ASIC Homework – 2

Name : J. Prashanth

Roll number : IMT2019507

Note : All the .tcl, verilog, excel files and data folders can be found along with this pdf submission. Please refer to the file in case the screenshots are too small / not readable. All the script files are well commented, please go through them for code explanations. The questions are in no way altered to get a positive slack.

(A) sel\_sense :

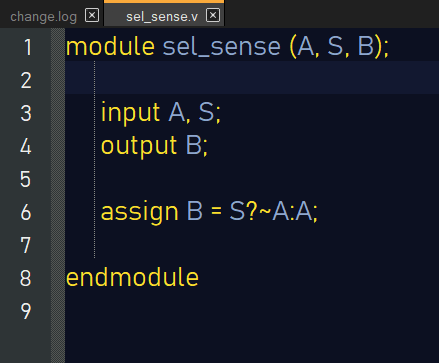
Description :

“sel\_sense” is a simple combinational circuit with two inputs and one output. A plain-English behavioral description of the logic is as follows:

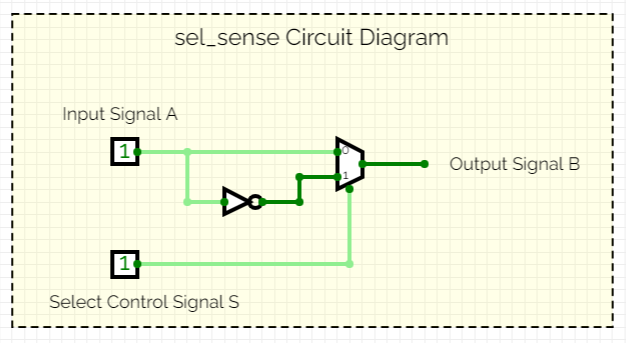
1. The circuit has one output, B, and two inputs, A and S

2. The output B is equal to input A if S=0 or is equal to the inverse of A if S = 1.

(1) Behavioral Verilog code for sel\_sense :



(2) Schematic of circuit that implements sel\_sense :

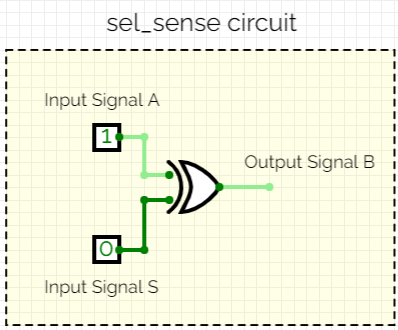
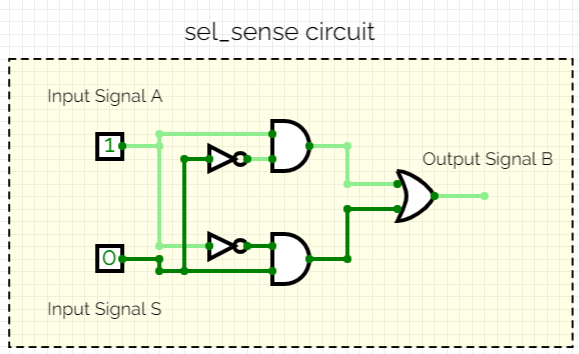


(3) Truth table of sel\_sense :

|  |  |  |
| --- | --- | --- |
| S | A | B |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

