University Pierre et Marie Curie

project MOCCA

Report MOCCA : Synthesys, place and route of a given MIPS architecture with CADENCE toolchain

Authors:
Youcef Sekouri, William Fabre

Teachers :
Mr Matthieu Tuna,Pirouz Bazarghan-Sabet

Year 2019-2020

Contents

Project script RTL compiler

This is our script, it has been inspired by this source[1]

```
#RTL
set rtl [list /users/enseig/fabre/work/MOCCA/ProjetMOCCA/src/mips_with_reset.vhd]
#OPTIMISATION
set_attribute endpoint_slack_opto true /
#NAME OF DESIGN
set DESIGN MIPS_32_1P_MUL_DIV
#EFFORT FOR SYNTHESIS
set SYN_EFF high
set MAP_EFF medium
set SYN_PATH "."
#LIBARRY
set_attribute library /users/enseig/tuna/ue_vlsi2/techno/cmos_120/
    cmos_120nm_core_Worst.lib
################################################################################
#READ VHDL
read_hdl -vhdl $rtl
#ELABORATE AND VALIDATE SYNTAX
elaborate $DESIGN
#Reports the time and memory used in the elaboration.
puts "Runtime & Memory after 'read_hdl'"
timestat Elaboration
#CHECK UNRESOLVED IT'S THE WORST THAT COULD HAPPEN
check_design -unresolved
#CONSTRAINT FILE
read_sdc /users/enseig/fabre/work/MOCCA/ProjetMOCCA/src/mips.sdc
################################################################################
#Synthesizing to generic cell
synthesize -to_generic -eff $SYN_EFF
puts "Runtime & Memory after 'synthesize -to_generic'"
timestat GENERIC
#Synthesizing to gates from the used PDK
synthesize -to_mapped -eff $MAP_EFF -no_incr
puts "Runtime & Memory after 'synthesize -to_map -no_incr'"
timestat MAPPED
#Incremental Synthesis
synthesize -to_mapped -eff $MAP_EFF -incr
#Insert Tie Hi and Tie low cells
insert_tiehilo_cells
puts "Runtime & Memory after incremental synthesis"
timestat INCREMENTAL
################################################################################
#write output files and generate reports
report area > ./out/${DESIGN}_area.rpt
report gates > ./out/${DESIGN}_gates.rpt
report timing > ./out/${DESIGN}_timing.rpt
report power > ./out/${DESIGN}_power.rpt
#Verilog
write_hdl -mapped > ./out/${DESIGN}_map.v
#generate the constaraints file> to be used in Encounter
write_sdc > ./out/${DESIGN}_map.sdc
puts "Final Runtime & Memory."
timestat FINAL
puts "====================="
puts "Synthesis Finished :)"
puts "====================="
```

[1] https://sudip.ece.ubc.ca/rtl-compiler/

Load the libraries

Definition Liberty Timing File (LIB) : The .lib file is an `ASCII` format file that represents timing and power parameters associated with cells in a particular semiconductor technology. They are gathered by simulating the cells under various conditions. It can have multiple information[1] :

- operating conditions such as temperature and voltage.
- fanout, capacitance, slew and all necessary thresholds.
- Lookup table for timing arcs, those are functions of output load capacitance and input slew.
- Area of the cell.

Explain the difference between these two different types.

- The Library MAX (worst case) is used to verify the stability of the signal before the edge, also called setup time violation.
- The Library MIN (best case) is used to verify the stability of the signal after the edge, also called hold time violation.

Which one is needed for the synthesis step ? Explain your choice.

During synthesis stage we only have a bunch of cells without any routing delays between them. However, we still have for each cells, timing aspects:

- Rise Delay (Cell Rise): Propagation delay between output and input when output changes from 0 to 1.
- Fall Delay (Cell Fall): Propagation delay between output and input when output changes from 1 to 0.
- The threshold points from which 0 goes to 1 or 1 goes to 0 for the input and for the output.
- Rise Slew (Rise Transition):
  - lower slew threshold: Threshold from which it is considered to go to 1.
  - upper slew threshold: Threshold from which it is considered to be 1.
- Fall Slew (Fall Transition):
  - upper slew threshold: Threshold from which it is considered to go to 0.
  - lower slew threshold: Threshold from which it is considered to be 0.

Therefore, we can use the WC library during the synthesis phase to check if we meet the setup requirements. The sum of delays of a long path of gates can create a setup time violation. The hold time violation is a shortest path problem on the clock tree, here we don't have any clock tree. Furthermore, we can use statistical models (Wire Load Model) which will allow us to do a more coherent timing analysis during synthesis.

[1] https://www.csee.umbc.edu/ tinoosh/cmpe641/, Liberty Timing File courses

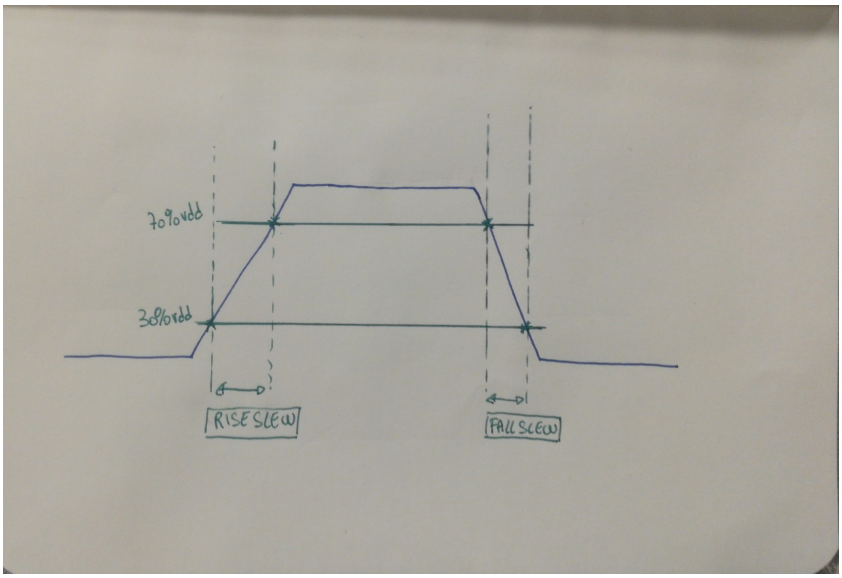

Figure 2.1: Representation of an imaginary gate slews if the lower and upper threshold are respectively 30% and 70% for the fall and the rise

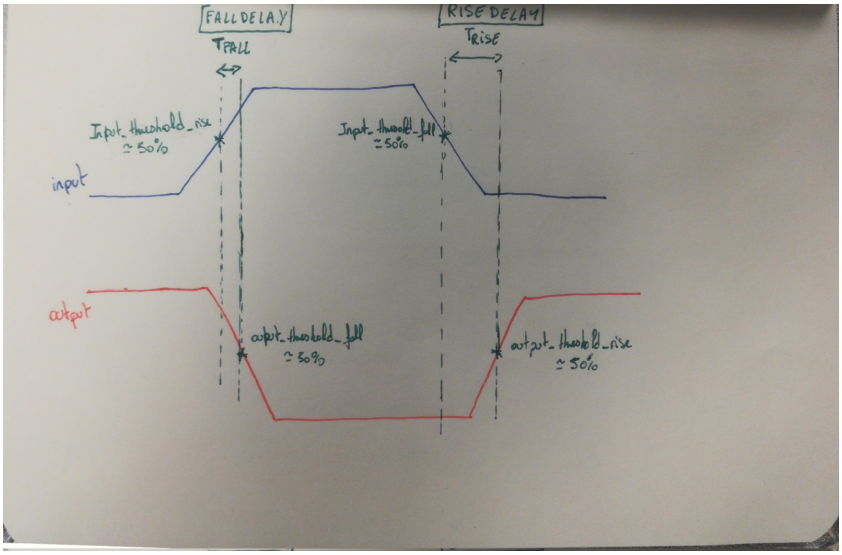

Figure 2.2: Representation of the Fall Delay and the Rise Delay of an imaginary inverter gate, in blue the input, in red the output.

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.

Here are some interesting differences between WC lib and BC lib to know before we watch the graphics. Values for Temperature and Voltage:

	WC lib	BC lib
nominal Voltage	1,08	1,32
nominal Temperature	0	125

Figure 2.3: Table of nominal Voltage and Temperature for WC and BC lib.

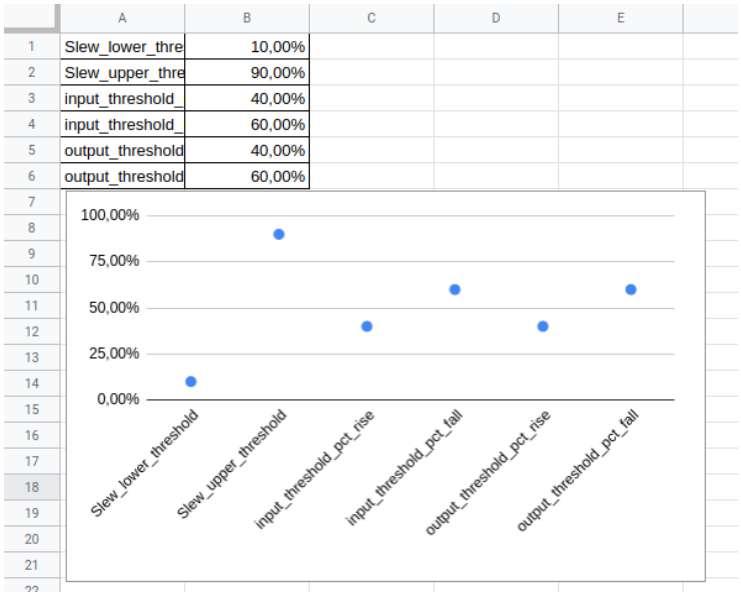

Figure 2.4: Representation of all the necessary thresholds to characterize the cell for both WC and BC (same results), slew, input, output.

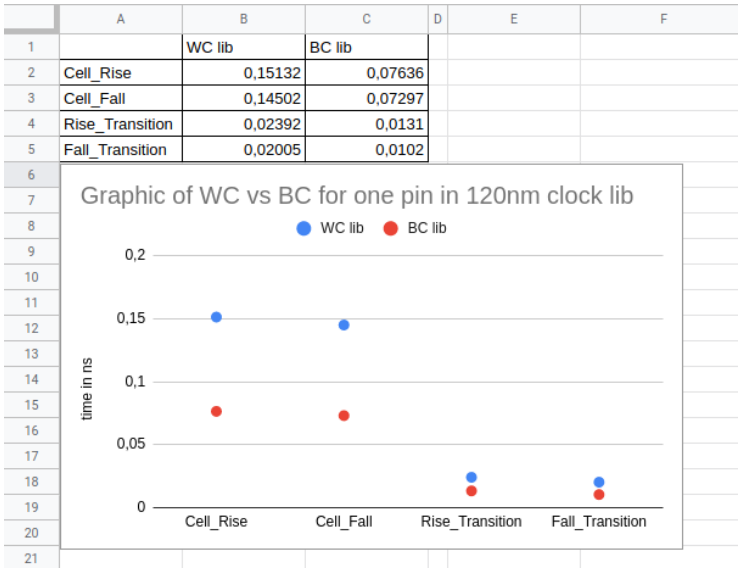

Figure 2.5: Representation of the Fall Delay, the Rise Delay, Rise Slew, Fall Slew, we can clearly see a difference between the WC and BC library.

Elaborate

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

From this link there is a clear definition of elaboration a design[2].

Synthesis might be seen as one big step for a user perspective. It consists of :

- Elaborating the design.
- Apply constraints to the design.
- High level optimizations of the design.
- technology mapping (mapping with the library).
- Low level optimizations of the design.

