UE-VLSI2 TP : Cadence RTL Compiler
Kévin Mambu, Nicolas Phan

Kevin Mambu

December 23, 2018

1 Set the environment

```
#!/bin/sh
export LM_LICENSE_FILE=30000@licences.cnfm.fr
export PATH=$PATH:/users/soft/opus/Linux/RC-13.12-000/bin
```

2 Load the libraries

n.b: all the libraries are located at `/users/enseig/tuna/ue-vlsi2/techno`.

Two types of cell libraries are available: `_Best.lib` and `_Worst.lib`. The difference between these libraries lies in the timing arcs of their cells : while the Best Library contains Best-Case timings, the Worst Library contains the Worst-Case equivalents. Each timing arcs are estimated via different PVT parameters (**Process - Voltage - Temperature**). These parameters an be found in the headers of each library.

```
/*****************************************************
 Synopsys Technology File

 genstf version 6.4
...
Process values given:
  Library nominal:              1.2
  Tech-file best case:          1.2
  Tech-file centre:             1.2
  Tech-file worst case:         1.2
Voltage values given:
  Library nominal:              1.08
  Tech-file best case:          1.20
  Tech-file centre:             1.08
  Tech-file worst case:         1.0
Temperature values given:
  Library nominal:              85
  Tech-file best case:          25
  Tech-file centre:             125
  Tech-file worst case:         125

  *****************************************************/
```

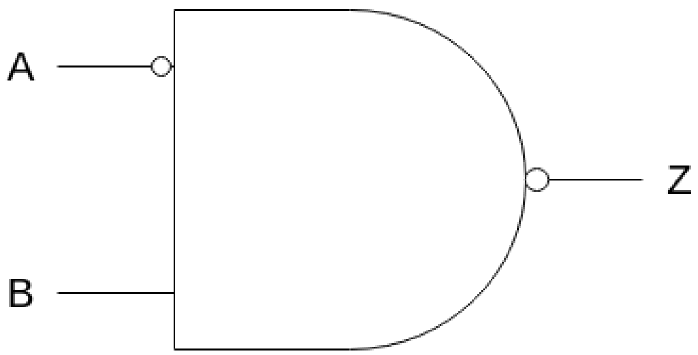
PVT specifications for cmos_120nm_core_Worst

```
/*****************************************************
 Synopsys Technology File

 genstf version 6.4
...
Process values given:
  Library nominal:                0.8
  Tech-file best case:            0.8
  Tech-file centre:               0.8
  Tech-file worst case:           0.8
Voltage values given:
  Library nominal:                1.32
  Tech-file best case:            1.32
  Tech-file centre:               1.32
  Tech-file worst case:           1.20
Temperature values given:
  Library nominal:                0
  Tech-file best case:            -40.0
  Tech-file centre:               0
  Tech-file worst case:           25.0

 *****************************************************/
```

PVT specifications for cmos_120nm_core_Best

In order to synthesize the RTL description, we will use cmos_120nm_core_Worst.lib. This library contains all the generic cells we will need, and with Worst-Case timings in order to thoroughly stress the longest combinational paths.

A practical example of difference in timing arcs between this library and its _Best equivalent can be seen when evaluating the ND2AHS cell, a "2 Input NAND w A Input Inverted and 1x Drive".



Gate-level schematic of the ND2AHS cell

3 Load the design

4 Elaborate

The eaboration process does not differ from the Synthesis per se. The elaboration step is the conversion of the RTL description to a model where every instance object (signal, process, arrays, etc) is converted into a component (adder, multiplexer, register, etc). This step is part of the Synthesis process.

5 Check design

6 Synthesis

7 Reset

There are two reset signals in the design :

1. RESET_N is the external reset signal. Its whereabouts are not bound by any timing constraints and its value can change at any time, as the signal comes from outside the design through an interface pad. This is an asynchronous reset signal.

2. RESET_RX is the internal reset of the design. This one is synchronous as its value is refreshed at every rising edge of the clock.

First we should precise that while the activation of the reset ($0 \rightarrow 1$) is asynchronous, its deactivation ($1 \rightarrow 0$) has to be synchronized with the clock. The design itself is synchronous and the transition from the reset state to the initialization state is synchronous as well. This is the utility of the RESET_RX register.

