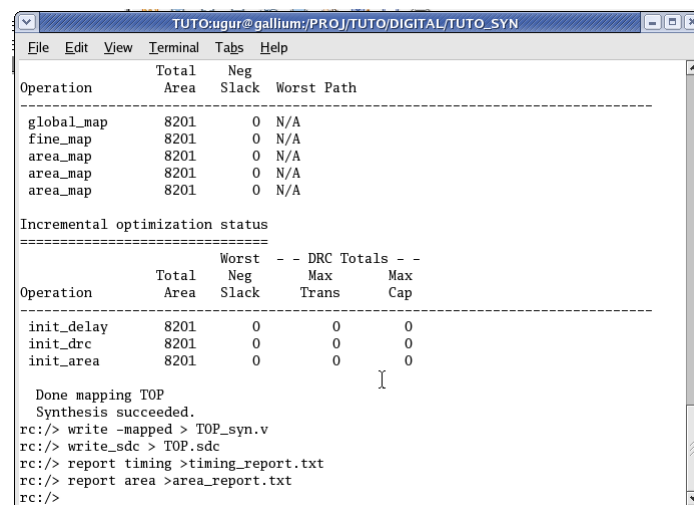
```
=====
Generated by:          Encounter(r) RTL Compiler v06.20-s027_1
Generated on:          Jan 22 2009  10:46:10 AM
Module:                TOP
Technology library:    c35_CORELIB 2.1
Operating conditions:  _nominal_ (balanced_tree)
Wireload mode:        enclosed
=====
```

Pin	Type	Fanout	Load (fF)	Slew (ps)	Delay (ps)	Arrival (ps)	
(clock clkln)	launch					80000	R
U2							
OUT_reg[0]/C				200		80000	R
OUT_reg[0]/Q	TFC3	2	49.3	222	+710	80710	F
inc_add_15_14/A[0]							
g60/B					+0	80710	
g60/CO	ADD22	1	23.2	194	+327	81037	F
g59/A					+0	81038	
g59/CO	ADD22	1	23.2	194	+298	81336	F
g58/A					+0	81336	
g58/CO	ADD22	1	23.2	194	+298	81635	F
g57/A					+0	81635	
g57/CO	ADD22	1	23.2	194	+298	81933	F
g56/A					+0	81933	

g56/CO	ADD22	1	23.2	194	+298	82232	F
g55/A					+0	82232	
g55/CO	ADD22	1	23.2	194	+298	82530	F
g54/A					+0	82531	
g54/CO	ADD22	1	23.2	194	+298	82829	F
g53/A					+0	82829	
g53/CO	ADD22	1	23.2	194	+298	83128	F
g52/A					+0	83128	
g52/CO	ADD22	1	23.2	194	+298	83426	F
g51/A					+0	83426	
g51/CO	ADD22	1	23.2	194	+298	83725	F
g50/A					+0	83725	
g50/CO	ADD22	1	23.2	194	+298	84023	F
g49/A					+0	84024	
g49/CO	ADD22	1	23.2	194	+298	84322	F
g48/A					+0	84322	
g48/CO	ADD22	1	23.2	194	+298	84621	F
g47/A					+0	84621	
g47/CO	ADD22	1	23.2	194	+298	84919	F
g46/A					+0	84919	
g46/Q	XOR21	1	15.2	226	+307	85226	F
inc_add_15_14/Z[15]							
OUT_reg[15]/D	DFC3				+0	85226	
OUT_reg[15]/C	setup			200	+1	85228	R
-----							
(clock clkin)	capture					180000	R
-----							
Timing slack : 94772ps							
Start-point : U2/OUT_reg[0]/C							
End-point : U2/OUT_reg[15]/D							

As we can see, there is a positive slack of 94.77ns, which means that our design is idle for 94.78ns. Since our clock was 100ns, we can say that the design calculates the output and satisfies our delay constraints in  $100 - 94.77 = 5.23\text{ns}$ , so we can roughly estimate that the design can work up to  $1 / 5.23\text{ns} = 191\text{MHz}$  clock frequency.

