

EXPERIMENT #4 ADDENDUM

Timing Analysis of 16-bit Adders

I. OBJECTIVE

This is an addendum exercise for Experiment #4, which has you run a critical path analysis for the three adders you created in Experiment #4. In your lab report (for extra credit), do a manual critical path analysis for each of the adders (you may assume that all gates have the same delay). For each of the three adders, compare the manual critical path analysis with the analysis provided by the TimeQuest tool. Note that the critical path analysis is related to but not equivalent to the fMax graph you are required to provide for the standard Lab Report 4. The fMax only tells you the maximum speed of the aggregate design, whereas the critical path tells you the specific path that is holding back the design.

II. TUTORIAL FOR CARRY RIPPLE ADDER

1. Select ripple carry adder from top level module. Comment out the other two adders (see figure 1)

```
// Addition unit
ripple_adder adder    (.A(extended_SW[15:0]), .B(Out[15:0]), .cin(1'b0), .cout(S[16]), .S(S[15:0]) );
//lookahead_adder adder1a (.A(extended_SW[15:0]), .B(Out[15:0]), .cin(1'b0), .cout(S[16]), .S(S[15:0]) );
//select_adder adders    (.A(extended_SW[15:0]), .B(Out[15:0]), .cin(1'b0), .cout(S[16]), .S(S[15:0]) );
```

Figure 1

2. Full-compile the whole project. Because the timing constrain file (.sdc) is not included in the project, the *Time Analyzer* in **Table of Contents** window appears red. The red color indicates that there are physical I/O paths are not constrained, or there are paths violate path timing requirements (see figure 2)

Table of Contents

Flow Summary

Flow Settings

Flow Non-Default Global Settings

Flow Elapsed Time

Flow OS Summary

Flow Log

Analysis & Synthesis

Fitter

Assembler

Timing Analyzer

EDA Netlist Writer

Flow Messages

Flow Suppressed Messages

Flow Summary

<<Filter>>

Flow Status

Successful - Mon Aug 03 22:10:47 2020

Quartus Prime Version

18.1.0 Build 625 09/12/2018 SJ Lite Edition

Revision Name

adder2

Top-level Entity Name

adder2

Family

MAX 10

Device

10M50DAF484C7G

Timing Models

Final

Total logic elements

79 / 49,760 (< 1 %)

Total registers

20

Total pins

66 / 360 (18 %)

Total virtual pins

0

Total memory bits

0 / 1,677,312 (0 %)

Embedded Multiplier 9-bit elements

0 / 288 (0 %)

Total PLLs

0 / 4 (0 %)


UFM blocks

0 / 1 (0 %)

ADC blocks

0 / 2 (0 %)

Figure 3

3. Start *Time Analyzer* from **Tool->Time Analyzer** (See figure 4) or by click  from the software action bar

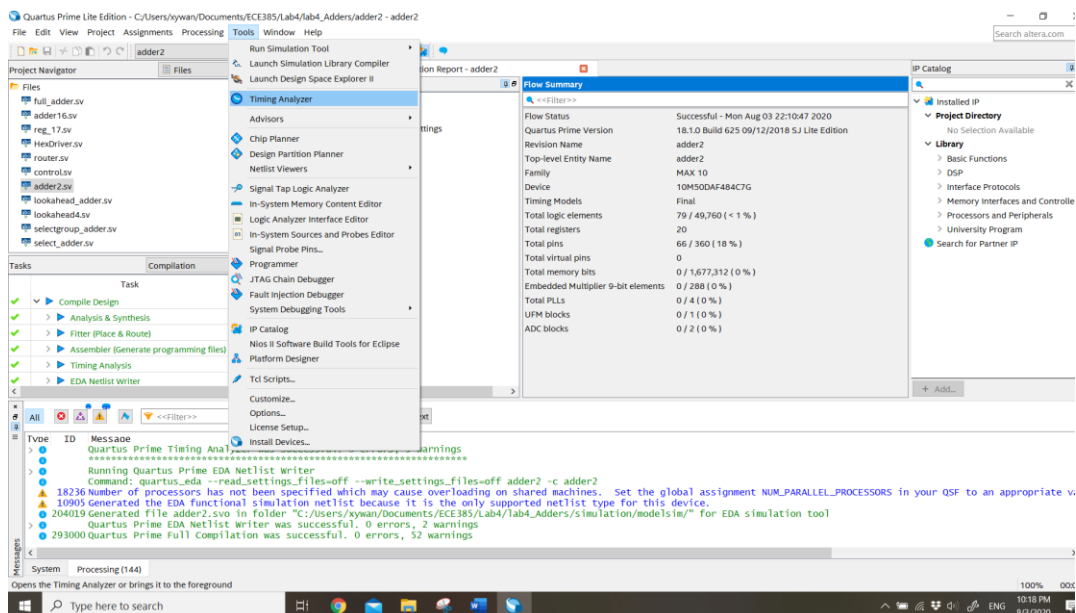


Figure 4

4. The TimeQuest Time Analyzer interface prompt is shown in figure 5. Choose “Create Timing Netlist” to create a timing netlist.