The Elaborate step consists is reading RTL file to recognize hardware structures and transform them into future placeholders for the technology mapping.

It seems that for some tools, the elaborate step also do some syntax checks and verify the connections between structure. For example if two outputs are connected etc.

This step is crucial because you cannot apply constraints on an RTL file, so those placeholders will be the premises of our netlist.

```
1       rc:/> elab -h
2       elaborate: elaborates previously read HDL files and creates corresponding
    design and subdesigns.
3
```

Figure 2.6: Help option from RC interpreter.

[2] https://forums.xilinx.com/t5/Synthesis/what-exactly-is-elaborating-a-design/td-p/682043

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 ?

Definition of check_design command : "Provides information on undriven and multi-driven ports and pins, unloaded sequential elements and ports, unresolved references, constants connected ports and pins, any assign statements and preserved instances in the given design. In addition, the command can report any cells that are not in both .lib and the physical libraries (LEF files).By default, if you do not specify an option, the check_design command reports a summary table with this information."[3]

An unresolved Reference is when during the elaborate step, we go from vhdl to design, "undefined modules and VHDL entities are labeled "unresolved" and treated as blackboxes"[4].

```
rc:/> check_design -unresolved
Checking the design.

Unresolved References & Empty Modules
-------------------------------------
No unresolved references in design 'MIPS_32_1P_MUL_DIV'

No empty modules in design 'MIPS_32_1P_MUL_DIV'

Done Checking the design.

```

Figure 2.7: Example of check_design usage on our design.

A missing model error seems to be when there is no IP it's a report that the cells are not in .lib nor in the physical libraries (LEF files).

[3] https://www.csee.umbc.edu/~tinoosh/cmpe641/tutorials/rc/rc_commandref.pdf page 316
[4] https://www.csee.umbc.edu/~tinoosh/cmpe641/tutorials/rc/rc_commandref.pdf, p284

Reset synchronizer

What kind of reset is used in this design ? Synchronous or asynchronous ?

As we can see in the design the reset is asynchronous. Indeed, it is just a signal that comes from outside of the rtl design.

```
port
  (
  signal   CK           : in    std_logic                        ;-- external clock
  signal   RESET_N      : in    std_logic                        ;-- external reset

  signal   IT_N         : in    std_logic_vector (  5 downto  0);-- hw interrupts

  signal   CPU_NBR      : in    std_logic_vector (  9 downto  0);

  signal   I_A          : out   std_logic_vector ( 31 downto  2);-- adr inst
  signal   I_RQ         : out   std_logic                        ;-- inst rqst
  signal   MODE         : out   std_logic_vector (  1 downto  0);-- mode
  signal   I_RBERR      : in    std_logic                        ;-- inst bus error
  signal   I_ACCPT      : in    std_logic                        ;-- rqst acpt
  signal   I_IN         : in    std_logic_vector ( 31 downto  0);-- inst
  signal   I_ACK        : out   std_logic                        ;-- rqst ack
  signal   I_BEREN      : out   std_logic                        ;-- err  enable
  signal   I_INLINE     : out   std_logic                        ;-- in cache line

  signal   D_A          : out   std_logic_vector ( 31 downto  2);-- adr
  signal   D_BYTSEL     : out   std_logic_vector (  3 downto  0);-- byt select
  signal   D_RQ         : out   std_logic                        ;-- data request
  signal   D_RW         : out   std_logic                        ;-- r-w
  signal   D_SYNC       : out   std_logic                        ;-- synchro
  signal   D_REG        : out   std_logic                        ;-- ext reg
  signal   D_LINKED     : out   std_logic                        ;-- linked acs
  signal   D_RSTLKD     : out   std_logic                        ;-- reset lnkd
  signal   D_CACHE      : out   std_logic                        ;-- cache oper
  signal   D_CACHOP     : out   std_logic_vector (  4 downto  0);-- cache oper
  signal   D_RBERR      : in    std_logic                        ;-- data bus error
  signal   D_WBERR      : in    std_logic                        ;-- data bus error
  signal   D_ACCPT      : in    std_logic                        ;-- rqst acpt
  signal   D_OUT        : out   std_logic_vector ( 31 downto  0);-- data
  signal   D_IN         : in    std_logic_vector ( 31 downto  0);-- data
  signal   D_ACK        : out   std_logic                        ;-- rqst ack

  signal   MCHECK_N     : in    std_logic                        ;-- machine chk


  signal   SCOUT        : out   std_logic                        -- scan out

);
end MIPS_32_1P_MUL_DIV;
```

Figure 2.8: RTL design interface, as we can see the second signal is the external reset.

What is a reset synchronizer, please explain why a circuit should have a reset synchronizer ? Please explain the differences between both? Add TO your report the figure and the waveforms of your reset synchronizer.

It is use to synchronize an external reset with the clock domain of IC and do to prevent hold/set up violation. Indeed, the release of an external reset can happen asynchronously. So, reset must be synchronized to prevent meta-stability problem. There are two types of reset synchronizer :

- Asynchronous : The output signal activation is done asynchronously.
- Synchronous : The output signal activation is done synchronously.

The output signal deactivation, is always done synchronously.

Synchronous Reset synchronizer

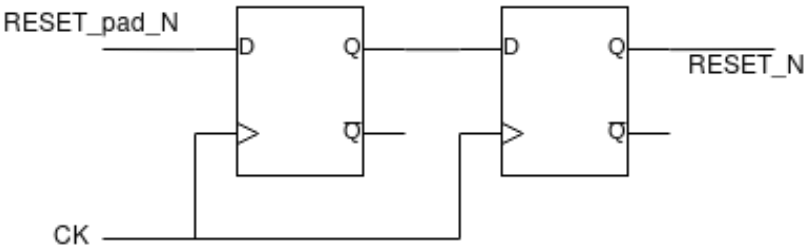

Figure 2.9: A synchronous reset synchronizer

```
synchronious_synchronizer : process (CK)
signal flip1, flip2 : std_logic;
begin
    -- RESET_N actif 0
    -- activation synchrone
    -- desactivation synchrone
    if rising_edge(CK) then
      flip1 <= RESET_pad_N;
      flip2 <= flip1;
  end if;
end process;
```

Figure 2.10: VHDL example of a synchronous reset synchronizer

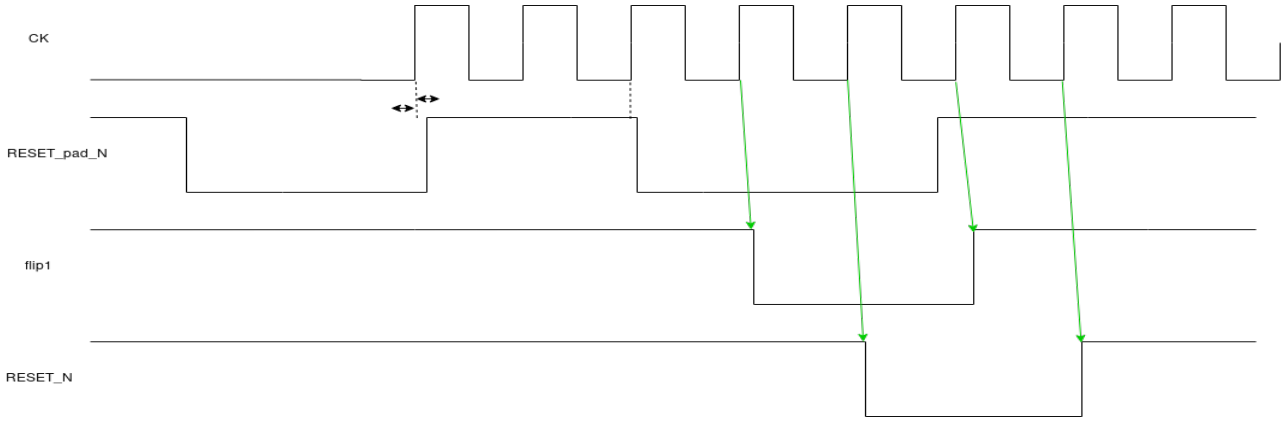

Figure 2.11: Waveform of a synchronous reset synchronizer

Asynchronous Reset synchronizer

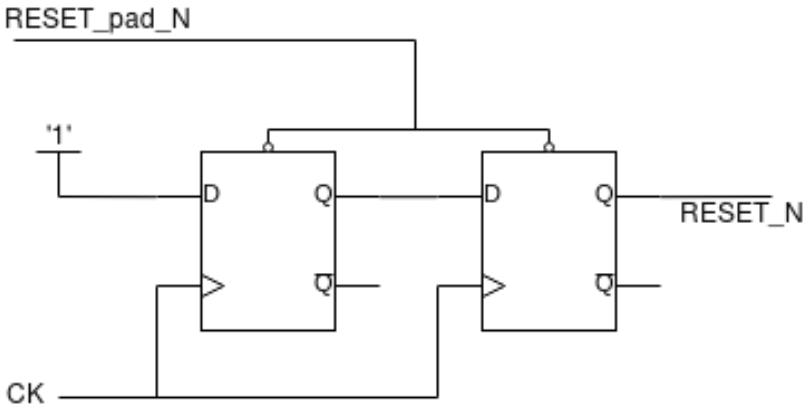

Figure 2.12: An asynchronous reset synchronizer

```vhdl
asynchronious_synchronizer : process (CK, RESET_pad_N)
signal flip1, flip2 : std_logic;
begin
    -- RESET_N actif 0
    -- activation asynchrone
    -- desactivation synchrone
    if(RESET_pad_N = '0') then
        flip1 <= '0';
        flip2 <= '0';
    elsif rising_edge(CK) then
      flip1 <= '1';
      flip2 <= flip1;
  end if;
end process;
```

Figure 2.13: VHDL example of an asynchronous reset synchronizer

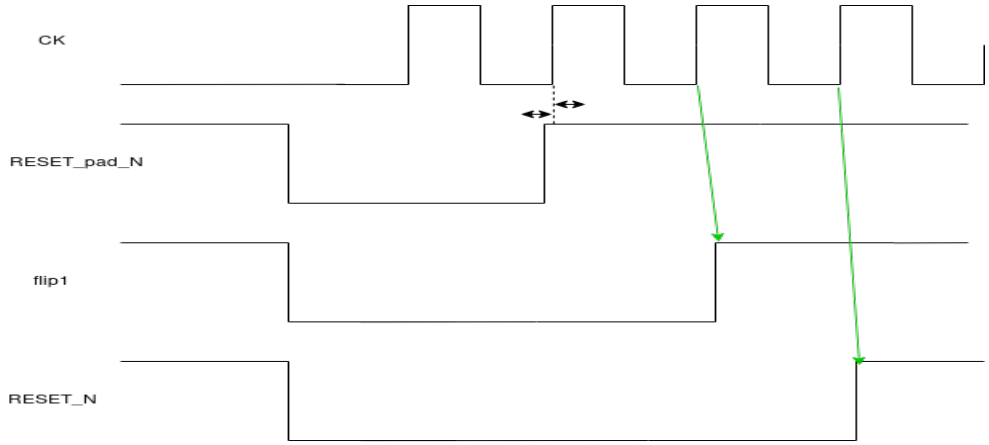

Figure 2.14: Waveform of a asynchronous reset synchronizer

Is there any reset synchronizer ? Code a reset synchronizer for asynchronous reset and add it to the design.

There was no synchronizer in the design at first, here is our implementation :

```
  -- ### ------------------------------------------------------ ###
  -- #          synchronous reset synchronizer          #
  -- ### ------------------------------------------------------ ###
synchronious_synchronizer : process (CK)
begin
    -- RESET_N actif 0
    -- activation synchrone
    -- desactivation synchrone
    if rising_edge(CK) then
      RESET_SYNC1 <= RESET_N;
      RESET_SYNC2 <= RESET_SYNC1;
  end if;
end process;
```

Figure 2.15: VHDL of the reset synchronizer in the RTL design.