Next, let's see the area our design occupies. In order to do so, let's enter **report\_area >area\_report.txt**



Now let's see what the area report says:

```

=====
Generated by:      Encounter(r) RTL Compiler v06.20-s027_1
Generated on:      Jan 22 2009  10:46:33 AM
Module:           TOP
Technology library: c35_CORELIB 2.1
Operating conditions: _nominal_ (balanced_tree)
Wireload mode:    enclosed
=====

```

Instance	Cells	Cell Area	Net Area	Wireload
TOP	38	7589	612	10k (S)
U2	31	7098	261	10k (S)
inc_add_15_14	15	2166	126	10k (S)
U1	7	491	54	10k (S)

(S) = wireload was automatically selected

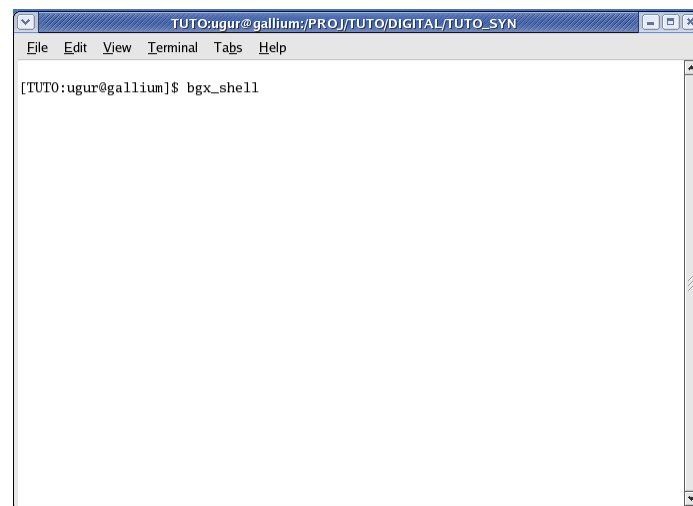
We can see that all of our design occupies an area of  $7589\mu\text{m}^2$ . We can also see that more than half of our area is occupied by D type flip flops, which are located in CNT\_16 (U2) excluding the adder ( $7098-2166=4932\mu\text{m}^2$ ).

Our work in RTL Compiler is now complete, and we can type **exit** to return to terminal screen. Now here comes the best part; there is a file called rc.cmd in the folder which includes

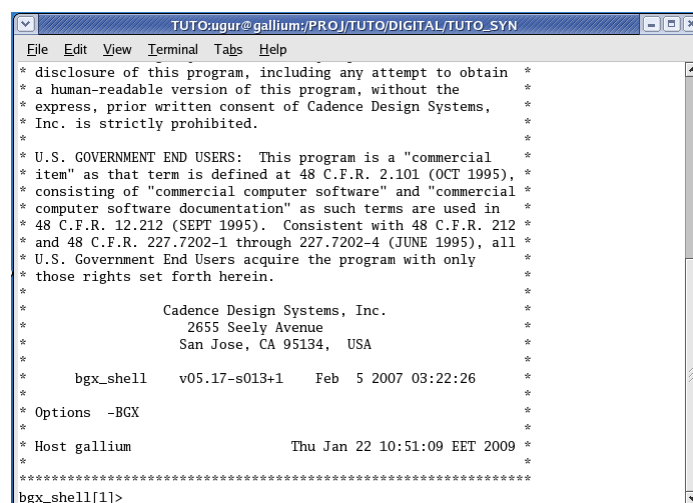


all of our commands we entered in the RTL Compiler command line. To do exactly the same things later, we can enter **source rc.cmd** and our commands in the rc.cmd file would be executed one by one.

One remaining issue with RTL Compiler is that it does not effectively write SDF files, which includes all the delays of cells and some timing checks such as setup and hold times of flip flops. Since SDF is an important file for post-synthesis simulation, we should create it with another program, called **buildgates**. Since we don't need graphical user interface to create an SDF file, we can start buildgates in shell mode (command window only). In order to do so, let's enter **bgx\_shell** in the same folder as RTL Compiler

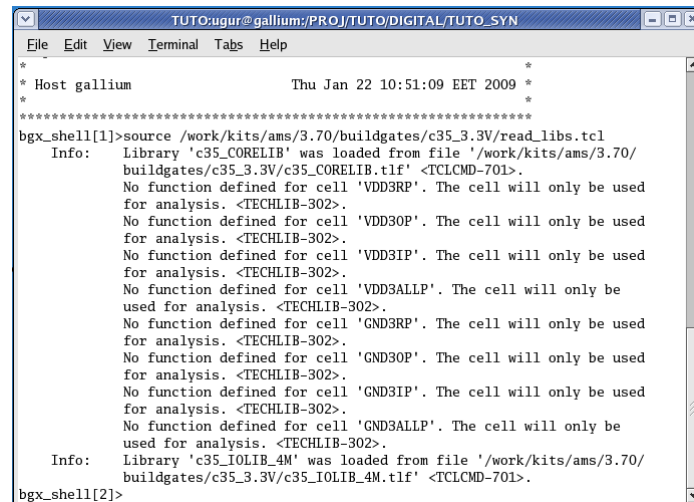


Command line of buildgates is like follows:



Thanks to AMS, the buildgates configuration is simplified with **readlibs.tcl** file located for each technology in buildgates folder of the design kit. In order to use the file, we should simply enter

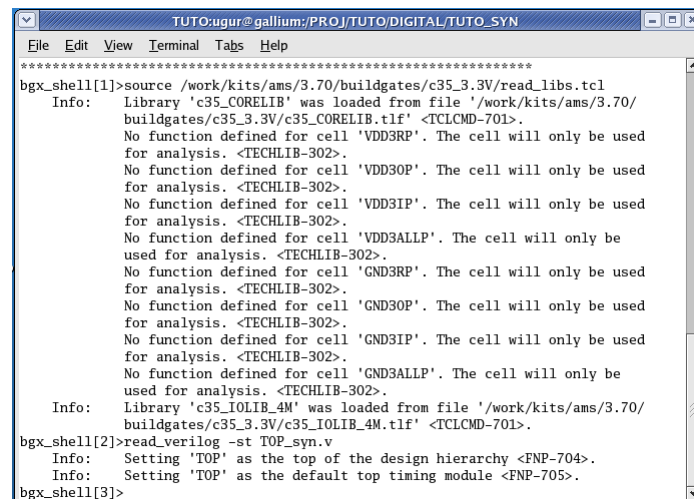
**source /work/kits/ams/3.70/buildgates/c35\_3.3V/read\_libs.tcl**



```
TUTO:ugur@ gallium:/PROJ/TUTO/DIGITAL/TUTO_SYN
File Edit View Terminal Tabs Help
* Host gallium Thu Jan 22 10:51:09 EET 2009 *
*****
bgx_shell[1]>source /work/kits/ams/3.70/buildgates/c35_3.3V/read_libs.tcl
Info: Library 'c35_CORELIB' was loaded from file '/work/kits/ams/3.70/
buildgates/c35_3.3V/c35_CORELIB.tlf' <TCLCMD-701>.
No function defined for cell 'VDD3RP'. The cell will only be used
for analysis. <TECHLIB-302>.
No function defined for cell 'VDD3OP'. The cell will only be used
for analysis. <TECHLIB-302>.
No function defined for cell 'VDD3IP'. The cell will only be used
for analysis. <TECHLIB-302>.
No function defined for cell 'VDD3ALLP'. The cell will only be
used for analysis. <TECHLIB-302>.
No function defined for cell 'GND3RP'. The cell will only be used
for analysis. <TECHLIB-302>.
No function defined for cell 'GND3OP'. The cell will only be used
for analysis. <TECHLIB-302>.
No function defined for cell 'GND3IP'. The cell will only be used
for analysis. <TECHLIB-302>.
No function defined for cell 'GND3ALLP'. The cell will only be
used for analysis. <TECHLIB-302>.
Info: Library 'c35_IOLIB_4M' was loaded from file '/work/kits/ams/3.70/
buildgates/c35_3.3V/c35_IOLIB_4M.tlf' <TCLCMD-701>.
bgx_shell[2]>
```

Now let's import our mapped design and make its TOP module the top timing module and top hierarchical module. In order to do that, let's enter