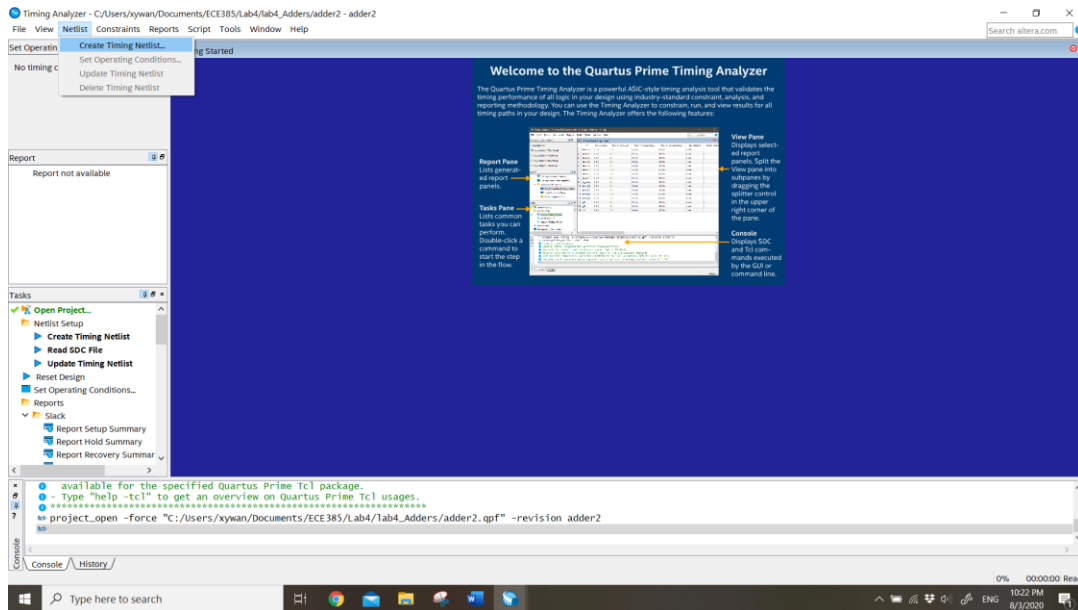


Figure 5

5. Choose default settings *Post-fit* and *Slow corner*, as shown in figure 6, and then click “OK”

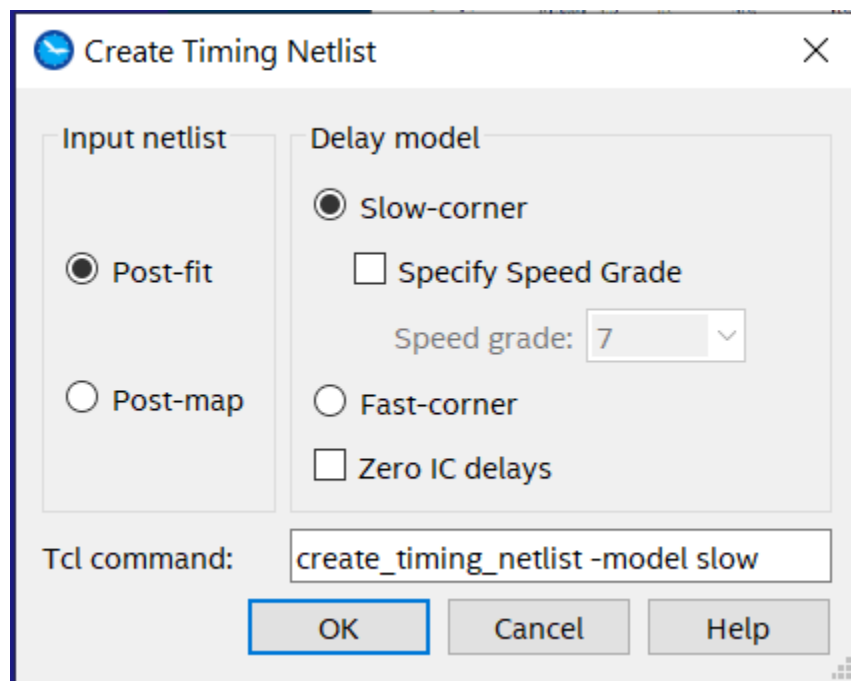


Figure 6

6. Select “Constraints -> Create Clock” to create a clock signal as shown in figure 7.

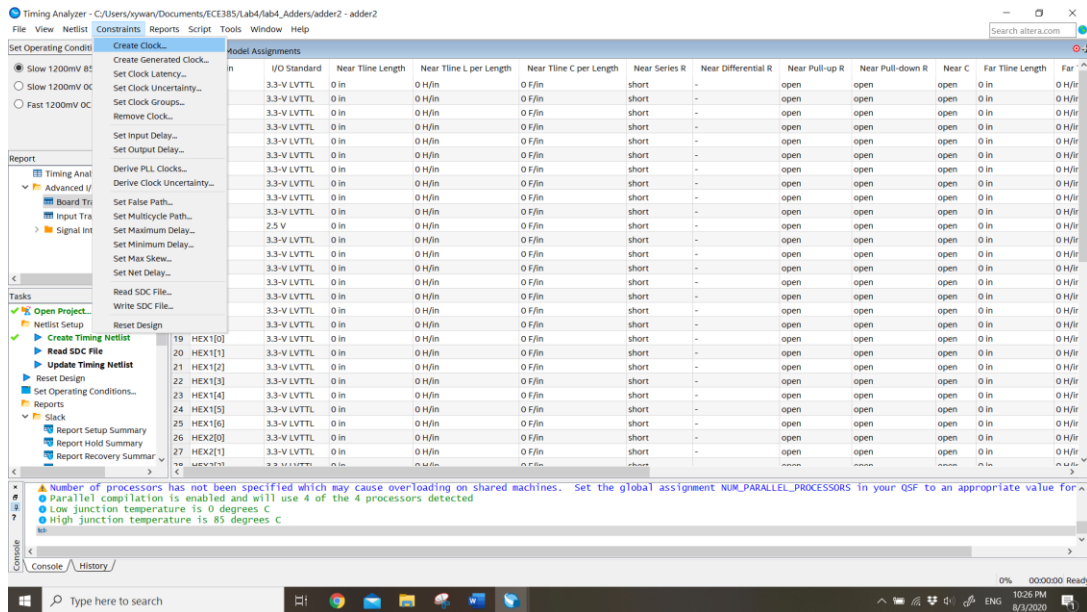


Figure 7