Still the external reset is a potential hazard for the design, as it comes from the outside. We cannot simply assume that the propagation of the signal was flawless and must consider that it may have had issues on its way:

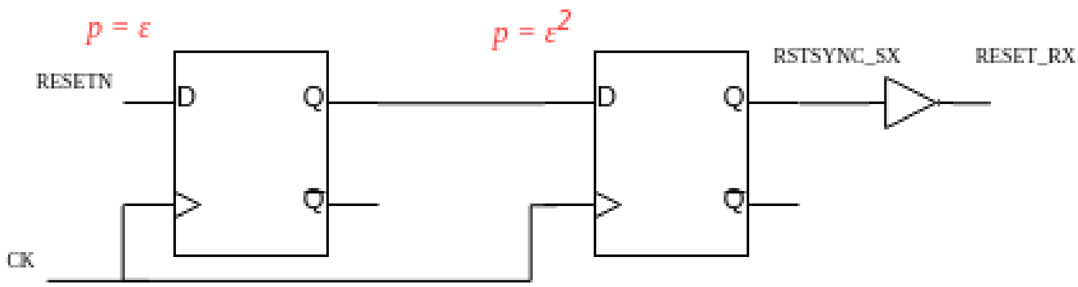
1. The external reset may come from a different clock domain. In that case, the signal would have been synchronous in its previous clock domain but would appear asynchronous when entering the design.

2. The external reset may be already asynchronous.

In any case, the external reset signal may arrive in the design without respecting constraints relative to the setup time of its imminent receptor (a flip-flop, for example). All these possibilities may bring the value of the signal to a metastable state, where its value is indetermined before setting itself to a non-deterministic, but stable value. The probability of such an event is rare but not impossible, and the propagation of such a signal may bring the design to a non-deterministic state, ruining the function of the design.

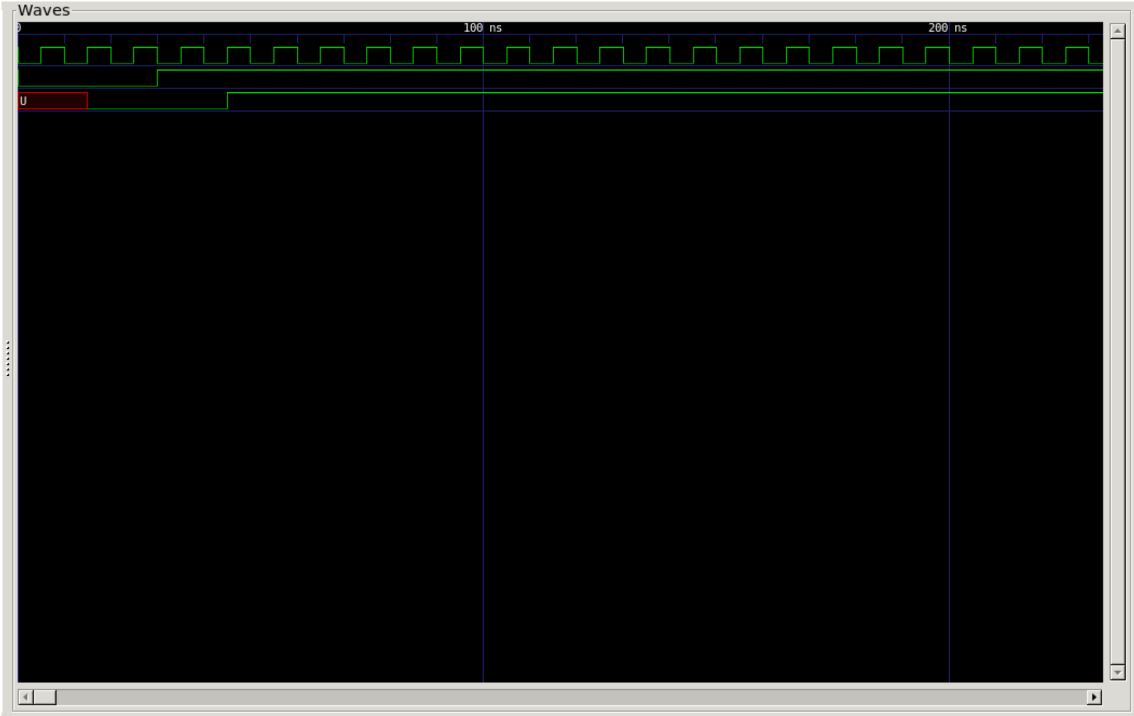
A Reset Synchronizer is a component with an asynchronous signal as input and an asynchronous signal as output. In order to prevent the propagation of a metastable value, the reset synchronizer uses the probabilistic properties of meta-stability : a meta-stable state is always short-lived, and while we cannot predict the end of a meta-stable state, we know for sure that it won't perdure long before setting itself for good.