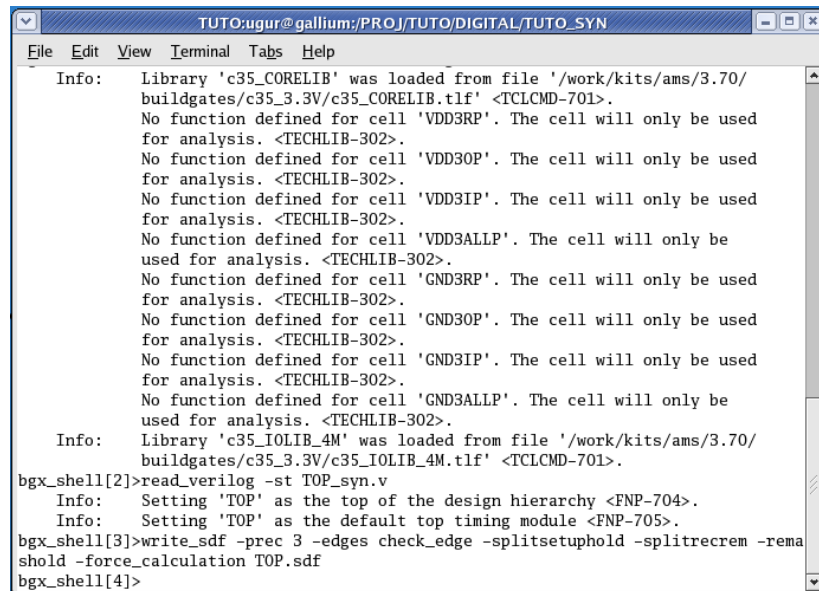
**read\_verilog -st TOP\_syn.v**



```
TUTO:ugur@ gallium:/PROJ/TUTO/DIGITAL/TUTO_SYN
File Edit View Terminal Tabs Help
*****
bgx_shell[1]>source /work/kits/ams/3.70/buildgates/c35_3.3V/read_libs.tcl
Info: Library 'c35_CORELIB' was loaded from file '/work/kits/ams/3.70/
buildgates/c35_3.3V/c35_CORELIB.tlf' <TCLCMD-701>.
No function defined for cell 'VDD3RP'. The cell will only be used
for analysis. <TECHLIB-302>.
No function defined for cell 'VDD3OP'. The cell will only be used
for analysis. <TECHLIB-302>.
No function defined for cell 'VDD3IP'. The cell will only be used
for analysis. <TECHLIB-302>.
No function defined for cell 'VDD3ALLP'. The cell will only be
used for analysis. <TECHLIB-302>.
No function defined for cell 'GND3RP'. The cell will only be used
for analysis. <TECHLIB-302>.
No function defined for cell 'GND3OP'. The cell will only be used
for analysis. <TECHLIB-302>.
No function defined for cell 'GND3IP'. The cell will only be used
for analysis. <TECHLIB-302>.
No function defined for cell 'GND3ALLP'. The cell will only be
used for analysis. <TECHLIB-302>.
Info: Library 'c35_IOLIB_4M' was loaded from file '/work/kits/ams/3.70/
buildgates/c35_3.3V/c35_IOLIB_4M.tlf' <TCLCMD-701>.
bgx_shell[2]>read_verilog -st TOP_syn.v
Info: Setting 'TOP' as the top of the design hierarchy <FNP-704>.
Info: Setting 'TOP' as the default top timing module <FNP-705>.
bgx_shell[3]>
```

Now let's get to the SDF part. For flawless SDF annotation in NCLaunch, the command we should enter is

**write\_sdf -prec 3 -edges check\_edge -splitsetuphold -splitrecrem -remashold -force\_calculation TOP.sdf**



```
TUTO:ugur@gallium:/PROJ/TUTO/DIGITAL/TUTO_SYN
File Edit View Terminal Tabs Help
Info: Library 'c35_CORELIB' was loaded from file '/work/kits/ams/3.70/
buildgates/c35_3.3V/c35_CORELIB.tlf' <TCLCMD-701>.
No function defined for cell 'VDD3RP'. The cell will only be used
for analysis. <TECHLIB-302>.
No function defined for cell 'VDD30P'. The cell will only be used
for analysis. <TECHLIB-302>.
No function defined for cell 'VDD3IP'. The cell will only be used
for analysis. <TECHLIB-302>.
No function defined for cell 'VDD3ALLP'. The cell will only be
used for analysis. <TECHLIB-302>.
No function defined for cell 'GND3RP'. The cell will only be used
for analysis. <TECHLIB-302>.
No function defined for cell 'GND30P'. The cell will only be used
for analysis. <TECHLIB-302>.
No function defined for cell 'GND3IP'. The cell will only be used
for analysis. <TECHLIB-302>.
No function defined for cell 'GND3ALLP'. The cell will only be
used for analysis. <TECHLIB-302>.
Info: Library 'c35_IOLIB_4M' was loaded from file '/work/kits/ams/3.70/
buildgates/c35_3.3V/c35_IOLIB_4M.tlf' <TCLCMD-701>.
bgx_shell[2]>read_verilog -st TOP_syn.v
Info: Setting 'TOP' as the top of the design hierarchy <FNP-704>.
Info: Setting 'TOP' as the default top timing module <FNP-705>.
bgx_shell[3]>write_sdf -prec 3 -edges check_edge -splitsetuphold -splitrecrem -rema
shold -force_calculation TOP.sdf
bgx_shell[4]>
```

These parameters are necessary for the complete matching of delays in the SDF with the delays defined in the technology file. The meanings of these parameters are:

- prec 3:** Use 3 digits floating point in delay calculation
- edges check\_edge:** Check which edge delays are defined in technology file and calculate only those edges.
- splitsetuphold:** Don't use SETUPHOLD delay and define SETUP and HOLD delays separately.
- splitrecrem:** Don't use RECREM delay and define RECOVER and REMOVAL delays separately.
- remashold:** Define REMOVAL delays as HOLD delays
- force\_calculation:** Calculate all delays

After the SDF file is created, we can exit from buildgates by entering **exit**. Similar to **rc.cmd**, the folder now has **ac\_shell.cmd** which can later be used with **source ac\_shell.cmd** command.

