CSCE 337: Digital Design II

Projects 2 (Static Timing Analysis) Report

Amr Galal - 900140044

Kareem Farid - 900141695

Omar Nawawy - 900142042

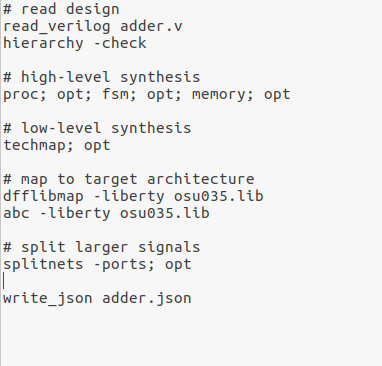
Sherif Hussein - 900140265

# Introduction:

This document is to explain our static timing analysis project. The project’s main objective is to analyze a given design, find all the timing paths in the design, get the critical path, check for timing violations and report the results, get the arrival, required time and the slack for all the gates of the design.

# Description:

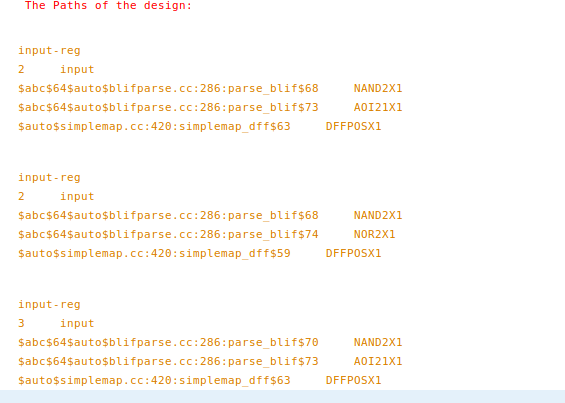
**1- Gate level netlist:**

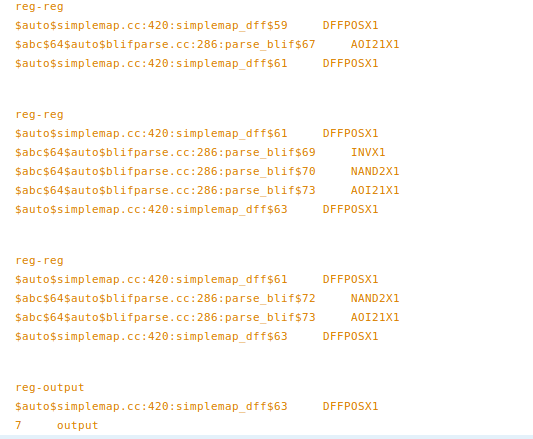
* Getting the gate level netlist of the design from yosys using the OSU library
* Writing the gate level netlist in a json file
* The verilog file we are using is changed in line 2 in script
* The json file that is used is generated in the last line of the script

Yosys Script (.ys)

**2- Creating a DAG from the gate level netlist and finding all paths:**

* Used graphlib library to create a graph object to represent the design and to be used in all the next requirements
* The graph takes all the inputs/outputs from the json file and add them as nodes in the graph and are identified by their unique name in the json file
* Then it manages to take the logical components of the circuit to be implemented as nodes and are identified by their unique name in the json
* Then the graph sets directed edges between the graph nodes by reading the connections between them in the json file
* Traversed the graph to get all the timing paths types (Input to reg, reg to reg, reg to output and input to output)
* Started the traversal from inputs/DFFs to DFFs/outputs then extracted each path and identified its type.