7. A window popped up as shown in figure 8. Give a clock name and set the period to be 20ns.

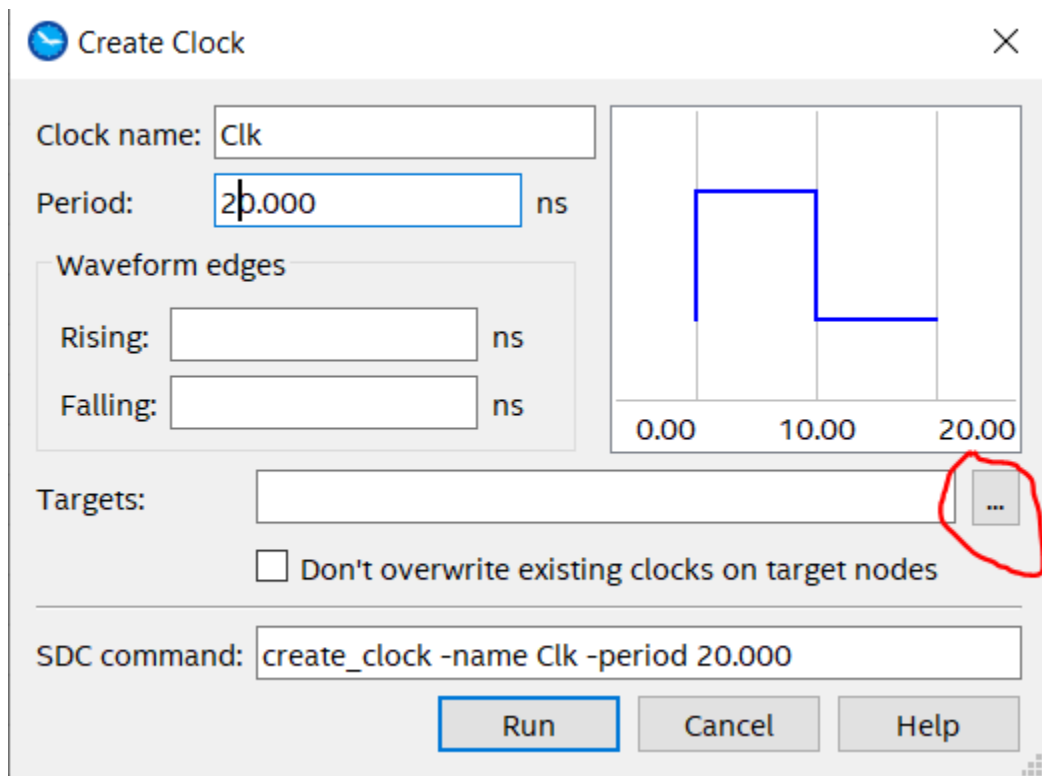


Figure 8

- Click “Targets” browsing icon in the red circle. A window is popped up as shown in figure 9. In “Collection” select “get_ports”, and then click “List”. All the I/O ports defined in top-level module is listed in the space below. Select “Clk” port. Then click “OK” and then “Run” to set.