Reporting

What is the area of the design ? How many cells are used ?

"Area of the design" definition : It is the sum of all the gates area plus the total net area. During synthesis there is no net area.

The Total Area can be found in the report table below, you can also see the number of cells :

Instance	Cells	Cell Area	Net Area	Total Area	Wireload
MIPS_32_1P_MUL_DIV	14058	221458	0	221458	area_216Kto240K (S)

(S) = wireload was automatically selected

Which type of wireload model is used ? Why ?

The wireload model is specified as : "Wireload mode : enclosed" by the command report area. We can see down the table that this WLM was chosen by default.

What are the other types ? What wireload model are used for each hierarchy ? Why ?

According to the article On the Relevance of Wire Load Models[1] the flow of creation of a chip can be divided into (All the quotations will come from this article in the next paragraph).:

- The front end : from "RTL description and moves through logic synthesis and optimization"
- The back end : "placement, routing, extraction and performance analysis"

Before P&R our design net parasitics and delays are unknown. We currently only know the fanout of the net in the block that the net belongs to. Therefore, in order to estimate the total length and capacitance of the net, WLM keep statistics of previously routed chips inside lookup tables.

There are 3 types of wire-load models[5] :

- Top : All nets in the hierarchy will inherit the same wireload mode as the top-level block.
- Enclosed : All nets will inherit the WLM of the block that completely encloses the net even if it goes through sub-blocks.
- Segmented : If the net goes across several WLM, it will use the corresponding WLM on every portion of the net enclosed by a block.

Graphically[6] :

[5] https://www.edaboard.com/showthread.php?33505-Difference-between-quot-top-quot-amp-quot-enclosed-quot
[6] http://asic-soc.blogspot.com/2013/07/wire-load-models-for-synthesis.html

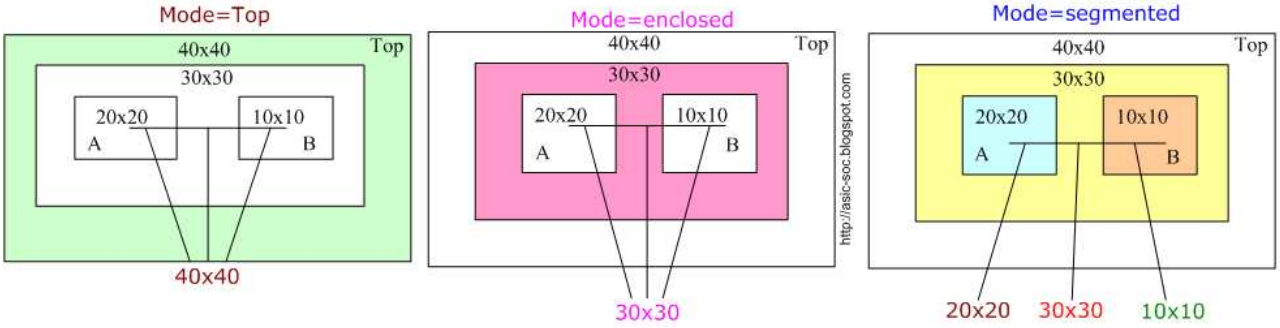

Figure 2.16: Graphic modelization of the wire with different type of WLM, true value is at the bottom. We can see the different blocks colored

`report timing` print the timing on the longest path. Examine the path (start point / end point etc.) and explain it.

- Start-point : I_RI_reg[31]/CP
- End-point : NEXTPC_RD_reg[30]/D

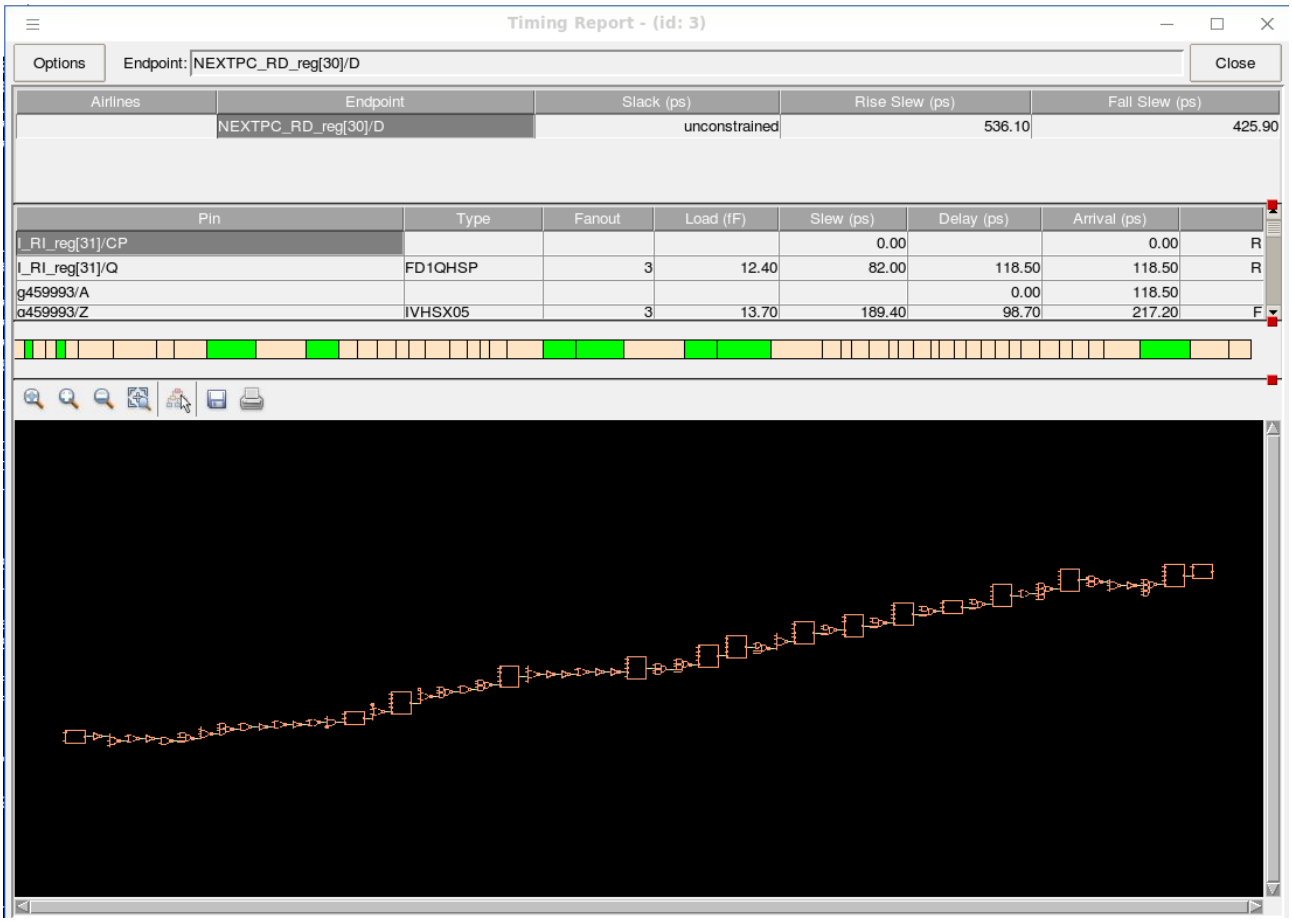

Figure 2.17: It is the longest path because

We can see a report and the path from the selected flip-flop to the farthest flip-flop and all the cells between. The delay for each cell of the path is reported.

What is the timing of the path ? What is the supposed max frequency the design is able to run ?

The timing of the path is visible at the last line :

```
    NEXTPC_RD_reg[30]/CP    setup                    0  +310    13992 R
```

It is the sum of all delay in picoseconds to go through all the logic from the reg at the start-point to the reg at the endpoint. The max frequency will be the inverse of this longest path delay.

You can see at the end of the report the following comment: `Timing slack : UNCONSTRAINED` What is a 'timing slack' ? Why it is 'UNCONSTRAINED' ?

As we can see in this course[7] the `report - timing - worst Path` will allow us to see the slowest path. We can see the value Slack on the graph above. but there is no value because we didn't constraint our design. This value will be positive in picoseconds if we have spare time. This implies the clock frequency can be increased. If it is a negative slack the design is failing. It is currently unconstrained because we didn't put any constraints.

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.

```
    Lint summary
    Unconnected/logic driven clocks                                  0
    Sequential data pins driven by a clock signal                    0
    Sequential clock pins without clock waveform                  2877
    Sequential clock pins with multiple clock waveforms              0
    Generated clocks without clock waveform                          0
    Generated clocks with incompatible options                       0
    Generated clocks with multi-master clock                         0
    Paths constrained with different clocks                          0
    Loop-breaking cells for combinational feedback                   0
    Nets with multiple drivers                                       0
    Timing exceptions with no effect                                 0
    Suspicious multi_cycle exceptions                                0
    Pins/ports with conflicting case constants                       0
    Inputs without clocked external delays                          88
    Outputs without clocked external delays                        116
    Inputs without external driver/transition                       88
    Outputs without external load                                  116
    Exceptions with invalid timing start-/endpoints                  0

    Total:       3285
```

Figure 2.18: `report -lint` output, as we can see there are more than 3 categories of errors.

- Sequential clock pins without clock waveform : pin not driven.
- Inputs without clocked external delays : we need to use set_input_delay.
- Outputs without clocked external delays : we need to use set_output_delay.
- Inputs without external driver/transition : we need to set input ports transition slew rates.

[7] `http://www.siue.edu/~gengel/ece484LabMaterial/LogicSynthesisTutorial.pdf`

- Outputs without external load : The delay is a function between external load capacitance and input gate capacitance. thus it needs to be set.[8]

[8] http://www.cse.psu.edu/~kxc104/class/cmpen411/14f/lec/C411L10InvDynamic.pdf,p12

Constraints

2.6.1 Reg-To-Reg

Add in the RTLC script the command: `read_sdc mips.sdc`

We have three main constraints :

```
create_clock -name clk_500MHz -period 2 [get_port Clk]
set_input_delay 0.5 -clock clk_500MHz [get_port In]
set_output_delay 0.3 -clock clk_500MHz [get_port Out]
```

Figure 2.19: The three main constraints to set

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

The `report timing -lint` is better :

- Errors about sequential clock pins without clock waveform has decreased from 2877 to 0.
- Errors about inputs/outputs without clocked external delays has decreased from 88/116 to 0/0.
- Inputs without external driver/transition has decreased from 88 to 87.

Why does the report show the clock now ?

That is because we applied constraints on :

- input delay
- reg-to-reg delay
- output delay

The remaining error reports are due to input and output which are not connected.

```
I_A[14]                    out port                          +0    2311 F
(mips.sdc_line_14_93_1)    ext delay                       +300    2611 F
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(clock clk_500MHz)         capture                                 2000 R
-------------------------------------------------------------------------
Cost Group   : 'clk_500MHz' (path_group 'clk_500MHz')
Timing slack :    -611ps (TIMING VIOLATION)
Start-point  : I_RI_reg[30]/CP
End-point    : I_A[14]
```

Figure 2.20: Report timing : Timing not met after STA with WLM

What is the worst path ? What is the timing ? Is the slack positive ?

We have a timing violation on our new worst path :

- Start -point : I_RI_reg [30]/ CP9

- End -point : I_A [14]

We can either optimize RTL or relax the clock or both. Nevertheless, if we relax the clock to resolve this worst path, all the timings in the design will be met. Currently, the worst path returns a timing negative slack, which value is -611ps, which is a reg-to-output path. In order to relax the clock perio,d we can take this value into account. At the moment there is no clock skew.

Report the 10 worst paths.

