UE VLSI-2 TP: RTL Compiler

Matthieu Tuna

matthieu.tuna@gmail.com

1. Set the environment

To be able to run Cadence tools, you must setup the environment. Add to your environment:

```
> export LM_LICENSE_FILE=30000@licences.cnfm.fr
> export PATH=$PATH:/users/soft/opus/2008.bis/Linux/RC-8.1/tools/bin
```

2. Load the libraries

The libraries are in this directory: /users/outil/indus/techno/backend/

There are two types available **_Best.lib** and **_Worst.lib**:

/users/outil/indus/techno/backend/cmos_120nm_clock_Best.lib
/users/outil/indus/techno/backend/cmos_120nm_core_Best.lib
/users/outil/indus/techno/backend/cmos_120nm_filler_Best.lib
/users/outil/indus/techno/backend/cmos_120nm_io_3v3_Best.lib
/users/outil/indus/techno/backend/cmos_120nm_io_Best.lib

/users/outil/indus/techno/backend/cmos_120nm_clock_Worst.lib
/users/outil/indus/techno/backend/cmos_120nm_core_Worst.lib
/users/outil/indus/techno/backend/cmos_120nm_filler_Worst.lib
/users/outil/indus/techno/backend/cmos_120nm_io_3v3_Worst.lib
/users/outil/indus/techno/backend/cmos_120nm_io_Worst.lib

Explain the difference between these two different types.
Which one is needed for the synthesis step ? Explain your choice.
Pick up a timing arc in the WC lib and the same timing arc in the BC, what is the value in the first library and in the second library. Add in your report a figure of the cell and the corresponding timing arc.

Open a ".tcl" file to store the RTLC commands. The command in RTLC to specify the library to be used is:

```
> set_attribute library library.lib
```

3. Load the design

The design is in the directory ~tuna/ue_indus/tp/rtl/zigbee_vhdl_src/
in RTLC you to load the design, use:

```
> set rtl [list \
>    ~tuna/ue_vlsi2/rtl/mips_32_1p_mul_div.vhd   \
     >         ]
>
> read_hdl -vhdl $rtl
```

4. Elaborate

To elaborate the design enter the following command:

```
> elaborate MIPS_32_1P_MUL_DIV
```

The "elaborate" step does not synthesize the design, we will synthesize the design later on. What is the difference between elaboration and synthesis ?

You can now navigate throughout the design using the GUI and the unix-like database. To get the GUI opened, you have to add the option “-gui” when you launch rc: rc -gui .

5. Check design

GIGO: Garbage In-Garbage Out
If the loaded design is incorrect the generated netlist will also be incorrect. Just after elaboration you can check if everything is ok:

```
> check_design
```

Is everything OK ? Please explain what are the problems. The most important one is the 'Unresolved References', what is a missing the model ?

6. Synthesis

```
> synthesize -to_mapped
```

Check the design to see if everything is fine.

7. Reset

This signal should be handled carefully. What kind of reset is used in this design ? Synchronous or asynchronous ? Please explain the differences between both and which one is used in the design. What is a reset synchronizer, please explain why a circuit should have a reset synchronizer ? Is there any reset synchronizer ? Code a reset synchronizer for asynchronous reset and add it to the design. Add in your report the figure and the waveforms of your reset synchronizer.

8. Reporting

The command:

```
> report
```

generates various reports such as: area, timing, datapath, clock_gating etc.
Report the area:

```
> report area
```

What is the area of the design ?
How many cells are used ?
Which type of wireload model is used ? Why ? What are the other types ?
What wireload model are used for each hierarchy ? Why ?

Let's have a look at the timing:

```
> report timing
```

This command print the timing on the longest path. Examine the path (start point / end point etc.) and explain it.
What is the timing of the path ?
What is the supposed max frequency the design is able to run ?