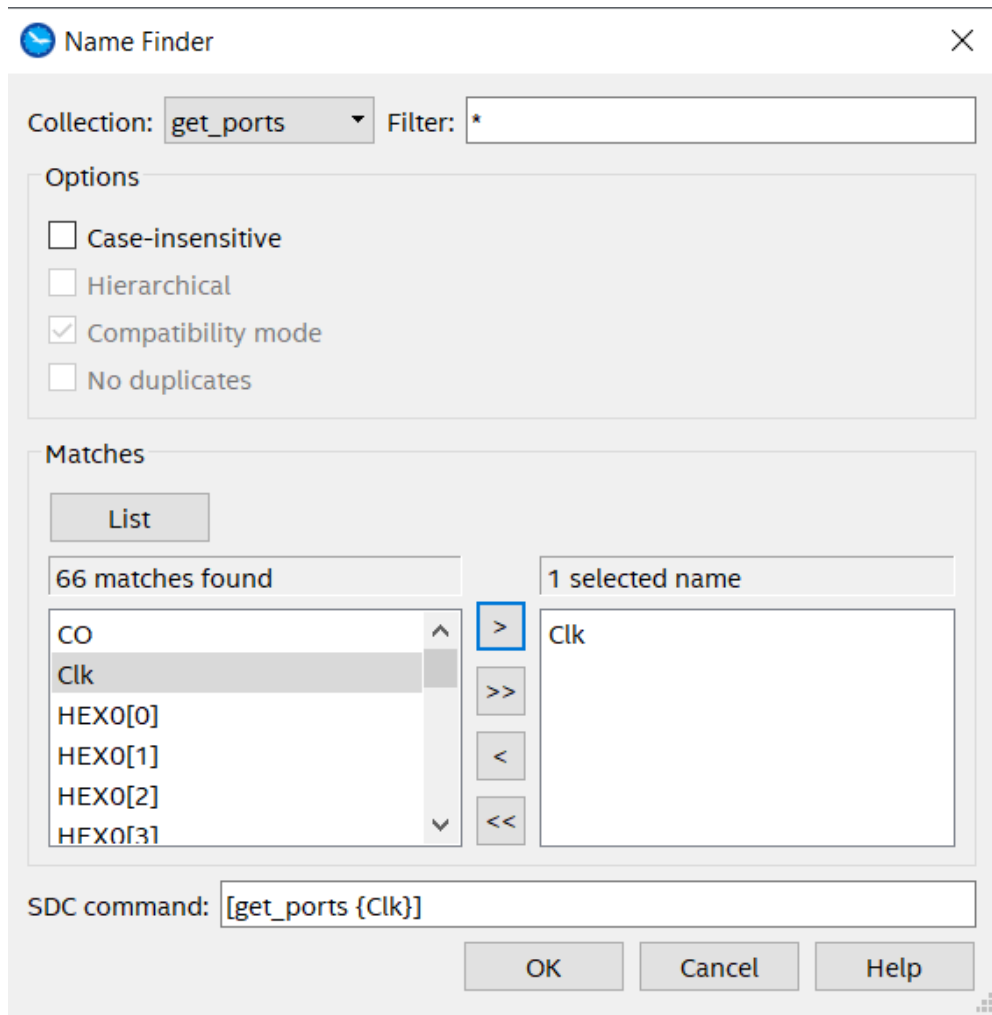


Figure 9

- Set input constrain by selecting “Constrain->set input delay” as shown in figure 10.

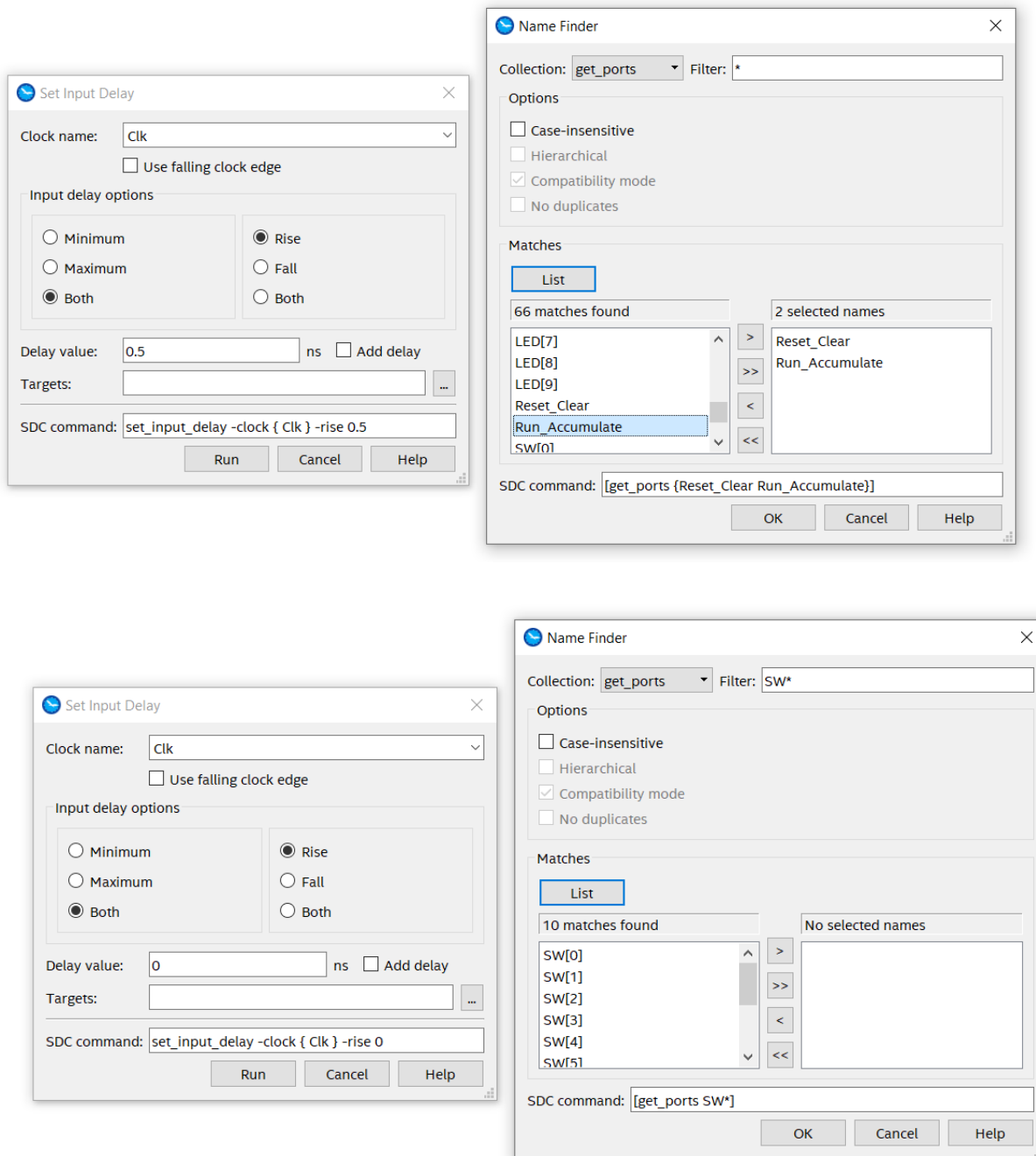


Figure 11

11. Repeat step 10 for output delay as shown in figure 12. Set output delay to be 0 ns for all the output ports.

Set Output Delay

Clock name: Clk

☐ Use falling clock edge

Output delay options

☐ Minimum ☒ Rise ☐ Fall ☐ Both

Delay value: 0 ns ☐ Add delay

Targets:

SDC command: set_output_delay -clock { Clk } -rise 0

Run Cancel Help

Name Finder

Collection: get_ports Filter: HEX*

Options

☐ Case-insensitive ☐ Hierarchical ☒ Compatibility mode ☐ No duplicates

Matches

List

42 matches found

HEX0[0]
HEX0[1]
HEX0[2]
HEX0[3]
HEX0[4]
HEX0[5]

No selected names

SDC command: [get_ports HEX*]

OK Cancel Help

Set Output Delay

Clock name: Clk

☐ Use falling clock edge

Output delay options

☐ Minimum ☒ Rise ☐ Fall ☐ Both

Delay value: 0 ns ☐ Add delay

Targets:

SDC command: set_output_delay -clock { Clk } -rise 0

Run Cancel Help

Name Finder

Collection: get_ports Filter: LED*

Options

☐ Case-insensitive ☐ Hierarchical ☒ Compatibility mode ☐ No duplicates

Matches

List

10 matches found

LED[0]
LED[1]
LED[2]
LED[3]
LED[4]
LED[5]

No selected names

SDC command: [get_ports LED*]

OK Cancel Help

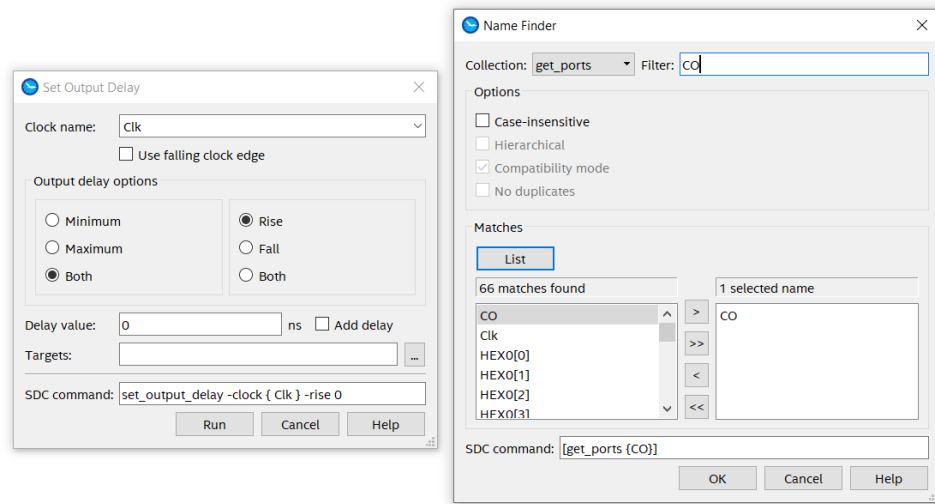


Figure 12

12. Once all the path are constrained. Click “ Write SDC File” in Tasks window (figure 13 left) to save all the constrains into “add2.out.sdc”, which will be saved in the project folder.

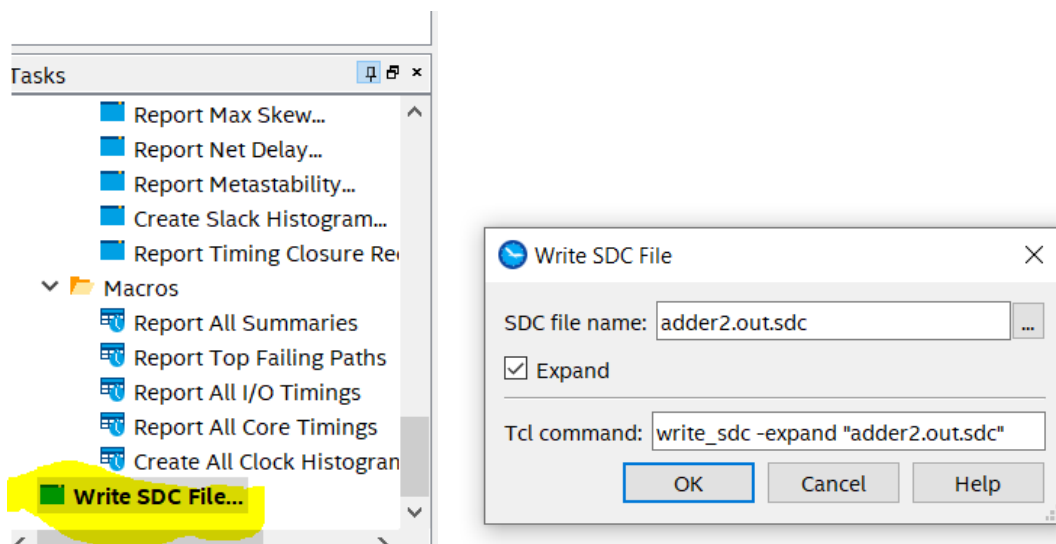


Figure 13

13. Going to the main interface of Quartus, select “Project -> add/remove files to project”. Add the sdc file into project as shown in figure 14. In the following, do a full-compilation. If all path are constrained and there is no path violating timing requirements, you will notice that the “Timing Analyzer” turns to orange instead of red.