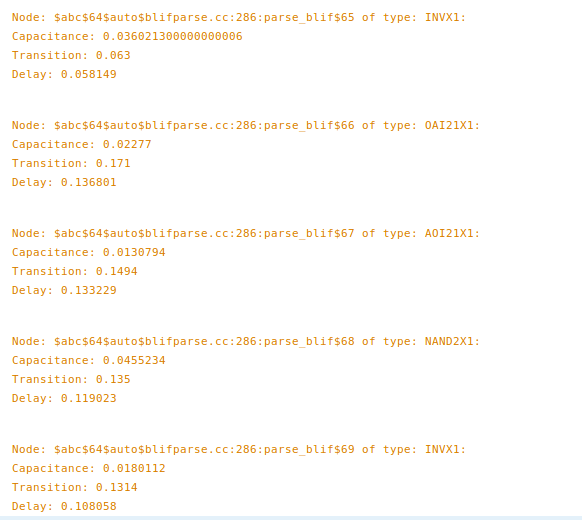


****

Sample paths output

**3- Reading the library file and getting the input transition and capacitance:**

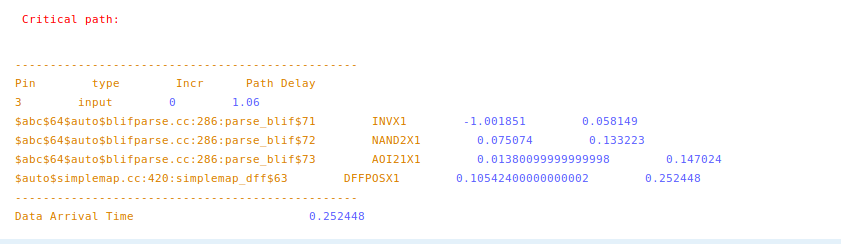
* Traversed the graphs and got the input transition for all the gates
* Calculated the output capacitance for all the gates by summing the capacitance of all the gates input pins connected to its output pin
* Assumed the net capacitance for the wire is a constant number 0.3
* Gate names, output capacitance, transition time and all info are stored in a “Mapp” 2d array.
* Calculated the delay for all the gates from the library file using the input transition time and the output capacitance as two indices to get the delay from the table using extrapolation.

****

Gates capacitance, transition and delay sample

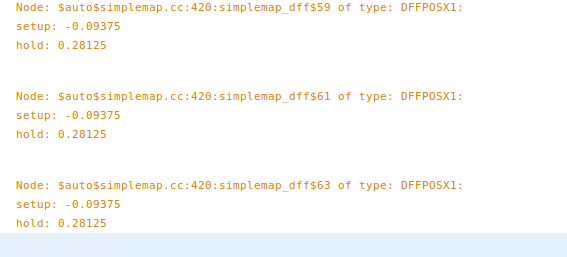
**4- Critical path:**

* Applied CPM method to check for the path of the highest delay

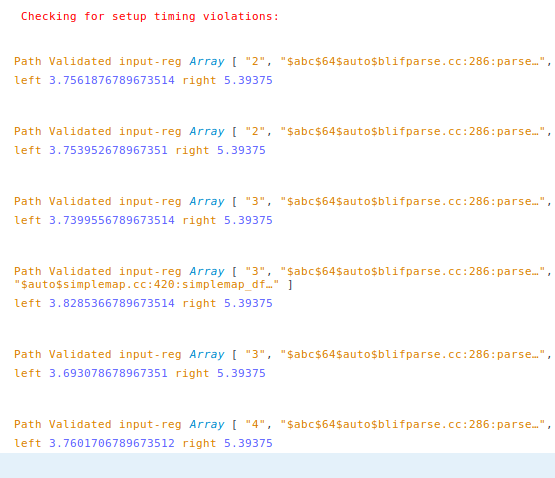


Critical path sample output

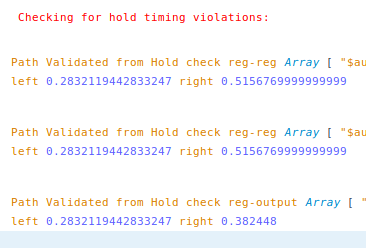
**5- Checking Timing Violations:**

* Calculated the propagation delay for each path
* Specified the clock period from the timing constraint file, as well as input delay and output delay
* Specified skew time by a random range from clock period
* Got the hold time and setup time for each flip flop from the library file
* Calculated contamination delay for each path (minimum delay)
* Checked all paths for setup violation and validated them according to the inequality T ck2q-max + T pd < T skew + T cycle - T setup
* If the path is reg to output, output delay is added to the right hand side
* If the path is input to reg, input delay is added to the left hand side
* Checked all paths for setup violation and validated them according to the inequality T ck2q-min + T cd < T skew + T hold