The command is : rc:/>report timing -worst 10

Timing slack (ps)	Start-point	End-point
-611	I_RI_reg[30]/CP	I_A[14]
-611	I_RI_reg[30]/CP	I_A[17]
-611	I_RI_reg[30]/CP	I_A[14]
-611	I_RI_reg[27]/CP	I_A[14]
-611	I_RI_reg[30]/CP	I_A[17]
-611	I_RI_reg[27]/CP	I_A[17]
-611	I_RI_reg[30]/CP	I_A[14]
-611	I_RI_reg[30]/CP	I_A[14]
-611	I_RI_reg[27]/CP	I_A[14]
-610	I_RI_reg[30]/CP	I_A[14]

Figure 2.21: Report of the 10 worst path of the design

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

- launch clock : It is an event, the moment where data is launched by a FF.
- capture clock : It is an event, the moment where data is captured by another FF.

Here a graph representing the launch and capture from berkley university courses[9] :

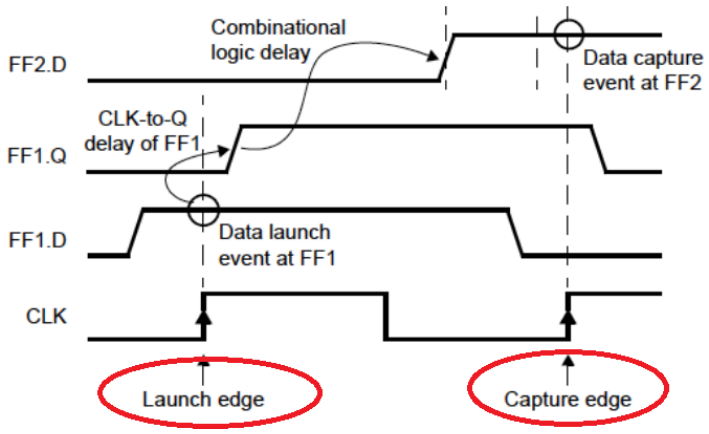

Figure 2.22: Example of Launch and capture clock represented from

[9] https://inst.eecs.berkeley.edu/~ee290c/sp17/lectures/Lecture27.pdf

```
          Pin                Type       Fanout  Load Slew Delay Arrival
                                                (fF) (ps)  (ps)   (ps)
-------------------------------------------------------------------------
(clock clk_500MHz)        launch                                    0 R
...
...
...
(clock clk_500MHz)        capture                                2000 R
```

Figure 2.23: Representation of the report from rtl compiler

Why did the tool choose this timing between the launch clock and the capture clock ?

Because it is the maximum time we have to capture the right data.

What is the problem with your timing path in this case where the capture clock is another clock: CLOCK2 ? 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.

- set_false_path command order to the compiler to ignore a path.
- set_multi_cycle command order to the compiler to look the capture after some cycle of the launch clock.

It depends if the clocks are in a phase relationship. If yes, we use the set_multicycle_path command to fix it. If no, the slack between the launch and the capture clock is variable. So it's a clock domain crossing problem. The design must be ready for it, and we set this path with set_false_path command.

"If two sequential elements are triggered by two different clocks then a common least common multiple (LCM) of these two different clock periods should be considered to find the launch edge and capture edge for setup and hold timing" analysis[10]

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 ?

We use the multi_cycle_path command.

[10] http://www.idc-online.com/technical_references/pdfs/electronic_engineering/Fundamentals_of_Timing.pdf

What is your worst path ? Is the timing met ?

There is only one clock domain, there might be a mistake in the questions.

Still the timing is not met. In order to meet the timing we have to lower the frequency of the clock. We tried this :

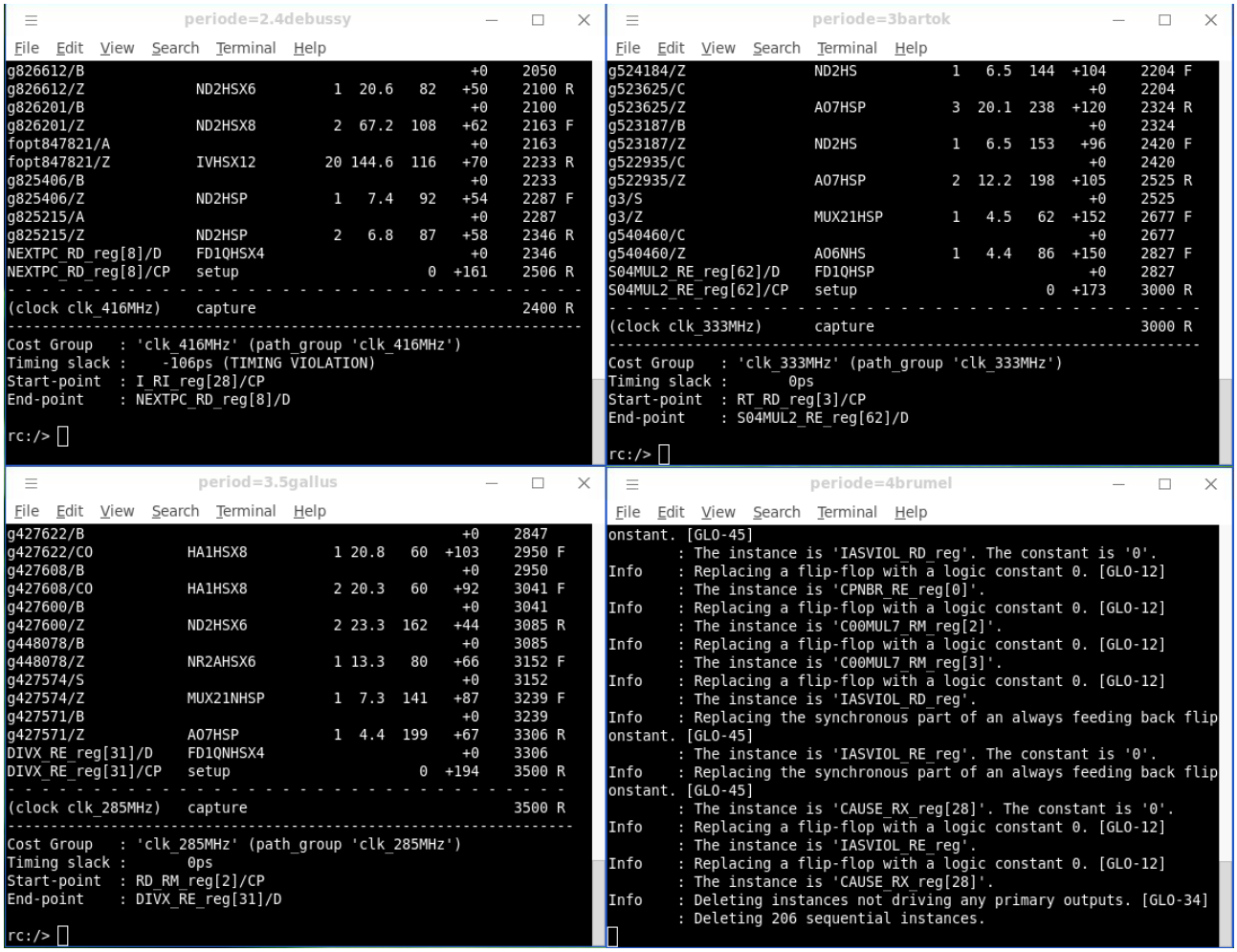

Figure 2.24: Multiple synthesis at the same time period = 3ns wins. Timing slack is now 0.

The problem is when you try report summary there is a fanout problem from DRC rule check :

```
...
Max_fanout design rule (violation total = 3.000)
...

```

We don't have this problem if we set the periode to 3.5ns.

It is overkill to do that because it can be resolved after and we can increase the fanout. Still, we want a robust design and it's our first full synthesis process so let's play safe with those values. We can still keep in mind that : This is the best value to have no timing slack violation.

```
Design Rule Check
-----------------
Max_transition design rule: no violations.

Max_capacitance design rule: no violations.

Max_fanout design rule: no violations.

```

```
Cost Group   : 'clk_392MHz' (path_group 'clk_392MHz')
Timing slack :       0ps
Start-point  : I_RI_reg[28]/CP
End-point    : NEXTPC_RD_reg[6]/D

```

2.6.2 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).

The clock period must be greater than the worst path additionned with the input delay. We have a majoration of Xns for the period in order to have a design that synthetize without any timing negative slack.

$$WPT = 2311ps \tag{2.1}$$
$$WPT\_minimal_majoration = 2550ps \tag{2.2}$$
$$WPT\_maximal_majoration = 3500ps \tag{2.3}$$
$$I/O\_delay = 1200ps \tag{2.4}$$
$$clock\_period > worst\_path\_timing + I/O\_delay \tag{2.5}$$
$$clock\_period > (2311 + 1200) = 3511ps \tag{2.6}$$
$$freq = 1/0.003511 < freq = 1/0.0036 \tag{2.7}$$
$$freq >= 276mhz \tag{2.8}$$
$$\tag{2.9}$$

We tried 2.4ns, 3.5ns and 4.7ns. As we could predict 2.4ns does not work at all but here are the results for 3.5 and 4.7 :

```
Cost Group   : 'clk_285MHz' (path_group 'clk_285MHz')
Timing slack :       0ps
Start-point  : I_RI_reg[31]/CP
End-point    : NEXTPC_RD_reg[26]/D


Cost Group   : 'clk_212MHz' (path_group 'clk_212MHz')
Timing slack :       0ps
Start-point  : RD_RE_reg[0]/CP
End-point    : DIVX_RE_reg[31]/D

```

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.

```
      create_clock -name clk_285MHz -period 3.5 [get_port CK]
      set_input_delay 1.2 -clock clk_285MHz [get_port RESET_N]
      set_input_delay 1.2 -clock clk_285MHz [get_port IT_N]
      set_input_delay 1.2 -clock clk_285MHz [get_port CPU_NBR]
      set_input_delay 1.2 -clock clk_285MHz [get_port I_RBERR]
      set_input_delay 1.2 -clock clk_285MHz [get_port I_ACCPT]
      set_input_delay 1.2 -clock clk_285MHz [get_port I_IN]
      set_input_delay 1.2 -clock clk_285MHz [get_port D_RBERR]
      set_input_delay 1.2 -clock clk_285MHz [get_port D_WBERR]
      set_input_delay 1.2 -clock clk_285MHz [get_port D_ACCPT]
      set_input_delay 1.2 -clock clk_285MHz [get_port D_IN]
      set_input_delay 1.2 -clock clk_285MHz [get_port MCHECK_N]

```

Figure 2.25: input delay commands.

Open the library .lib file and find the CK->Q timing (access time). Takes this value as input delay.

The flip-flop used here is the "FD1QHSP" cell in library. The the minimal edge timing of the output is equal to the best timing of a transition, it is 0.05438ns. The value of CK->Q is 0.05438ns.

Rerun the tool and report the new timing. What does it mean ? Is the timing met ? What is the timing slack ?