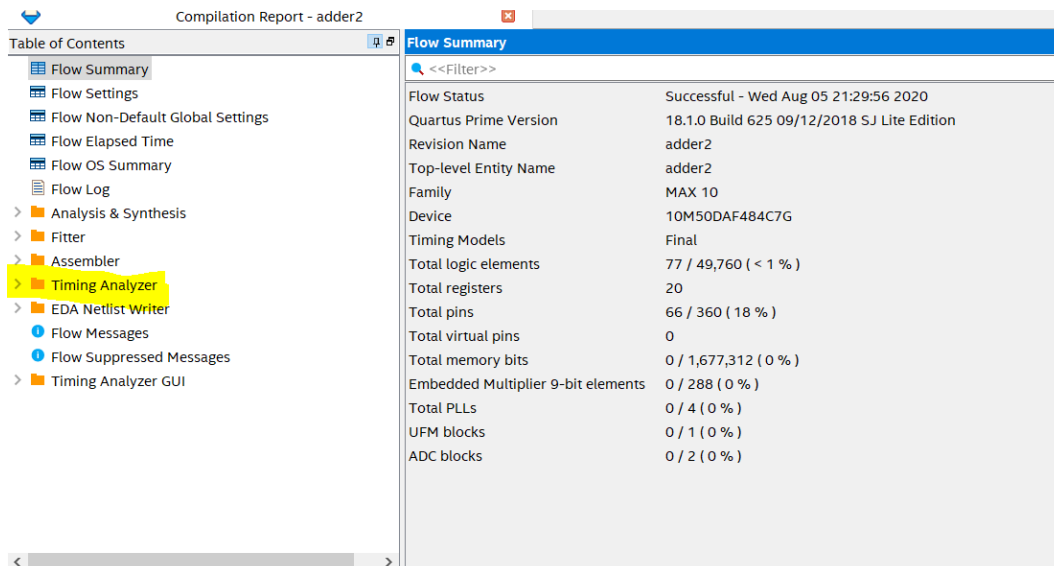
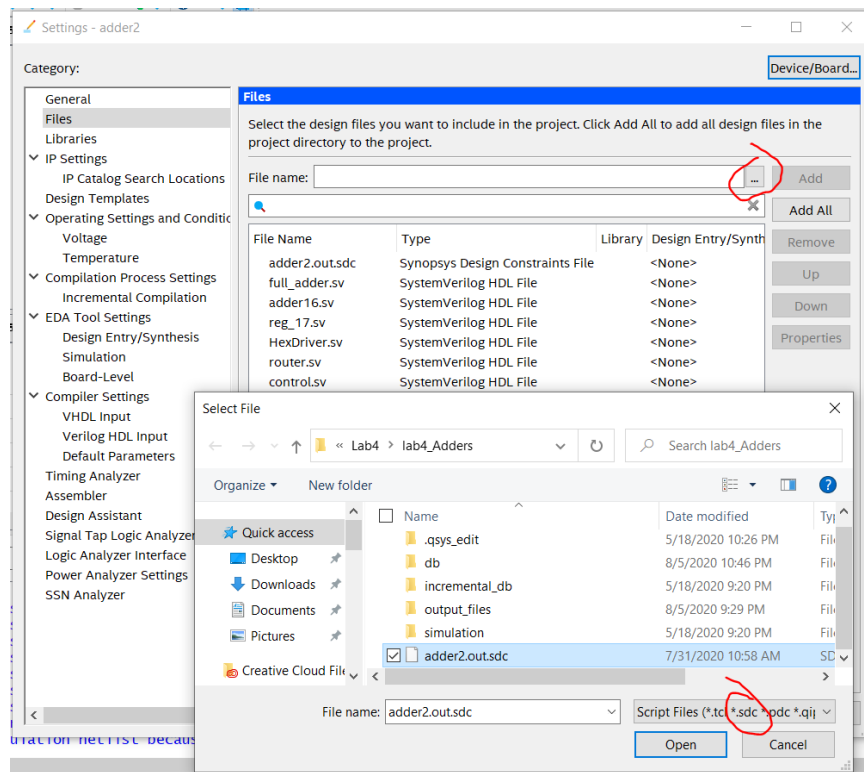


Figure 14

14. After compilation, launch “TimeQuest Timing Analyzer” again as shown in step 3. Click “Create Timing Netlist -> Read SDC File -> Update Timing Netlist” consecutively. Once you see the

three actions are green checked, which means the timing analysis has been completed successfully.