|  |  |  |
| --- | --- | --- |
| S \ A | 0 | 1 |
| 0 | 0 | 1 |
| 1 | 1 | 0 |

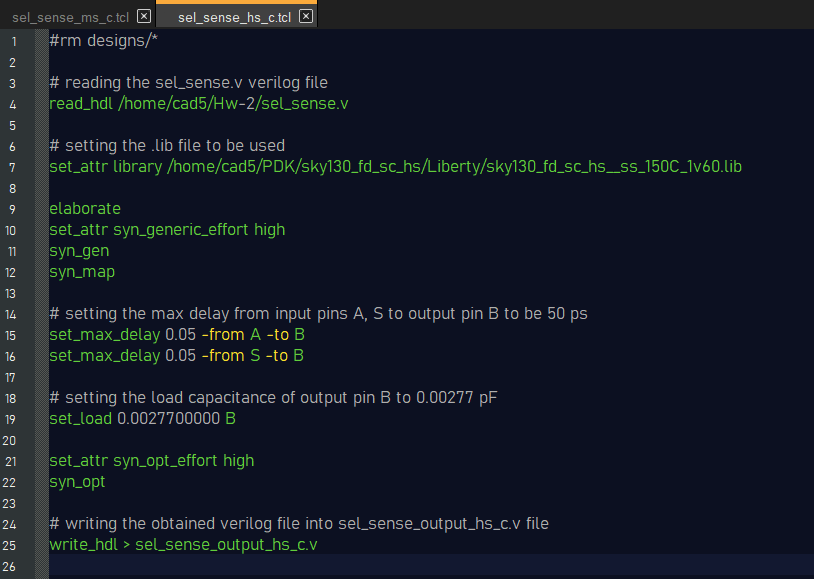
Making a K-map, we can see that, B = A(~S) + S(~A) = A ⊕ S,



( or )

(4) Synthesize sel\_sense using Skywater HS and MS libraries, netlist file results and synthesis scripts :

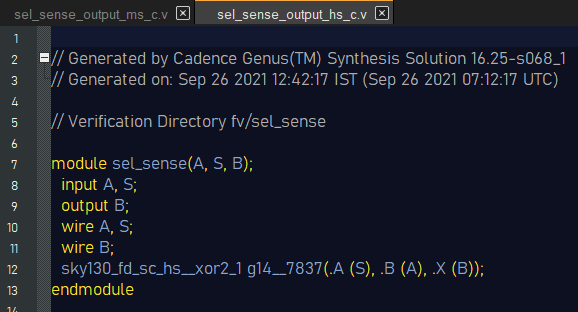
(ss, 150C, 1.6V - pvt)

The sel\_sense\_hs\_c.tcl file : (Synthesis file for HS library)

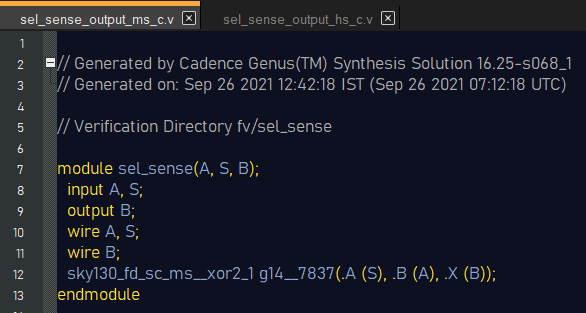
The sel\_sense\_ms\_c.tcl file : (Synthesis file for MS library)



The sel\_sense\_output\_hs\_c.v file : (The output HDL file for HS lib)