Here are the results for 392mhz and 285mhz clock :

```
      Cost Group   : 'clk_392MHz' (path_group 'clk_392MHz')
      Timing slack :       0ps
      Start-point  : I_RI_reg[29]/CP
      End-point    : NEXTPC_RD_reg[16]/D

      Cost Group   : 'clk_285MHz' (path_group 'clk_285MHz')
      Timing slack :       0ps
      Start-point  : RD_RM_reg[2]/CP
      End-point    : DIVX_RE_reg[31]/D

```

2.6.3 Reg-To-Output

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

```
create_clock -name clk_285MHz -period 2.55 [get_port CK]
set_output_delay 1.2 -clock clk_392MHz [get_port I_A]
set_output_delay 1.2 -clock clk_392MHz [get_port I_RQ]
set_output_delay 1.2 -clock clk_392MHz [get_port MODE]
set_output_delay 1.2 -clock clk_392MHz [get_port I_ACK]
set_output_delay 1.2 -clock clk_392MHz [get_port I_BEREN]
set_output_delay 1.2 -clock clk_392MHz [get_port I_INLINE]
set_output_delay 1.2 -clock clk_392MHz [get_port D_A]
set_output_delay 1.2 -clock clk_392MHz [get_port D_BYTSEL]
set_output_delay 1.2 -clock clk_392MHz [get_port D_RQ]
set_output_delay 1.2 -clock clk_392MHz [get_port D_RW]
set_output_delay 1.2 -clock clk_392MHz [get_port D_SYNC]
set_output_delay 1.2 -clock clk_392MHz [get_port D_REG]
set_output_delay 1.2 -clock clk_392MHz [get_port D_LINKED]
set_output_delay 1.2 -clock clk_392MHz [get_port D_RSTLKD]
set_output_delay 1.2 -clock clk_392MHz [get_port D_CACHE]
set_output_delay 1.2 -clock clk_392MHz [get_port D_CACHOP]
set_output_delay 1.2 -clock clk_392MHz [get_port D_OUT]
set_output_delay 1.2 -clock clk_392MHz [get_port D_ACK]
set_output_delay 1.2 -clock clk_392MHz [get_port SCOUT]

```

Figure 2.26: output delay commands

Is the timing met ? What is the timing slack ?

Here are the results for 285mhz and 212mhz clock :

```
Cost Group   : 'clk_285MHz' (path_group 'clk_285MHz')
Timing slack :     -85ps (TIMING VIOLATION)
Start-point  : I_RI_reg[28]/CP
End-point    : I_A[28]


Cost Group   : 'clk_212MHz' (path_group 'clk_212MHz')
Timing slack :       1ps
Start-point  : RD_RE_reg[3]/CP
End-point    : DIVX_RE_reg[30]/D

```

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

Is the timing met now ?

Timing is met for values : 392mhz and 0.05438ns of output delay (it was only for test). We started a synthesis with high effort : `synthesize -effort high -to_mapped`

```
      Cost Group   : 'clk_392MHz' (path_group 'clk_392MHz')
      Timing slack :       0ps
      Start-point  : I_RI_reg[28]/CP
      End-point    : I_TYPE_RD_reg[7]/D

```

Report timing

2.7.1 Through

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

We selected this flip-flip from netlist :

- Instance : I_TYPE_RM_reg[11]
- libcell: FD1QHSP

And we run the next command : report timing -through I_TYPE_RM_reg[11]

Return :

– Timing slack : 6ps
– Start-point : I_TYPE_RM_reg[11]/CP
– End-point : DIVY _RE_reg[26]/D

The instance is a flip-flop, this command in this case, is equal to a "report timing -from". So we use the same command on a cell of this path which is not a flip-flop to explain report-through result: report timing -through g1300010
And we get the same result :
– Timing slack : 6ps
– Start-point : I_TYPE_RM_reg[11]/CP
– End-point : DIVY _RE_reg[26]/D

In order to see this netlist path, we report the timing from start-point on gui :

2.7.2 From

Command : report timing -from I_TYPE_RM_reg[11]

Results :
– Timing slack : 6ps
– Start-point : I_TYPE_RM_reg[11]/CP
– End-point : DIVY_RE_reg[26]/D

2.7.3 To

Command : report timing -to I_TYPE_R_reg[11]

Airlines	Endpoint	Slack (ps)	Rise Slew (ps)	Fall Slew (ps)
	DIVY_RE_reg[26]/D	5.70	206.60	157.50

Pin	Type	Fanout	Load (fF)	Slew (ps)	Delay (ps)	Arrival (ps)	
(clock clk_416MHz)	launch					0.00	R
I_TYPE_RM_reg[11]/CP				0.00		0.00	R
I_TYPE_RM_reg[11]/Q	FD1QHSP	5	34.90	156.00	149.50	149.50	R
g1287309/C					0.00	149.50	
g1287309/Z	AO2HSP	1	6.30	146.60	109.40	258.90	F
g1300010/D					0.00	258.90	
g1300010/Z	AO17HSP	2	12.80	191.70	94.40	353.30	R
g1286269/A					0.00	353.30	
g1286269/Z	IVHSP	2	18.10	98.50	79.70	433.00	F
g1284869/A					0.10	433.10	
g1284869/Z	ND2HSX4	2	11.10	89.00	56.60	489.70	R
g1299661/A					0.00	489.70	
g1299661/Z	ND2AHSP	2	10.30	112.40	114.30	604.00	R
g1299465/C					0.00	604.00	
g1299465/Z	AO7NHSP	4	28.10	141.20	170.60	774.60	R
g207/B					0.00	774.60	
g207/Z	ND2HS	1	7.30	123.20	78.40	853.00	F
g1300132/B					0.00	853.00	
g1300132/Z	ND3ABHSX4	5	36.50	116.10	145.00	998.00	F
g1281553/B					0.00	998.00	
g1281553/Z	NR2HSX4	7	33.20	182.60	83.70	1081.70	R
g1281333/B					0.00	1081.70	
g1281333/Z	AN2HSP	8	42.70	182.40	168.40	1250.10	R
g1280910/A					0.00	1250.10	
g1280910/Z	NR2AHSP	11	55.00	454.90	245.10	1495.20	R
g1278761/C					0.00	1495.20	
g1278761/Z	AO11HS	1	4.60	349.10	269.70	1764.90	F
g1278203/C					0.00	1764.90	
g1278203/Z	ND3HS	1	4.80	185.90	148.20	1913.10	R
g1273179/B					0.00	1913.10	
g1273179/Z	NR4HS	1	4.60	259.30	153.70	2066.80	F
g1268899/C					0.00	2066.80	
g1268899/Z	AO7HS	1	4.50	206.60	132.80	2199.60	R
DIVY_RE_reg[26]/D	FD1QHS				0.00	2199.60	
DIVY_RE_reg[26]/CP	setup			0.00	194.70	2394.30	R
(clock clk_416MHz)	capture					2400.00	R

Figure 2.27: Resume of a reg-to-reg path with all cells delays.

Results :
– Timing slack : 12ps
– Start-point : DRQ_RE_reg/CP
– End-point : I_TYPE_RM_reg[11]/D

2.7.4 From-To

Command : report timing -from I_TYPE_RM_reg[11] -to DIVY_RE_reg[26]/D

Results :
– Timing slack : 6ps
– Start-point : I_TYPE_RM_reg[11]/CP
– End-point : DIVY_RE_reg[26]/D

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`

result :

```
      WITHOUT HIGH EFFORT
      Cost Group   : 'clk_392MHz' (path_group 'clk_392MHz')
      Timing slack :     -18ps (TIMING VIOLATION)
      Start-point  : I_RI_reg[26]/CP
      End-point    : NEXTPC_RD_reg[30]/D

      WITH HIGH EFFORT
      Cost Group   : 'clk_392MHz' (path_group 'clk_392MHz')
      Timing slack :       0ps
      Start-point  : I_RI_reg[30]/CP
      End-point    : I_TYPE_RD_reg[23]/D

```

Explain the difference between WNS and TNS.

That are slack for setup timing :

- WNS = Worst Negative Slack = the worst of all negative slack
- TNS = Total Negative Slack = sum of the negative slack paths

The TNS with WNS, lets appreciate the number of negative slack path.

NB : For hold timing violation after clock-tree, we look the WHS(=WNS) and the THS(=TNS).

Design for Test : DFT

After fabrication of IC, many defects can appear. There can be numerous failures and, found those is like looking for a needle in a haystack. Then, we use the DFT to do it. DFT boils down to an abstraction of a set defects to a faulty model. An example is a short-circuit between a gate and power supply, results in a signal "stuck" to a static state. So, detection of a specific abstract fault reveals a defect into a specific set of defects. This abstraction of defect makes their detection easier.

In summary, the DFT allows to detect fabrication errors. There are many models, like static stuck-at-fault model like above, or the same in a dynamic model (dynamic is a superset of static) to detect bad connection.

In fact, that resume to add "shadow" logic into the design. The DFT methods must be integrated in design conception step.

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

A scan-test is the use of a sequential circuit. It resumed the ATPG to a combinational test problem, which is easier to solve. The scan-test making all flip-flop scannable, by chain them. That allows to put a value in each flip-flop at a time, and to get them at another time. To compare an expected value with a real value to detect a fault. And allow to automate the test by using test-pattern.

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

```
read_sdc mips.sdc
set_attribute dft_scan_style muxed_scan /
define_dft shift_enable -name SHIFT_EN -active high scan_en
define_dft test_mode -name TEST_MODE -active high test_mode
define_dft test_clock -name TEST_CLOCK scan_clk
check_dft_rules
report dft_setup
synthesize -to_generic -effort high
set_attr dft_scan_map_mode tdrc_pass $DESIGN
synthesize -to_mapped -effort high
report gates
check_dft_rules
synthesize -incremental -effort high
report dft_registers
define_dft scan_chain -name chain1 -create_ports -sdi TDI -sdo TDO
set_attr dft_mix_clock_edges_in_scan_chains true $DESIGN
set_attr dft_min_number_of_scan_chains 4 $DESIGN
connect_scan_chains -dont_exceed_min_number_of_scan_chains -auto_create_chains -
    preview $DESIGN
report qor
```

Figure 2.28: Mains command to add a scan-chain into the design

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

No, we use a slow clock for the scan-chain and add the port : TDI, TDO, scan_en, test_mode. And we select the same I/O delay.

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

```
create_clock -name clk_200MHz -period 5 [get_port CK]
create_clock -name clk25MHz -period 40 [get_port scan_clk]

set_input_delay 0.05438 -clock clk25MHz [get_port scan_en]
set_input_delay 0.05438 -clock clk25MHz [get_port test_mode]
set_input_delay 0.05438 -clock clk25MHz [get_port scan_clk]
set_input_delay 0.05438 -clock clk25MHz [get_port TDI]
set_output_delay 0.05438 -clock clk25MHz [get_port TDO]

set_input_delay 0.05438 -clock clk_200MHz [get_port RESET_N]
...
```

Figure 2.29: Few constrains for test

Writing outputs

Open the generated netlist and check if everything seems 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 ?

We can see as different : The precision about the wireloadmodel, set of capacitance unit and clock unit, setclockgating, some setdontuse (the `set_dont_use` command is used to exclude cells from the target library. Design Compiler does not use these excluded cells during optimization.)

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 ?

Equivalence checking is the formal verification of a design, it uses boolean logic to verify the veracity of a design or if the design is equivalent after operations on it, for example optimisation. It is necessary to verify that even after that optimisation phase the function is still the same. It is impossible to prove a circuit without equivalence checking, there are too many cases for human lifetime of testing.

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

LEC does not work.

SoC Encounter

Environment

Design import

The netlists

The physical Libraries

For Technology/Physical Libraries, select LEF files and fill in the correct LEF files. What is a LEF file ?

Layer Exchange Format : Its a lite physical library for abstract model different from library used to GDSII

Global Nets : Power Nets

For the Power section, enter the name of the power nets. For "Power Nets" enter, let's say "vdd_core" and for "Ground Nets" "gnd_core". Why do we have to enter this power net information ?

Because those are the future global connection of VDD and GND for the whole die. It will be used for the ring/stripes.

IO assignment file

What is an IO assignment file ?

IO Assignement File : It is an optional file that is used to instruct the tool where to place IO pins. Without any file the tool will determine these locations automatically. [1]

Check Design

Check the standard cell number, the area, the number of instances, the number of primary inputs/outputs. Is it ok ?