You can see at the end of the report the following comment:

```
> Timing slack :  UNCONSTRAINED
```

What is a 'timing slack' ?
Why it is 'UNCONSTRAINED' ?

In order to check if the design is correctly constrained run:

```
> report timing -lint
```

This command categorized the errors in 3 types, explain those types.

8/ Constraints

Open a file .sdc where we will write the timing constraints.

a/ Reg-To-Reg

CLOCK

The port CK is the clock, the frequency of the clock should be 500MHz clock. Please write the corresponding SDC command.

Add in the RTLC script the command:

> read_sdc mips.sdc

Is the report timing -lint better ?
Redo the synthesis with the new constraints and report the timing.

Why does the report show the clock now ?
What is the worst path ?
What is the timing ?
Is the slack positive ?

Report the 10 worst paths.

What is a launch clock ?
What is a capture clock ?

What is the problem with your timing path in this case where the capture clock is another clock: CLOCK2 ? Why did the tool choose this timing between the launch clock and the capture clock ?

How would you solve the problem ?
Hint: between two asynchronous clock domains the paths in between are usually not timed, hence, considered as false paths.

If a CLOCK2 would come from a clock divider which will generate CLOCK2 from CLOCK, therefore CLOCK2 is generated from CLOCK: what is the corresponding SDC command ?

What is your worst path ? Is the timing met ?

b/ Input-To-Reg

Let's consider that the inputs in the clock domain CLOCK takes 1.2 nanosecond to come. Even before running the tool can you already know if the timing will be met ? Explain (drawing + maths).

Please write the corresponding SDC command and report the timing. Is the result match your expectation ?

As it is an unrealistic constraint at this frequency, let's consider the minimum input constraint. Open the library .lib file and find the CK-> Q timing (access time). Takes this value as input delay. Rerun the tool and report the new timing. What does it mean ? Is the timing met ? What is the timing slack ?

c/ Reg-To-Output

First, let's consider an output delay on all outputs, synchronized to CLOCK. The output delay is 1.2 ns. Please write the correct SDC command and report the timing.

Is the timing met ? What is the timing slack ?

What is the worst path ? Why is it so long ?

Is the timing met now ?

9/ Report timing

Thanks to the report timing command you can check every single path you need to examine. The report timing command supports several option to specify a path: -from / -to / -through .

A/ Through

Print the timing path through the output Q of a flip-flop of your choice. Draw the path and explain it.

B/ From

Select a start point in the design and print the timing path starting from this point.

C/ To

Select a end point in the design and print the timing path ending at this point.

D/ From-To

Select a start point and a end point as well, print the timing path between these two points.

10/ Optimizations

By default the tool will optimize only the worst negative slack (WNS). WNS optimization produces better area but more violations. You can tell the synthesizer to optimize the total negative slack

(TNS). The TNS approach produces fewer violations, meaning fewer issues in the place-and-route stage, but on the other hand runtime will increase. As a general rule, always turn on TNS optimization.

> set_attribute endpoint_slack_opto true /

Explain the difference between WNS and TNS.

11/ Design for Test: DfT

The insertion of the scan-chains are done during the synthesis. Explain what is a scan-chain and the purpose of it ?

Insert 4 scan-chains using functionnal pins for scan-ins and scan-outs.

For timing constraints, are you using the same sdc file as the functional mode ? Why not ?

Write the corresponding .sdc for the test mode as well as update the .sdc for the functional mode.

12/ Writing outputs

Write out the generated netlist and the sdc:

```
> write_hdl
> write_sdc
```

Open the generated netlist and check if everything seeems ok. Open up the generated sdc and compare it to the input one. What are the differences ? What is a “set_dont_use” ? Why the tool should not use some cells ?

13/ Equivalence checking

RTLC is able to generate the script to automatically run LEC (Conformal)

- what is an equivalence checking tool ?

- why do we need to run such a tool ?

- why do not use simulation ?

Run LEC and check that your netlist is equivalent to the original RTL.