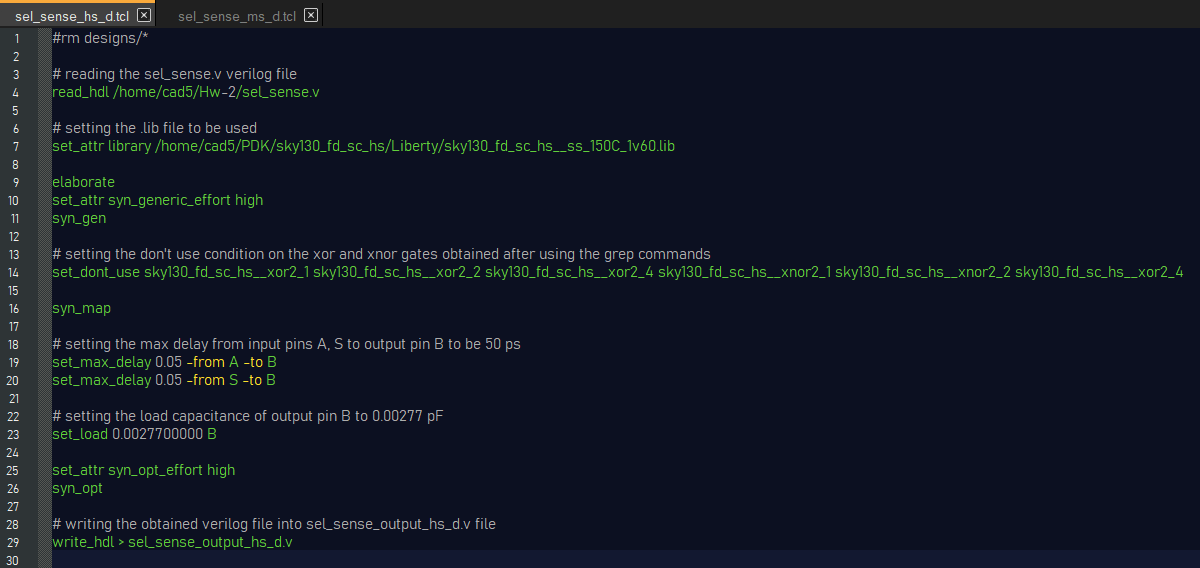
The sel\_sense\_output\_ms\_c.v file : (The output HDL file for MS lib)



(5) Synthesize sel\_sense using Skywater HS and MS libraries, netlist file results and synthesis scripts (without using XOR and XNOR gates) :

(ss, 150C, 1.6V - pvt)

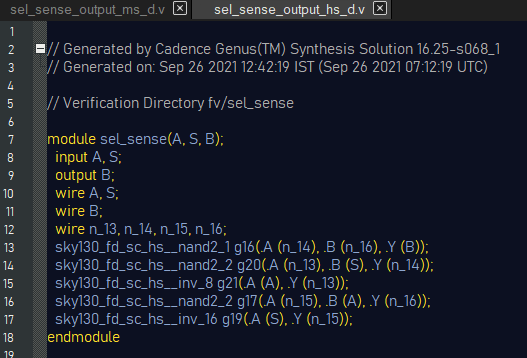
The sel\_sense\_hs\_d.tcl file : (Synthesis file for HS library)



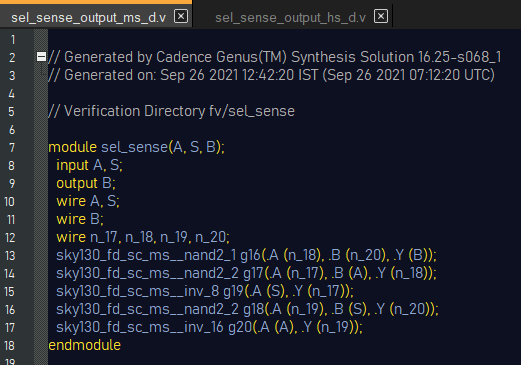
The sel\_sense\_ms\_d.tcl file : (Synthesis file for MS library)



The sel\_sense\_output\_hs\_d.v file : (The output HDL file for HS lib)