Flip Flop setup time and hold time



Setup violation check sample output



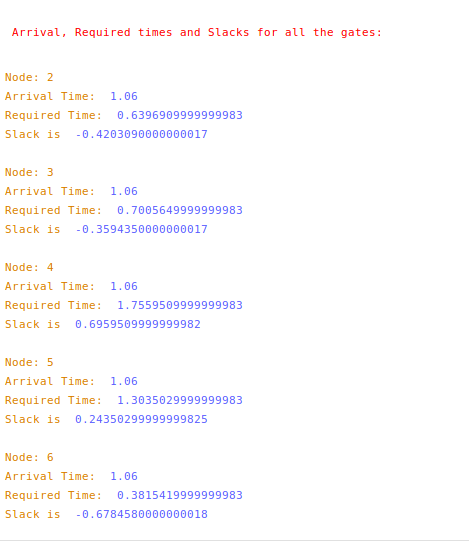
Time violation check sample output

**6- Arrival time, required time and slack of all gates:**

Required arrival time (RAT) is calculated by setting the last node in the graphs to tc (or tc-setup in sequential logic) and propagating backwards and picking the lowest required time possible.

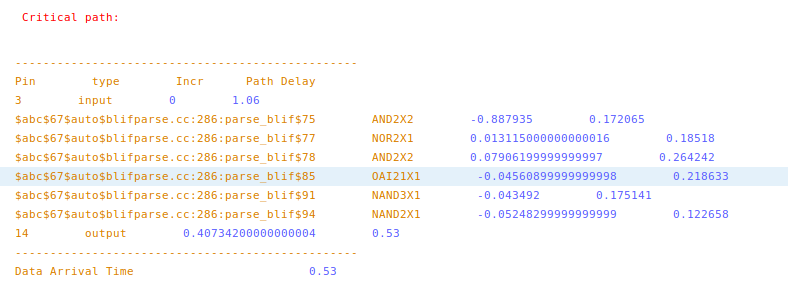
Actual arrival time (AAT) is calculated by starting from the input nodes and checking the delays and take the worst arrival time of a path to be saved in calculation of AAT of the next nodes

Slack is calculated by RAT - AAT for each node in the graphs

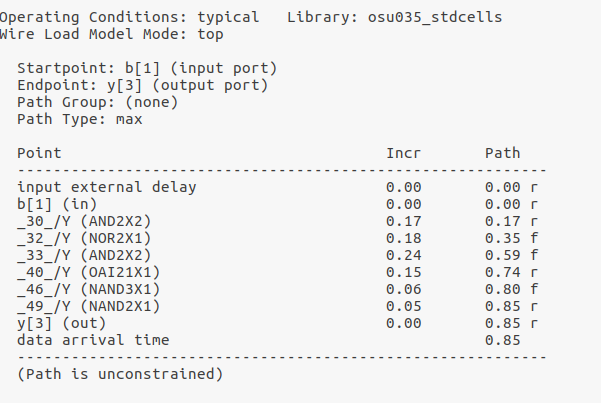
Arrival, Required time and Slacks sample output

# 

# Comparing Results

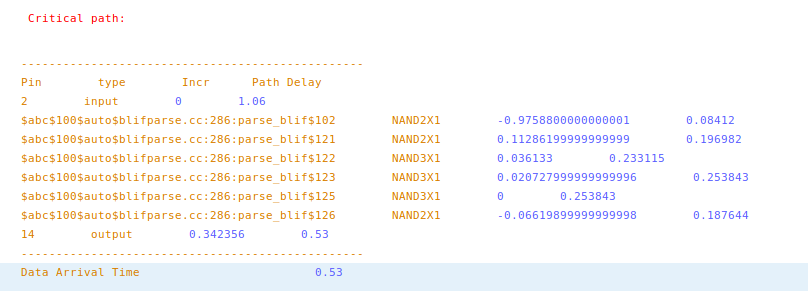
* Full Adder design:

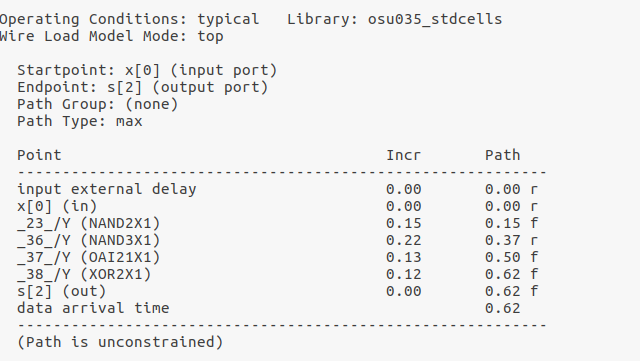
Critical path sample output



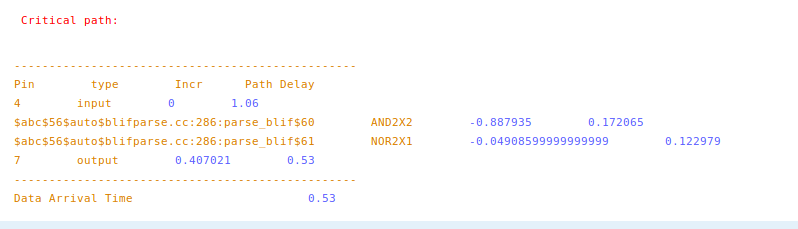
IC compiler critical path output

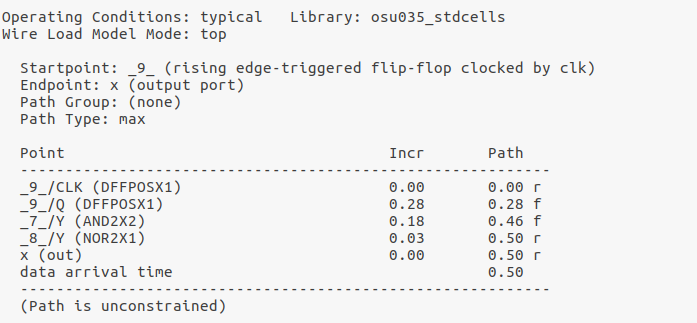
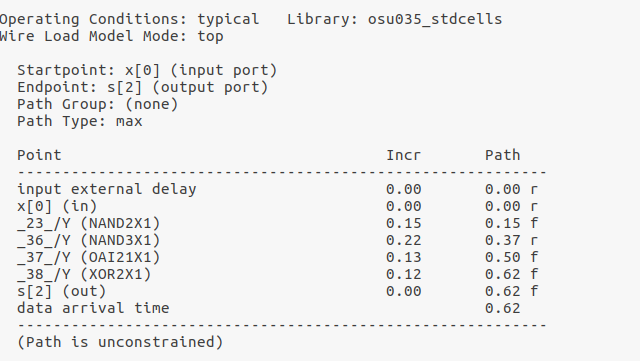
* Carry Skip Adder:

Critical path sample output 

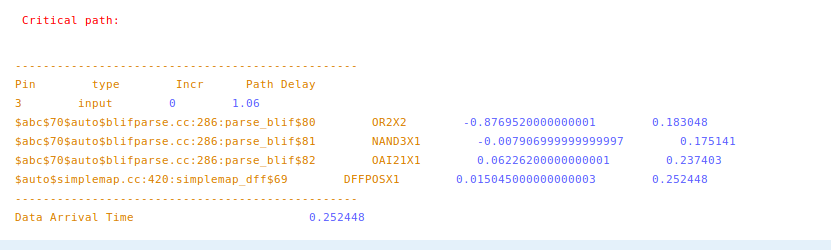
IC compiler critical path output

* Sequential Circuit 1:

Critical path sample output 

IC compiler critical path output

* Sequential Circuit 4:

Critical path sample output 

IC compiler critical path sample output