Currently everything is OK !

```
rc:/> report summary
...
Instance        Cells  Cell Area  Net Area  Total Area     Wireload
-----------------------------------------------------------------------------
MIPS_32_1P_MUL_DIV  14439     225299         0      225299 area_216Kto240K (S)

encounter 1> checkDesign -netlist
```

[1] https://community.cadence.com/cadence_technology_forums/f/digital-implementation/1378/design-io-assignment-file-generation

```
##############################################################################
# Encounter Netlist Design Rule Check
# Sun Feb 23 16:02:31 2020
##############################################################################
Design: MIPS_32_1P_MUL_DIV

------ Design Summary:
Total Standard Cell Number   (cells) : 14439
Total Block Cell Number      (cells) : 0
Total I/O Pad Cell Number    (cells) : 0
Total Standard Cell Area     ( um^2) : 225299.05

```

Are there multi-driver nets ? What is that ? Is it a problem ?

There is no multi-driver nets :[2]

```
encounter 1> checkDesign -netlist
...
Number of nets with multiple drivers           : 0

##############################
example of multi-driver net :
always @ (posedge CLK)
y = y + 1;

always @ (posedge CLK2)
y = y + 3;

```

Now check that the SDC is correctly read, run: `report_clocks` **Is it ok ? Explain L->L, L->LT, T->LL and T->LT.**

Everything is still ok.

- L->L : we didn't find that.
- L->LT : we didn't find that.
- T->LL : we didn't find that.
- T->LT : we didn't find that.

```
encounter 2> report_clocks
+-------------------------------------------------------------------------------+
|                              Clock Descriptions                               |
|-------------------------------------------------------------------------------|
|            |        |       |         |       |       |       Attributes      |
|------------+--------+-------+---------+-------+-------+-----------------------|
| Clock Name | Source |  View | Period  |  Lead | Trail | Generated | Propagated |
|------------+--------+-------+---------+-------+-------+-----------+------------|
| clk_200MHz |   CK   | setup |  5.000  | 0.000 | 2.500 |     n     |     n      |
+-------------------------------------------------------------------------------+

```

[2] https://wiki.nus.edu.sg/display/EE2020DP/%5BDRC+23-20%5D+Rule+violation+%28MDRV-1%29+Multiple+Driver+Nets

Reload encounter with the new SDC and check with report_clock that the new command is correctly understood. `set_propagated_clock [all_clocks]`

We can see that the clock is now propagated in the section Propagated inside the section Attributes.

```
encounter 1> report_clocks
+-------------------------------------------------------------------------------+
|                              Clock Descriptions                               |
|-------------------------------------------------------------------------------|
|            |        |       |        |       |       |       Attributes       |
|------------+--------+-------+--------+-------+-------+------------------------|
| Clock Name | Source |  View | Period |  Lead | Trail | Generated | Propagated |
|------------+--------+-------+--------+-------+-------+-----------+------------|
| clk_200MHz |   CK   | setup |  5.000 | 0.000 | 2.500 |     n     |     y      |
+-------------------------------------------------------------------------------+

```

Floorplanning

You can see beside the label "Core Utilization" the classical number 0.7. This specifies the core area, here it means that 70% of the design is filled with cells. You can play with this number to change the core area. This has an impact on the congestion of the design, in other words on the routability of the design. Explain why.

Our design needs to have some space on the floorplaning. The last 30% will be used for IOpad, rings etc. It is the allocated space for our design.

Power Planning

Connecting to Global Nets

check if everything is correct ? No ? Please correct and rerun. Click "Close". If you do not find the name of the power pins in the LEF try "vdd" and "gnd".

Everything is OK.

Add Rings

These layers are noted V, the first ones are noted H. Explain that :

The one noted V are for vertical and the one noted H are for horrizontal.

Add Stripes, SRoute, Saving the floorplan

Everything is still OK.

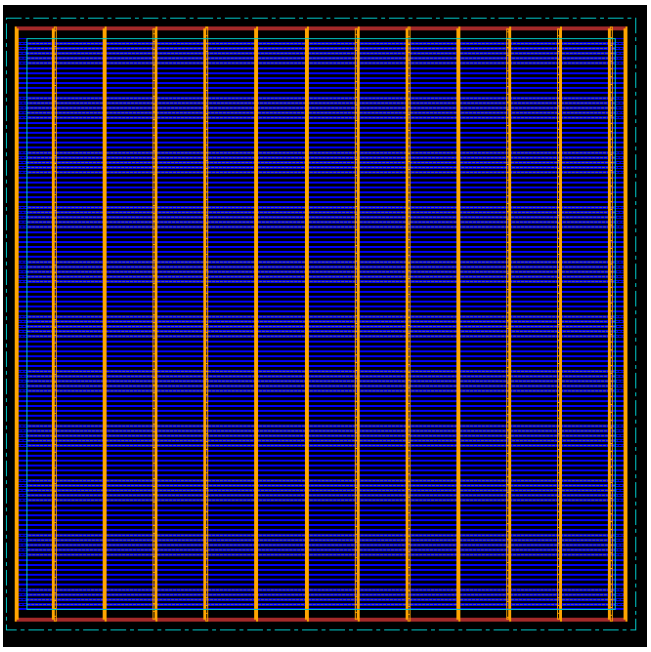

Figure 3.1: Result of the floor and power plan.

Place Design

Standard Cells Placement

Result :

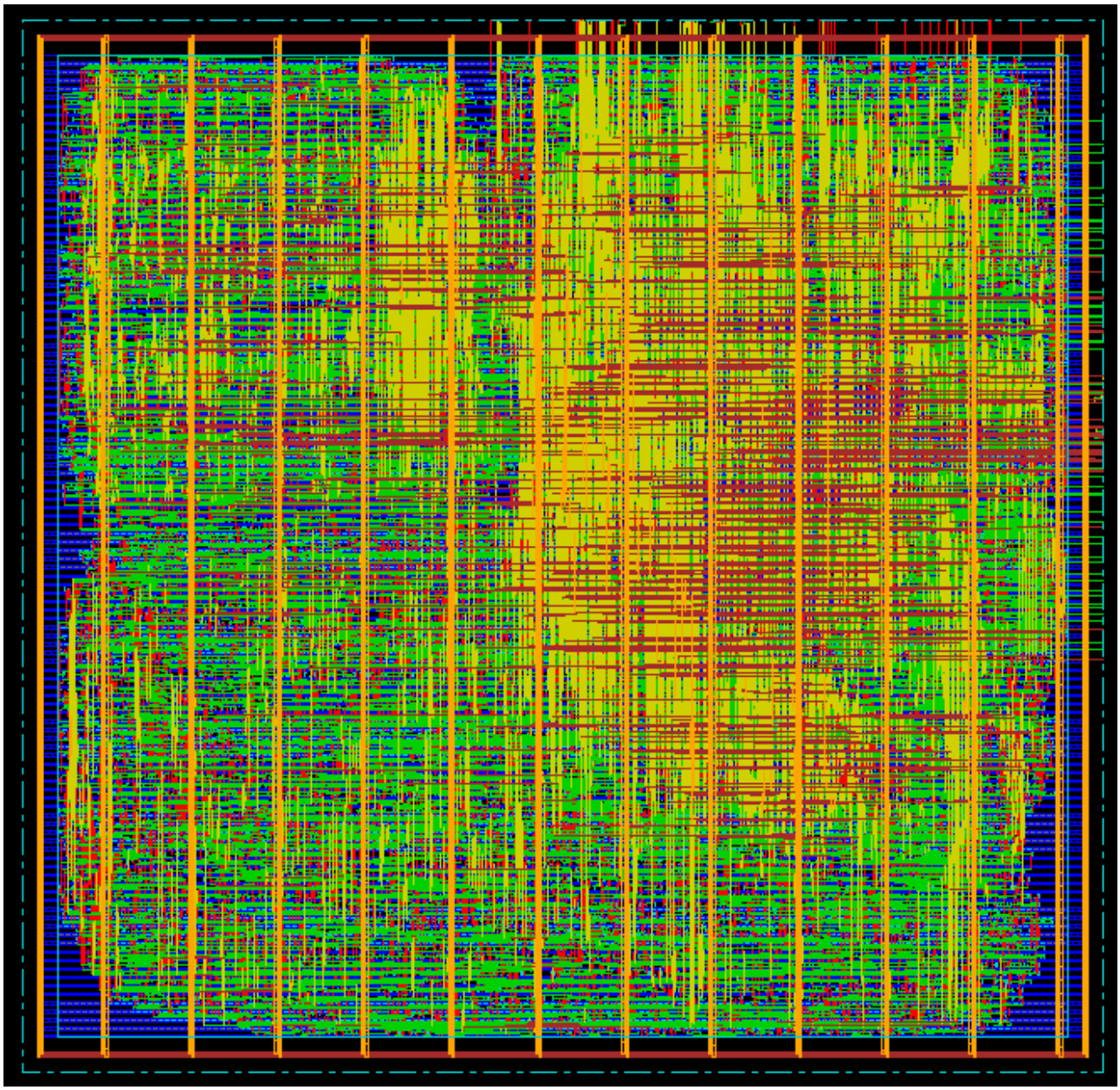

Figure 3.2: Result of place standard cells.

Trial Route

We didn't find the diamonds. We have remarked that there is a problem about the size of a cell that is too big :

```
**WARN: (ENCTR-3824): The standard cell row height appears to be too large.
Estimated standard cell row height = 20 metal3 tracks

```

Rest of the output is :

```
encounter 2> *** Starting trialRoute (mem=822.8M) ***

There are 0 guide points passed to trialRoute for fixed pins.
There are 0 guide points passed to trialRoute for pinGroup/netGroup/pinGuide pins
.
Options:  -maxRouteLayer 6 -gcellSizeX 4 -gcellSizeY 2 -floorPlanMode -noPinGuide
```

```
**WARN: (ENCTR-3824): The standard cell row height appears to be too large.
Estimated standard cell row height = 20 metal3 tracks

Phase 1a-1d Overflow: 0.02% H + 1.54% V (0:00:00.1 812.5M)

Phase 1e-1e Overflow: 0.00% H + 0.71% V (0:00:00.0 813.2M)

Phase 1l Overflow: 0.02% H + 3.98% V (0:00:00.2 810.4M)


Congestion distribution:

Remain  cntH    cntV
-------------------------------------
-5: 0  0.00%  48   0.56%
-4: 0  0.00%  26   0.30%
-3: 0  0.00%  28   0.33%
-2: 0  0.00%  37   0.43%
-1: 2  0.02%  65   0.76%
-------------------------------------
0:  6  0.07%  63   0.74%
1:  9  0.11%  80   0.93%
2:  16   0.19%  76   0.89%
3:  26   0.30%  101  1.18%
4:  20   0.23%  107  1.25%
5:  8489  99.08%  7937  92.64%


Total length: 6.906e+05um, number of vias: 109396
M1(H) length: 1.085e+04um, number of vias: 57458
M2(V) length: 2.167e+05um, number of vias: 40991
M3(H) length: 2.678e+05um, number of vias: 8296
M4(V) length: 1.204e+05um, number of vias: 2522
M5(H) length: 7.086e+04um, number of vias: 129
M6(V) length: 4.019e+03um

Peak Memory Usage was 818.7M
*** Finished trialRoute (cpu=0:00:01.0 mem=819.8M) ***

```

Timing Analysis

Choose the correct "Design Stage" and explain your choice

We are currently Pre-CTS, just before the clock tree synthesis but after the place.

Check if the the worst negative slack (WNS) is positive. Is the timing met ?