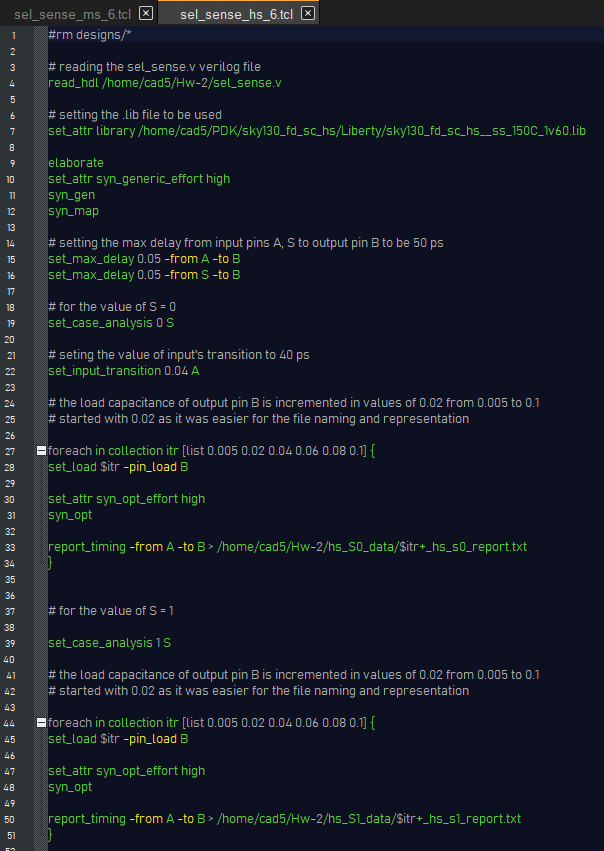


The sel\_sense\_output\_ms\_d.v file : (The output HDL file for MS lib)