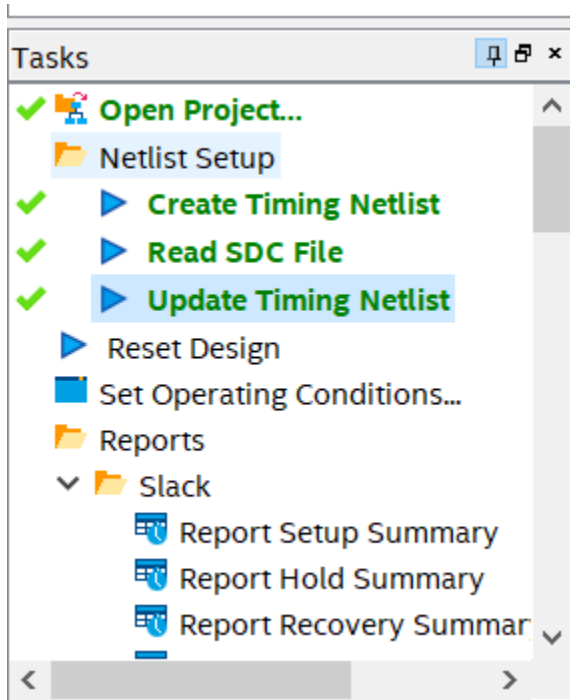
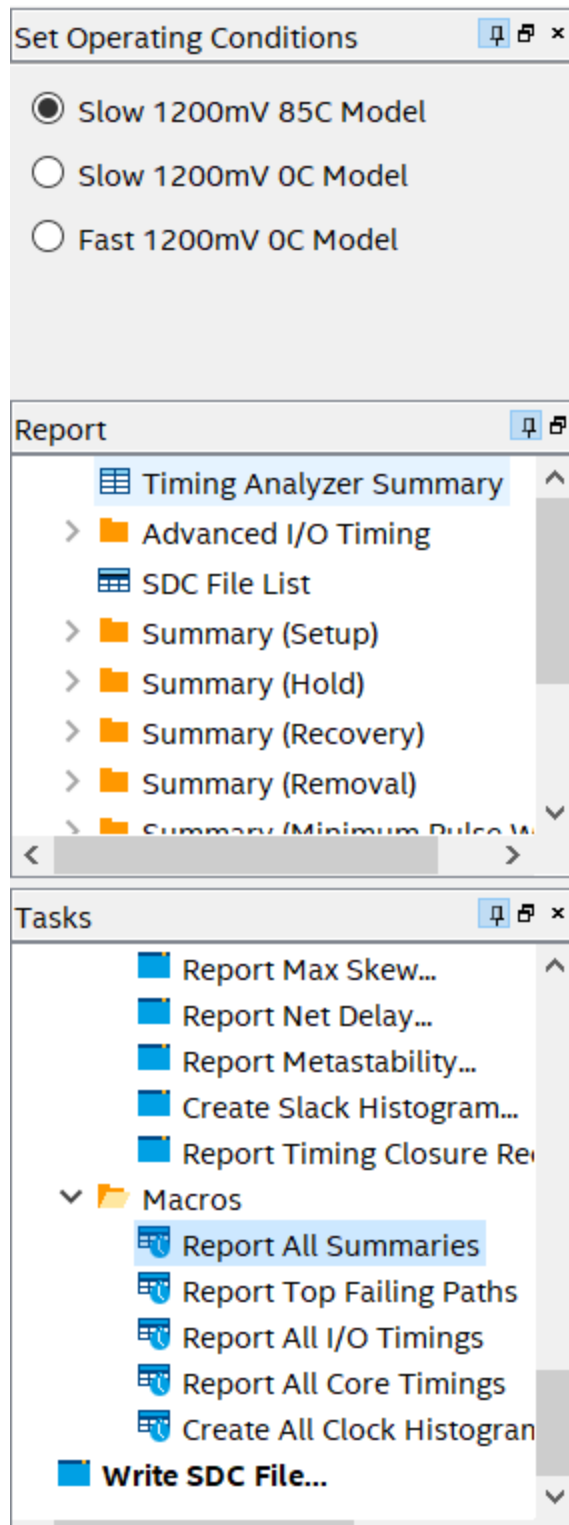


Figure 15

15. Click “Report All Summary” to generate the reports. All the timing report is presented in “Report” window as shown in figure 16. The most important reports are Setup and Hold. The Maximum Frequency is calculated using the setup time.



16. Click “Report timing” from Custom Reports window. The choose “Clk” in “From clock” section. And then click “Report Timing” button. A detail analysis is presented for the critical path as shown in figure 15