As we can see the timing is not met, WNS is negative.

```
 timeDesign Summary

+--------------------+---------+---------+---------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | in2reg  | reg2out | in2out  | clkgate |
+--------------------+---------+---------+---------+---------+---------+---------+
|           WNS (ns):| -24.406 | -24.406 | -24.126 |   N/A   |   N/A   |   N/A   |
|           TNS (ns):|-25961.3 |-25961.3 |-25216.7 |   N/A   |   N/A   |   N/A   |
|    Violating Paths:|  2148   |  2148   |  1770   |   N/A   |   N/A   |   N/A   |
|          All Paths:|  6714   |  6697   |  1798   |   N/A   |   N/A   |   N/A   |
+--------------------+---------+---------+---------+---------+---------+---------+
```

Optimize the design with the `optDesign` command. Redo a `timeDesign`. Is the timing met now ? What is the slack ?

Timing is met now, slack is positive.

```
 optDesign Final Summary

+--------------------+---------+---------+---------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | in2reg  | reg2out | in2out  | clkgate |
+--------------------+---------+---------+---------+---------+---------+---------+
|           WNS (ns):|  0.124  |  0.124  |  0.523  |   N/A   |   N/A   |   N/A   |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |   N/A   |   N/A   |   N/A   |
|    Violating Paths:|    0    |    0    |    0    |   N/A   |   N/A   |   N/A   |
|          All Paths:|  6714   |  6697   |  1798   |   N/A   |   N/A   |   N/A   |
+--------------------+---------+---------+---------+---------+---------+---------+
```

Report the timing and check in the floorplan where is the path.

```
encounter 4> report_timing
Path 1: MET Setup Check with Pin HI_RW_reg[29]/CP
Endpoint:   HI_RW_reg[29]/D     (^) checked with  leading edge of 'clk_200MHz'
Beginpoint: S00MUL7_RM_reg[4]/Q (^) triggered by  leading edge of 'clk_200MHz'

```

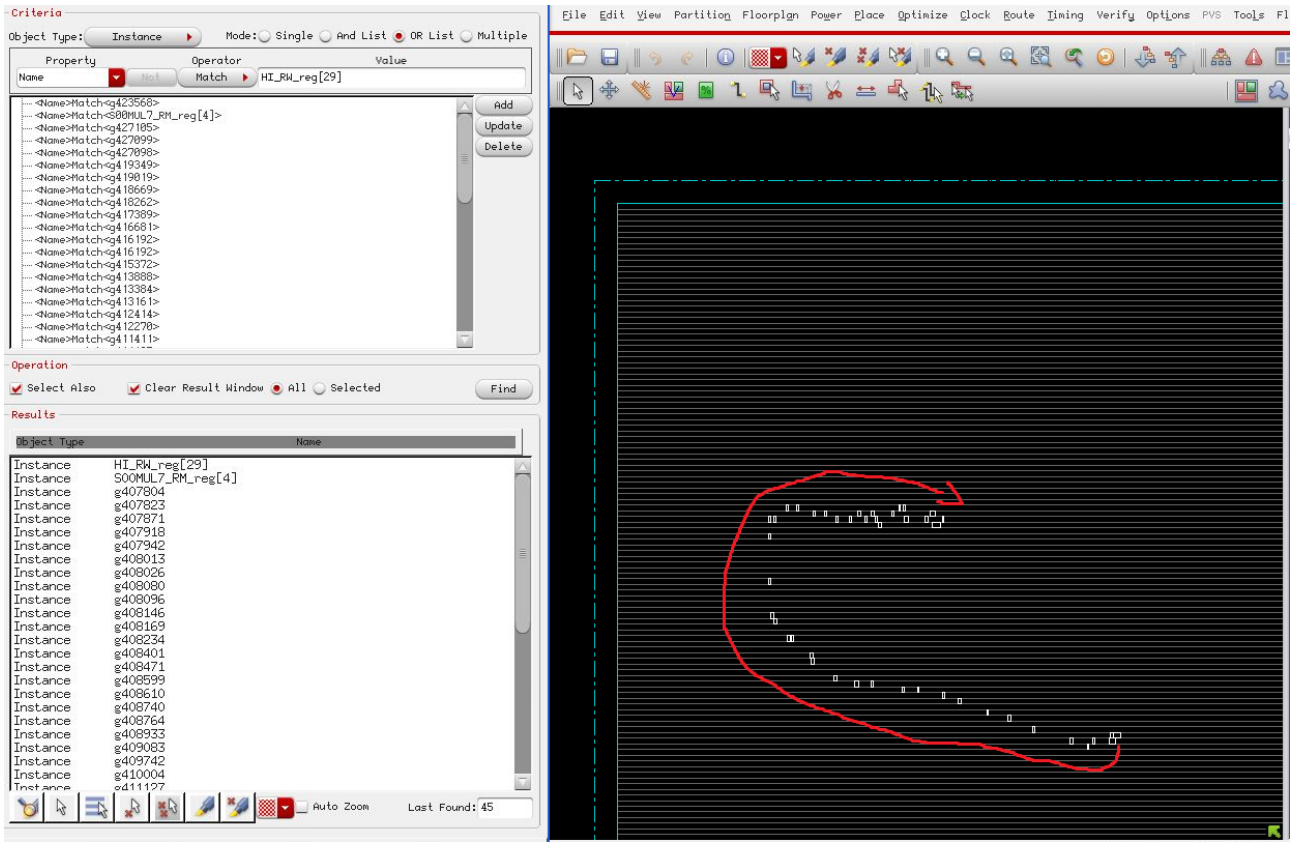

Figure 3.3: Longest path found between the two regs.

What is the TNS ? Why is important to have this information with the WNS ?

The worst negative slack is means different things it depends on the sign:

- Postive : The timing is respected and we arrive in time.
- Negative : The timing is not respect and the data arrive to late.

The Total negative slack is the sum of all negative slack in our design.

- If it's 0 then it means the design meets the timing.
- If it's positive, then it means that there are some negative slacks in the design and the timing are not met.
- It will never be negative.

The "Total Negative Slack (TNS)" is the sum of the (real) negative slack in your design. If 0, then the design meets timing. If it is a positive number, then it means that there is negative slack in the design (hence your design fails). It cannot be negative.

CTS: Clock Tree Synthesis

Why are there specific cells for the clock tree ?

The clock tree cells are specific because they contain specific buffer that will have a really good control of rise and fall time of clock edges. They keep noise from feeding back into another system and try not to create delay in the signal. They allow the clock to have a high fanout.[3]

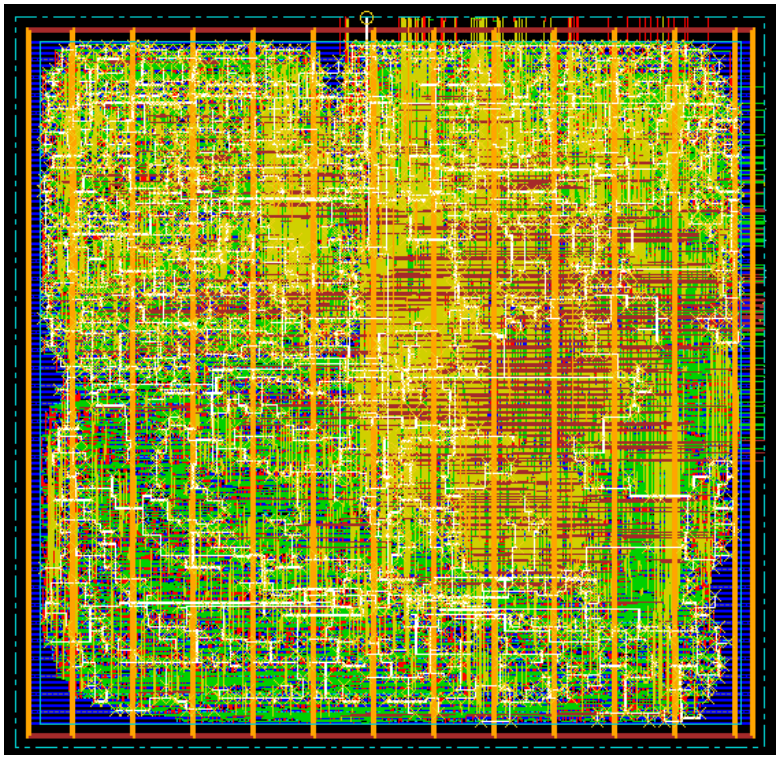

Figure 3.4: In white we can see the clock tree.

Is the timing met post CTS ? What is your worst timing path ?

```
encounter 10> report_timing
Path 1: MET Setup Check with Pin HI_RW_reg[29]/CP
Endpoint:   HI_RW_reg[29]/D     (^) checked with  leading edge of 'clk_200MHz'
Beginpoint: S00MUL7_RM_reg[4]/Q (^) triggered by  leading edge of 'clk_200MHz'



timeDesign Summary

+--------------------+---------+---------+---------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | in2reg  | reg2out | in2out  | clkgate |
+--------------------+---------+---------+---------+---------+---------+---------+
|           WNS (ns):|  0.091  |  0.091  |  0.826  |   N/A   |   N/A   |   N/A   |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |   N/A   |   N/A   |   N/A   |
```

[3] https://bitcointalk.org/index.php?topic=128286.0

```
|    Violating Paths:|    0    |    0    |    0    |   N/A   |   N/A   |   N/A   |
|           All Paths:|  6714   |  6697   |  1798   |   N/A   |   N/A   |   N/A   |
+--------------------+---------+---------+---------+---------+---------+---------+
```

Maybe an optDesign postCTS will optimize that ? Is it better ?

As we can see it is a bit better

```
optDesign Final Summary

+--------------------+---------+---------+---------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | in2reg  | reg2out | in2out  | clkgate |
+--------------------+---------+---------+---------+---------+---------+---------+
|           WNS (ns):|  0.113  |  0.113  |  0.824  |   N/A   |   N/A   |   N/A   |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |   N/A   |   N/A   |   N/A   |
|    Violating Paths:|    0    |    0    |    0    |   N/A   |   N/A   |   N/A   |
|           All Paths:|  6714   |  6697   |  1798   |   N/A   |   N/A   |   N/A   |
+--------------------+---------+---------+---------+---------+---------+---------+
```

Now that the clock tree is built, you can check if you meet the timing in hold mode. Run: `timeDesign -postCTS -hold` **Is the timing met in hold mode ?**

Timing is not met, we will try an optimization for both setup and hold.

```
timeDesign Summary

+--------------------+---------+---------+---------+---------+---------+---------+
|     Hold mode      |   all   | reg2reg | in2reg  | reg2out | in2out  | clkgate |
+--------------------+---------+---------+---------+---------+---------+---------+
|           WNS (ns):| -0.176  |  0.037  | -0.176  |   N/A   |   N/A   |   N/A   |
|           TNS (ns):| -11.026 |  0.000  | -11.026 |   N/A   |   N/A   |   N/A   |
|    Violating Paths:|   84    |    0    |   84    |   N/A   |   N/A   |   N/A   |
|           All Paths:|  6714   |  6697   |  1798   |   N/A   |   N/A   |   N/A   |
+--------------------+---------+---------+---------+---------+---------+---------+
```

Report the timing in hold mode with the report_timing command. In order to distinguish the setup and hold mode report_timing uses “early” and “late”. Which one is which one ? Explain.

Timing is now met post CTS on holde mode.

```
timeDesign Summary

+--------------------+---------+---------+---------+---------+---------+---------+
|     Hold mode      |   all   | reg2reg | in2reg  | reg2out | in2out  | clkgate |
+--------------------+---------+---------+---------+---------+---------+---------+
|           WNS (ns):|  0.001  |  0.037  |  0.001  |   N/A   |   N/A   |   N/A   |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |   N/A   |   N/A   |   N/A   |
|    Violating Paths:|    0    |    0    |    0    |   N/A   |   N/A   |   N/A   |
|          All Paths:|  6714   |  6697   |  1798   |   N/A   |   N/A   |   N/A   |
+--------------------+---------+---------+---------+---------+---------+---------+



timeDesign Summary

+--------------------+---------+---------+---------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | in2reg  | reg2out | in2out  | clkgate |
+--------------------+---------+---------+---------+---------+---------+---------+
|           WNS (ns):|  0.113  |  0.113  |  0.824  |   N/A   |   N/A   |   N/A   |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |   N/A   |   N/A   |   N/A   |
|    Violating Paths:|    0    |    0    |    0    |   N/A   |   N/A   |   N/A   |
|          All Paths:|  6714   |  6697   |  1798   |   N/A   |   N/A   |   N/A   |
+--------------------+---------+---------+---------+---------+---------+---------+


```

We didn’t found early and late but it is written Hold mode and Setup mode.