Let ϵ be the probability for an asynchronous reset to bring a flip-flop to a meta-stable state. For n being the number of flip-flops on a given datapath, the probability of its output being still metastable is equals to ϵ^n. This property makes so that a Reset Synchronizer slims down the probability of the propagation of meta-stability to virtually none.



Schematic of a 2-register Reset Synchronizer. In red are the corresponding probabilities of a propagation of meta-stability.

This model was implemented to the design. The verification of the functionnality was done using GHDL.



Waveform of the Reset Synchronizer using GTKWave

Listing 1: RTL description of the Reset Synchronizer

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity reset_synchronizer is
        port ( i   : in  std_logic;
               clk : in  std_logic;
               q   : out std_logic );
end reset_synchronizer;

architecture behav of reset_synchronizer is
        signal reg0 : std_logic;
        signal reg1 : std_logic;
begin
        q    <= reg1;
        clocked : process(clk)
        begin
                if rising_edge(clk) then
                        reg0 <= i    ;
                        reg1 <= reg0;
                else
                        reg0 <= reg0;
                        reg1 <= reg1;
                end if;
        end process clocked;
end behav;
```

8 Reporting

From the area report, the design is composed of 13878 cells, and has a surface of 220290 square cells.

The timing report gives the longest datapath of our design. We can see that its start point is the I_RI register and its end point the NEXTPC_RD register. The total delay of this datapath is 11.538ns. The maximum frequency, evaluated only from this timing report, equals $\frac{1}{11538 \times 10^{-15}}$ Hz, or approximately 86.670 GHz. The timing slack of a connection is the difference between its required time and its arrival time. In our design the timing slack is UNCONSTRAINED because no timing constraints have been specified yet. Because of this omission, no clock signal are emitted on the design.

When anayzing the lint timing report, we can see the following warning catergories :

1. Generated clocks without clock waveform
2. Inputs without clocked external delays
3. Outputs without clocked external delays
4. Inputs without external driver/transition
5. Outputs without external load

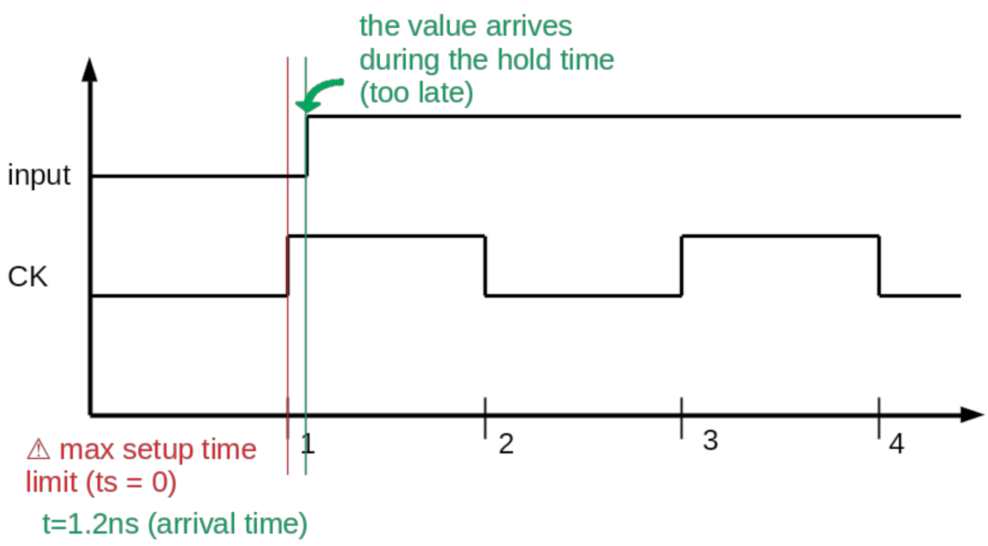
9 Constraints

9.1 Reg-To-Reg

The clock constraint removed the "Generated clocks without clock waveform" issue.

9.2 Input-To-Reg

Let us consider the inputs for the CLOCK domain takes 1.2 nanoseconds to come. Since the clock period in the CLOCK domain is 2 ns (from the addition to our .sdc file in the previous question), it means the arrival time if its signal is 1 ns, for the falling as well as the rising edge. This means that our inputs will always be late to the setup time limit by 0.2 nanoseconds and no inputs will actually be saved in our registers.



No sequential element in our design will manage to actully retain transmitted signals.