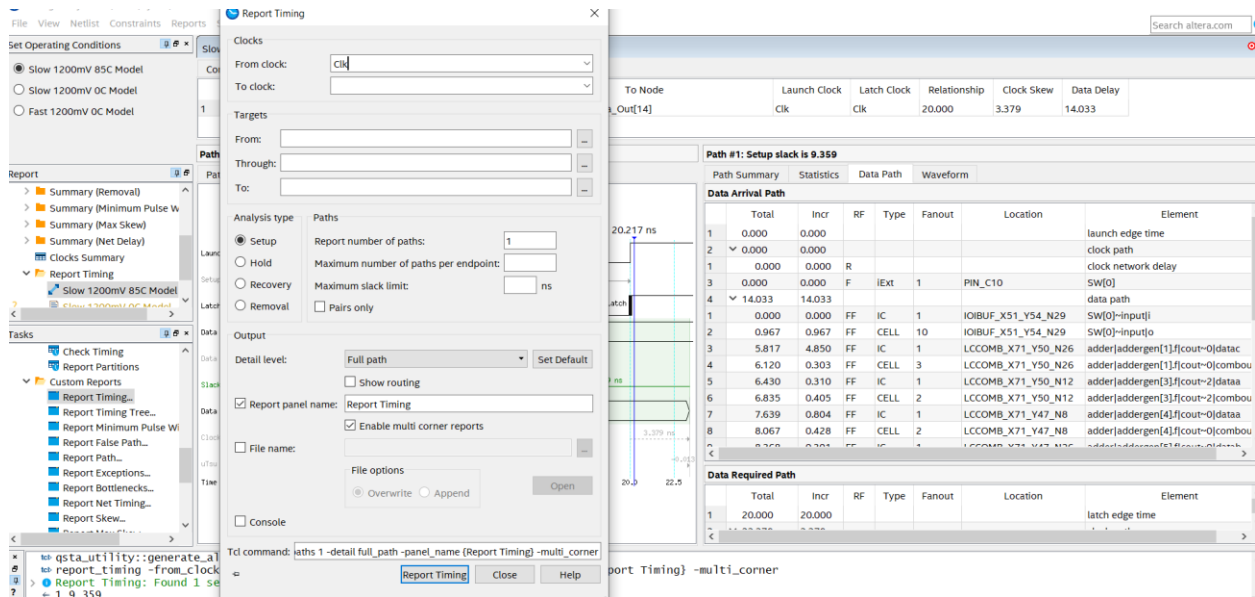
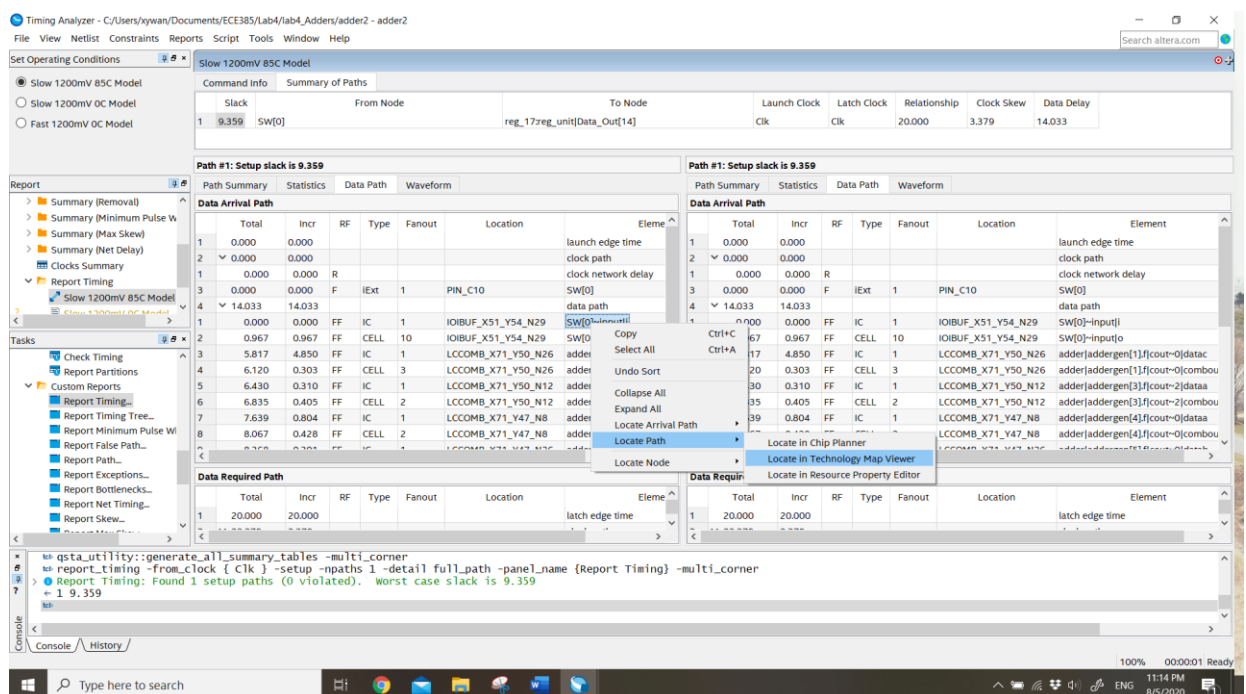


Figure 15

17. If you would like to see the path diagram, right click the data path cell as shown in figure 16.



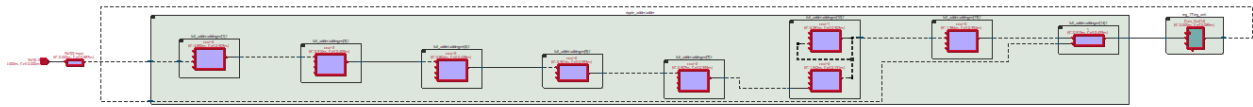


Figure 16

18. For other adders such as Carry Look Ahead and Carry select adder, uncomment the specific adder in the top level. Conduct a full-compilation and then repeat from step 14. Since the I/O paths for the three adders are the same, there is no need to modify the sdc file.

For some of the adders, you may notice that the critical path is nonsense. Namely, it starts at a switch SW[x] goes through a combinational HEX driver, and terminates in a LED HEX[x]. The reason is that although this path does not require clock/register at all (it starts at an I/O pin, goes through a combinational module, and terminates at an I/O pin), the limited I/O speed of the FPGA pins (~150 MHz) becomes the limiting factor in your design, especially since the design is quite small/trivial. You have two choices to solve this:

1. You may calculate the 50 slowest paths rather than just the slowest path, and find the slowest path which does not involve I/O. You can do this by going to step 16 and modifying where it says "Report number of paths" from 1 to 50. This will give you the 50 slowest paths, and then you can manually find the slowest path which does not involve starting at SW[x] and ending at HEX[x]. Attach this path to your lab report and use this path as the critical path for your explanation.
2. Add 'false paths' which start from SW[x] and end at HEX[x]. How to do this is left as an exercise to the student, but this excludes all the 'false paths' from the timing calculation. As a hint, your false path 'from' should involve the switches, your false path 'to' should involve the HEX displays.

Note that this is the reason why some students have fMax values that are very close to each other, it is limited by the I/O port speed, rather than the adder itself. Namely, method 2 is a better solution for solving this problem, as it also improves the quality of your fMax data.