Routing the design: NanoRoute

Is the post-route timing met ? No ? Let's optimize the timing, Is the timing better ? Please report the worst timing path. What is it ?

Timing were not met. Timing are met after optimisation `setAnalysisMode -analysisType onChipVariation` BEFORE `optDesign -postRoute`, else error ENCOPT 6080 (Thanks to Quentin).

```
    timeDesign Summary

    +--------------------+---------+---------+---------+---------+---------+---------+
    |     Setup mode     |   all   | reg2reg | in2reg  | reg2out | in2out  | clkgate |
    +--------------------+---------+---------+---------+---------+---------+---------+
    |           WNS (ns):|  0.112  |  0.112  |  0.819  |   N/A   |   N/A   |   N/A   |
    |           TNS (ns):|  0.000  |  0.000  |  0.000  |   N/A   |   N/A   |   N/A   |
    |    Violating Paths:|    0    |    0    |    0    |   N/A   |   N/A   |   N/A   |
    |          All Paths:|  6714   |  6697   |  1798   |   N/A   |   N/A   |   N/A   |
    +--------------------+---------+---------+---------+---------+---------+---------+
```

Worst path :

```
    Path 1: MET Setup Check with Pin HI_RW_reg[30]/CP
    Endpoint:   HI_RW_reg[30]/D     (v) checked with  leading edge of 'clk_200MHz'
    Beginpoint: S00MUL7_RM_reg[4]/Q (v) triggered by  leading edge of 'clk_200MHz'
    Analysis View: setup
    Other End Arrival Time          0.261
    - Setup                         0.173
    + Phase Shift                   5.000
    = Required Time                 5.088
    - Arrival Time                  4.976
    = Slack Time                    0.112
```

The worst path is inside the multiplier.

Adding filler cells

What is a filler cell ? Why do we need it ?

The design rules tell us that there is a need to have continuity between N-well, P-well and Power continuity. If not, cells need to be spaced farther apart. It will be a loss of space. Moreover the wells need to be tied to a single global VDD/VSS. Thus, all the wells of standard cells have to be tied to maintain continuity. It is impossible for a design to fill 100% of the die with only regular cells. Filler cells are used to fill this space. They reduce DRC violation and help maintain power rail connection continuity. [4]

Choose “Place ¿ Physical Cells ¿ Add Filler”. Why do call these cells “physical cells” ?

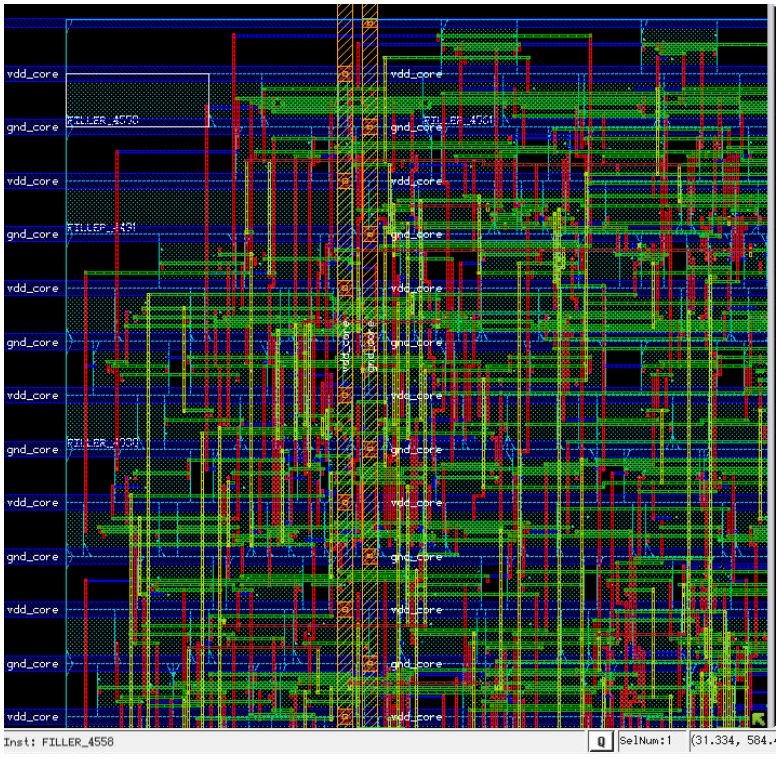

Figure 3.5: Here we can see a filler cell inside the design. It is called physical cell because they physically take some place on the design.

```
encounter 14> *INFO: Adding fillers to top-module.
*INFO:   Added 10 filler insts (cell FILLER2V5CAP64 / prefix FILLER).
...
*INFO: Total 4613 filler insts added - prefix FILLER (CPU: 0:00:00.1).
For 4613 new insts, *** Applied 2 GNC rules (cpu = 0:00:00.0)
Saving Drc markers ...
... 0 markers are saved ...
*INFO: Checking for DRC violations on added fillers.
...
*INFO: Iteration 0-#5, Found 0 DRC violation  (real: 0:00:00.0).
...
... 0 markers are loaded ...
*INFO: End DRC Checks. (real: 0:00:04.0 ).
*INFO: Replaced 175 fillers which had DRC vio's, with 395 new fillers.
```

[4] https://www.quora.com/Why-do-we-use-filler-cells-for-N-well-continuity

```
15
```

Power Rail Analysis

It does not work.But I found a course that explain the whole flow[5]

```
    Rail Analysis is unsuccessful due to errors.

    Finished Rail Analysis at 20:08:37 02/23/2020 (cpu=0:00:00, real=0:00:01, peak
    mem=1083.01MB)
    Current Power Analysis resource usage: (total cpu=0:48:36, real=3:48:48, mem
    =1083.01MB)
    voltus_rail exited unsuccessfully.
    **ERROR: (PRL-387): "Rail Analysis" failed to finish successfully.
    **ERROR: (EMS-19):  emsIssueMsg failed because the message number 1 does not
    exist in the catalog file /users/soft/opus/Linux/EDI-13.24.026/share/cdssetup/
    errormessages/fe/vts.msg.
    Update the catalog file with correct data and re-register the message catalog.
    **ERROR: (VOLTUS-5003): Net/Group-Net '' does not exist in the design '
    MIPS_32_1P_MUL_DIV'. Use always-on net name as the group-net name.
    **ERROR: (VOLTUS-5062): You must load the state directory before plotting 'ir'.
    **ERROR: **ERROR: (VOLTUS_ERA-2395):  Check error messages in log file.

```

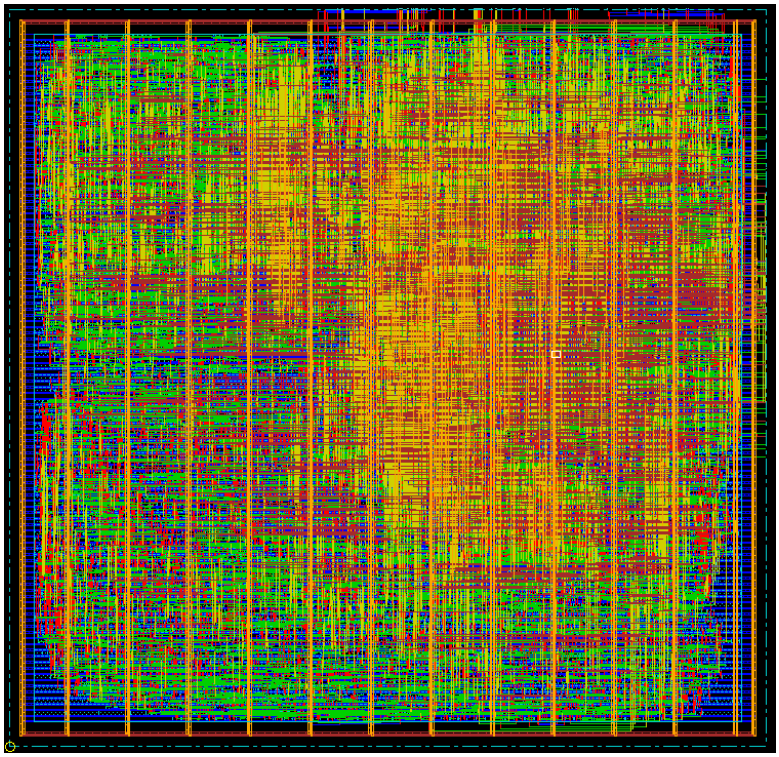

Figure 3.6: Ending result.

[5] http://www.ee.ncu.edu.tw/~jfli/vlsidi/lecture/SOC_Encounter.pdf

Bibliography

[1] A. B. Kahng K. D. Boese and S. Mantik. On the relevance of wire load models. *ACM International Workshop on System-Level Interconnect Prediction*, pages 91–98, April 2001.

List of Figures

Annexe

Glossary :

- EDA : Electronic design automation
- DFT : Design For Test
- CTS : Clock Tree Synthesys :
- STA : Static Timing Analysis
- DRC : Design Rule Check
- LVS : Logical Vs Schematic
- IC : Integrated Circuit
- PCB : Printed Circuit Board
- Plot/Pad :
- FP : Floor Planing
- ATPG : Automatic Test Pattern generation
- stuck-at fault : is a fault model for ATPG
- PVT : Process Voltage Temperature
- Fanout : how much cells my signal can attack without being degraded
- .lib : LIBerty : Technology library source files containing all required information for synthesis and static timing analysis
- library max (WC=Worst Case) : The timing in this kind of library are the longest (Max path/-Max Data path). They are used for setup analysis.
- library min (BC=Best Case) : The timing in the kind of library are the shortest (Min path/Min Data path). They are used for hold analysis.
- WNS : Worst negative slack : The longest path in the design
- TNS : Total Negative Slack : The Sum of all the longest path in the design
- WLM : Wire-Load Model : Statistical Model that
- WHS : Worst Hold Slack (¡=¿WNS)
- THS : Total Hold Slack (¡=¿TNS)
- WPWS : Worst Pulse Width Slack
- TPWS : Total Pulse Width Slack
- DC : Design Compiler
- DCP : Design Compiler Physical
- RC : RTL Compiler
- RCP : RTL Compiler Physical

- SS : Setup Slack : Real time between data latch arrival and clock latch edge, less setup timing. To be TC, all SS must be positive or void.
- HS : Hold Slack : Real time between end of data established and end of hold timing. To be TC, all HS must be positive or void.
- CDC : Clock Domain Crossing
- GDSII : Graphic Database System II
- PnR : Place .& Route
- floor planning : Macro planing did by human to help EDA tool
- spare cells : set of logic cells
- ECO : Engineering Change Order : using of spare cells to prevent any change in design after generated lithographic photomasks, its use to reduced the cost of a design error detecting in back-end flow
- Boundary Scan :
- JTAG : Join Test Action Group :
- Trial Route : quick route to predict congestion route, is without fixing DRC or LVS violation
- Timing Met
- Signoff : Abstract to real, generating GDSII
- LEF : Layer Exchange Format : Its a lite physical library for abstract model different from library used to GDSII
- SDC : Synopsis Design Constraints : Define timing constraints like input delay, output delay, clock
- Shmoo plots ("excel" table)