EE/CprE 465 -- Fall 2015

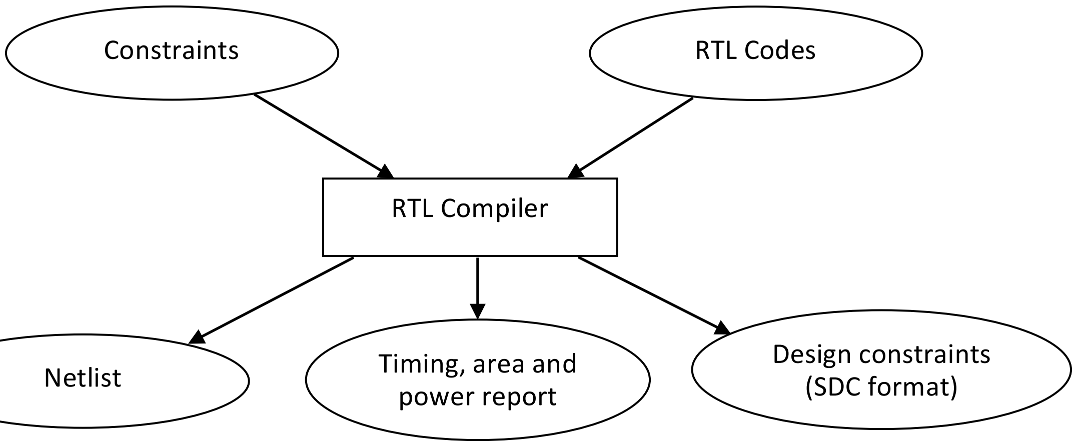
# Lab 5: Synthesis Constraints

The synthesis procedure depends to a large extent on how we set the constraints. Actually, setting the constraints is the job of architecture or system engineers, because it is them who design the whole system and who should know which signal should be ready at what time. By setting constraints, we emulate the real working environment for our design. Our task is to make sure the chip will not fail in this real working environment after it is manufactured. Otherwise, we will pay the expensive cost to re-design and re-manufacture the chip. In this lab, we will discover how the constraints can affect the synthesis result. There are many kinds of constraints, but in this lab we only cover the most important and most common used ones because of limitation on time.

The objective of this lab is to get a hands-on experience with:

1. Setting up constraints for synthesis
2. Circuit timing, area and power analysis

The inputs and outputs of RTL Compiler are described in the diagram below:


Constraints
RTL Codes
RTL Compiler
Netlist
Timing, area and power report
Design constraints (SDC format)

## Lab Tasks:

1. Exploring the effects of various constraints on the timing.

   Step 1: The file named design.sdc located in "scripts" directory contains all of the constraints we applied to our design in the last lab. The file named run_synth.tcl located in "scripts" directory is the main script which drives the software to run synthesis. In this file, "read_sdc ./scripts/design.sdc" is a command to load the constraints. Now before you start, please delete the line "read_sdc ./scripts/design.sdc" and everything below it in your run_synth.tcl file, because in this lab we will set up the constraints step by step manually to see what will happen in our synthesis result.

   Step 2: Launch the synthesis software RTL Compiler and run the script named run_synth.tcl as we did in Lab 4. The software will stop after "elaborate" step in the run_synth.tcl file, because we already deleted everything below it.

   Step 3: Then in your software's command window, type the commands below.
   To set time_unit and load unit, please use these commands:

```
dc:: set_time_unit –picoseconds
dc:: set_load_unit –picofarads
```

Now, begin setting of timing constraints:

1) To set clock period and to name the clock signal as clk, please use this command:

```
define_clock -period 400 -name clk [clock_ports] -rise 50 -fall 50
```

2) To set clock network latency, please use this command:

```
set_attribute clock_network_late_latency 1 clk
```

3) To set clock source latency, please use this command:

```
set_attribute clock_source_early_latency 1.5 clk
```

4) For setup time violation checking, please use this command to set clock uncertainty:

```
set_attribute clock_setup_uncertainty {100 50} clk
```

5) For clock transition, to set minimum rise value to 100, minimum fall value to 110, maximum rise value to 110, and maximum fall value to 120 on clk, use this command:

```
set_attribute slew {100 110 110 120}  [find / -clock clk]
```

6) Now you can review what you have done for the clock constraints by:

```
report clocks
```

7) To set all input delay, please use this command:

```
external_delay -clock clk -input 200 -name in_delay /designs/ALT_MULTADD/ports_in/*
```

   Please note, ALT_MULTADD is the Verilog design module name, if your module has different name, please replace it with yours.