Let's summarize what we have done until this point:

1. We started RTL Compiler and defined the technology libraries
2. We imported our behavioral verilog files into RTL Compiler and elaborated top module
3. We defined clock constraints
4. We defined input and output delays
5. We synthesized our design and mapped it to the technology
6. We exported our mapped design, SDC (clock constraints) file, timing and area reports
7. We extracted SDF (Standart Delay Format) file from the mapped verilog file by buildgates

Now we are ready to make port-synthesis simulation. In order to do that, we need a testbench module. For example, let's consider the testbench used in behavioral simulation:

```
`timescale 1ns/1ps

module TEST_TOP;

reg CLK, RSTB;
wire OUT;

initial begin
    RSTB = 0;
    CLK = 0;
    #3 RSTB = 1;
    #10000 $finish;
end

always #5 CLK = ~CLK;

TOP U1(CLK, RSTB, OUT);

endmodule
```

The testbench should be changed in such a way that:

1. Since behavioral simulation is technology independent and since in the behavioral simulation, the design is assumed to have zero delay, clock frequency in behavioral testbench is not a problem. However, mapped verilog file has real cells with non-zero delays defined in the SDF file so the clock frequency should be changed according to the clock constraint defined in RTL Compiler, which is 100ns.
2. The SDF file should be annotated, telling which subblock the SDF data belongs to.

3. The cells used in the mapped verilog file is technology dependent so a verilog file which -for example- tells the simulator that DFC3 is a D type flip flop with active-low asynchronous reset, is needed.

The resulting testbench is:

```
`timescale 1ns/1ps
`include "/work/kits/ams/3.70/verilog/c35b4/c35_CORELIB.v"
`include "/work/kits/ams/3.70/verilog/udp.v"

module TEST_TOP;

reg CLK, RSTB;
wire OUT;

initial $sdf_annotate("TOP.sdf",U1);

initial begin
    RSTB = 0;
    CLK = 0;
    #30 RSTB = 1;
    #100000 $finish;
end

always #50 CLK = ~CLK;

TOP U1(CLK, RSTB, OUT);

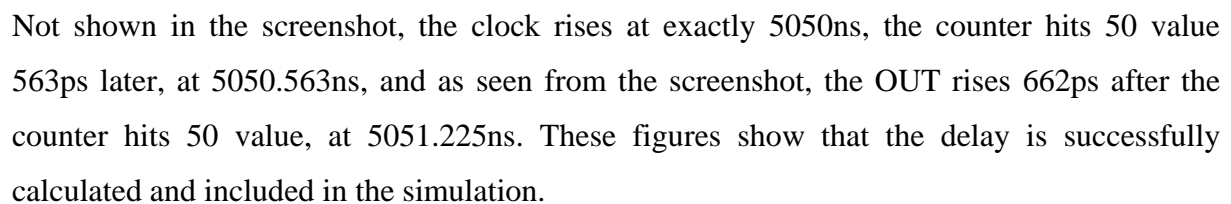
endmodule
```

One last thing to do is to add

```
`timescale 1ns/1ps
```

line to the beginning of the TOP\_syn.v file because the mapped verilog file lacks a timescale command.

The screenshot shows the Waveform 1 - SimVision interface. The main window displays a timing diagram with a clock signal (CLK) and a data signal (DUT[15:0]). The time scale is in picoseconds (ps), ranging from 0 to 100,000,000 ps. A cursor is positioned at Time A = 5,051,225ps. The left sidebar shows the signal list with DUT[15:0] selected. The bottom status bar indicates '1 object selected'.



Not shown in the screenshot, the clock rises at exactly 5050ns, the counter hits 50 value 563ps later, at 5050.563ns, and as seen from the screenshot, the OUT rises 662ps after the counter hits 50 value, at 5051.225ns. These figures show that the delay is successfully calculated and included in the simulation.