8) To set all output delay, please use this command:

```
external_delay -clock clk -output 200 -name out_delay /designs/ALT_MULTADD/ports_out/*
```

9) To set external driver, please use this command:

```
set_attribute external_driver [find [find / -libcell MUX2ND2] -libpin ZN] /designs/ALT_MULTADD/ports_in/*
```

   Please note, for each input port, here we set the ZN port of a MUX to be the driver.

10) To set output load, please use this command:

```
set_attribute external_pin_cap 2 /designs/ALT_MULTADD/ports_out/*
```

11) To specify the maximum fanout for a net connected to an output port, please use this command:

```
set_attribute max_fanout 10 /designs/*
```

<u>Step 4:</u> Type the following command in the command window to begin synthesis:

```
synthesize -to_mapped -effort high
```

<u>Step 5:</u> Type the following commands to generate various reports:

```
report area > ${OUTPUT_DIR}/area.rpt
report gates > ${OUTPUT_DIR}/gates.rpt
report timing > ${OUTPUT_DIR}/timing.rpt
report timing -lint > ${OUTPUT_DIR}/lint.rpt
report summary > ${OUTPUT_DIR}/summary.rpt
report power > ${OUTPUT_DIR}/power.rpt
```

Please note ${OUTPUT_DIR} is a variable and its value is set to "run_dir", so all of your reports are stored in that directory. Please find them.

Step 6: Type the following commands to store your results:

```
write -mapped > ${OUTPUT_DIR}/jpeg_mapped.v
write_script > ${OUTPUT_DIR}/jpeg_mapped.g
write_sdc > ${OUTPUT_DIR}/jpeg_mapped.sdc
```

Now, please find the timing report in your "run_dir" directory and study it. You may find the document How_to_read_timing_report.pdf posted in Lab 4 to be very useful. Then for every single timing constraint you have set in Step 3 above, please explain what it means and how it affects the timing calculation. To perform this task, please follow this process:

a. Change only one timing constraint value in Step 3, re-run synthesis and get timing report. For your convenience, you may directly put those constraint setting commands into your run_synth.tcl below "elaborate". For second time running, just modify the constraint's value in this file and re-run this script. In this way, you do not need to repeatedly type everything again in the command window.
b. Compare the timing report with the original one, and report the differences.
c. Explain what the constraint means and why there are such differences in the timing report.

The meaning of each constraint can be found in the document RTL_Compiler_Constraints_and_STA.pdf posted. You may also consult your textbook, search online or ask our TA. To explain the meaning of various constraints, you may need to draw timing diagrams to illustrate the concept (like Figure 1 in How_to_read_timing_report.pdf and Figure 1-11 in RTL_Compiler_Constraints_and_STA.pdf).

2. Controlling area/power tradeoff.

If you set power constraints, RTL Compiler performs timing, area, and power optimization simultaneously. The tool tries to meet timing constraints, while balancing area and power. By allowing the area to increase, you may get better power results. To control the tradeoff between area and power, you can use:

```
set_attribute power_optimization_effort  <level>
```

where <level> can be set to low, medium, or high. By default, this attribute is set to medium, which allows power optimization to happen at a cost of some area increase (up to 5% or 10% depending on library, design and timing constraints). When you set the effort to low, power can be optimized with no or very minimal area increase (below 1%) compared to a synthesis run without power optimization enabled. When you set the effort to high, the tool will aggressively improve power without much consideration for area impact.

So, please set power constraint to be low/high/medium, and report what you find in the power and area reports that located in "run_dir" and briefly explain.