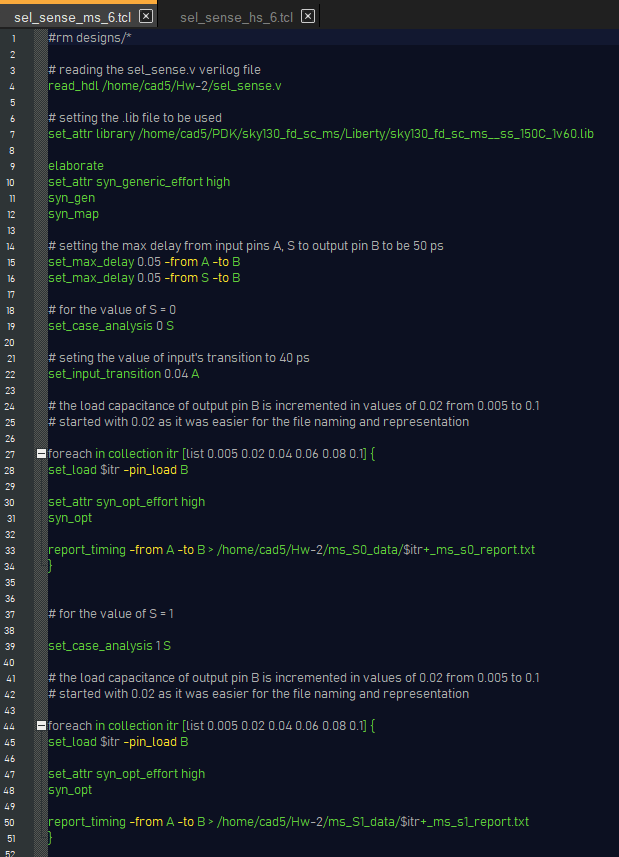


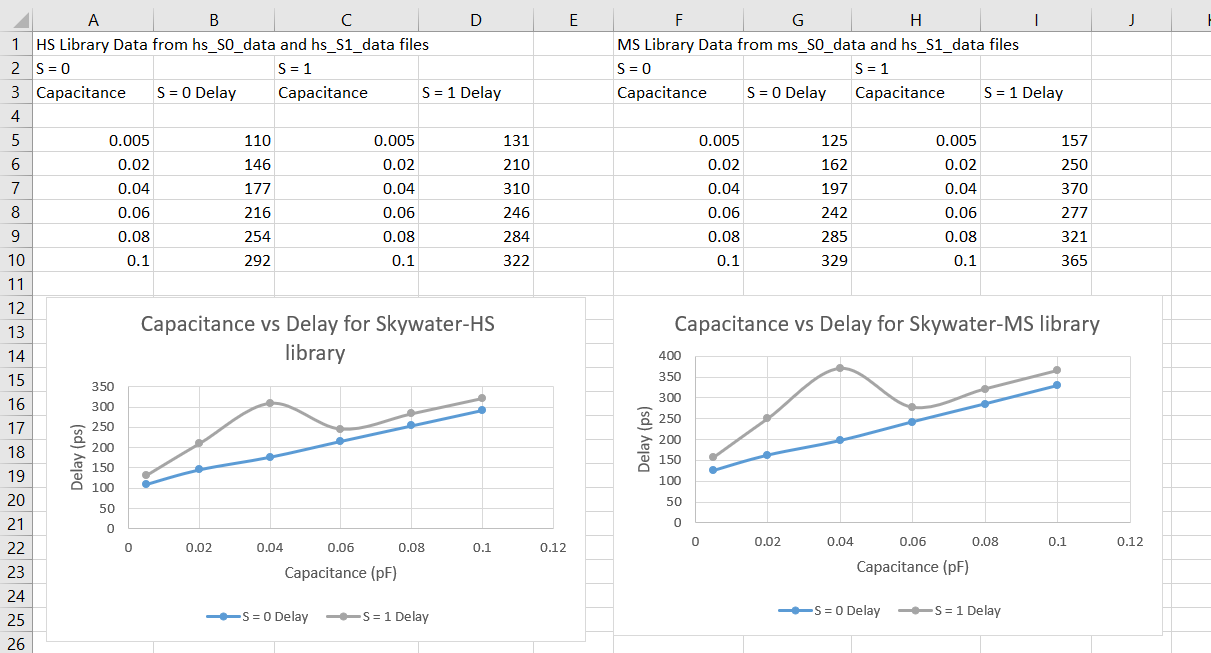
(6) For this question, first we write a synthesis file for each HS and MS libraries, the tcl files written for those libraries are “sel\_sense\_hs\_6.tcl” and “sel\_sense\_ms\_6.tcl” respectively. All the data acquired from the tcl scripts are in the “hs\_S0\_data”, “hs\_S1\_data”, “ms\_S0\_data” and “ms\_S1\_data” folders. I have gathered the data and put them in an excel sheet “Graph\_data.xlsx”. The screenshots of the graphs and data from the excel sheet are also given below.

Synthesis file used for getting data for HS library :



Synthesis file used for getting data for MS library :



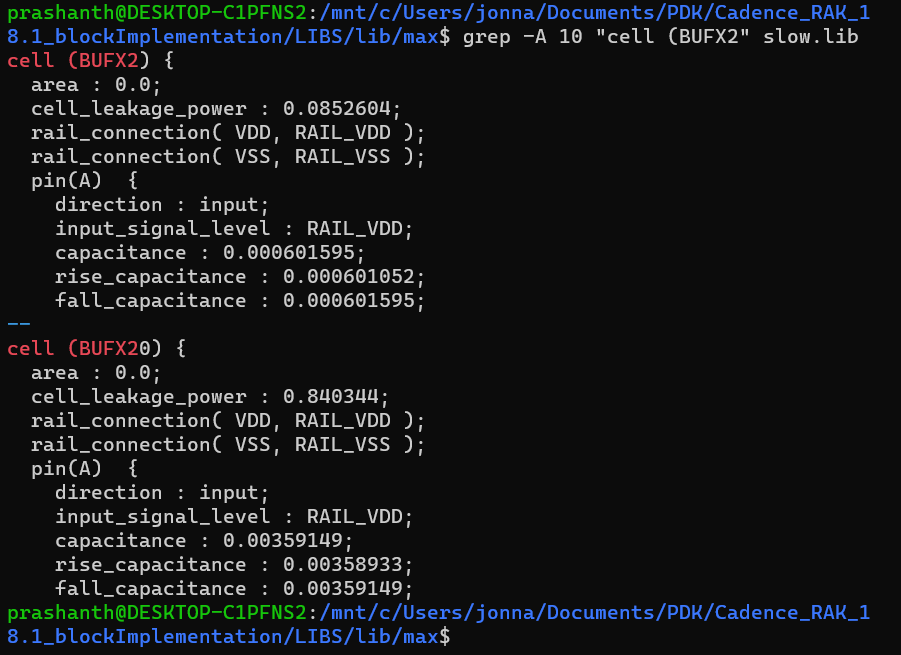


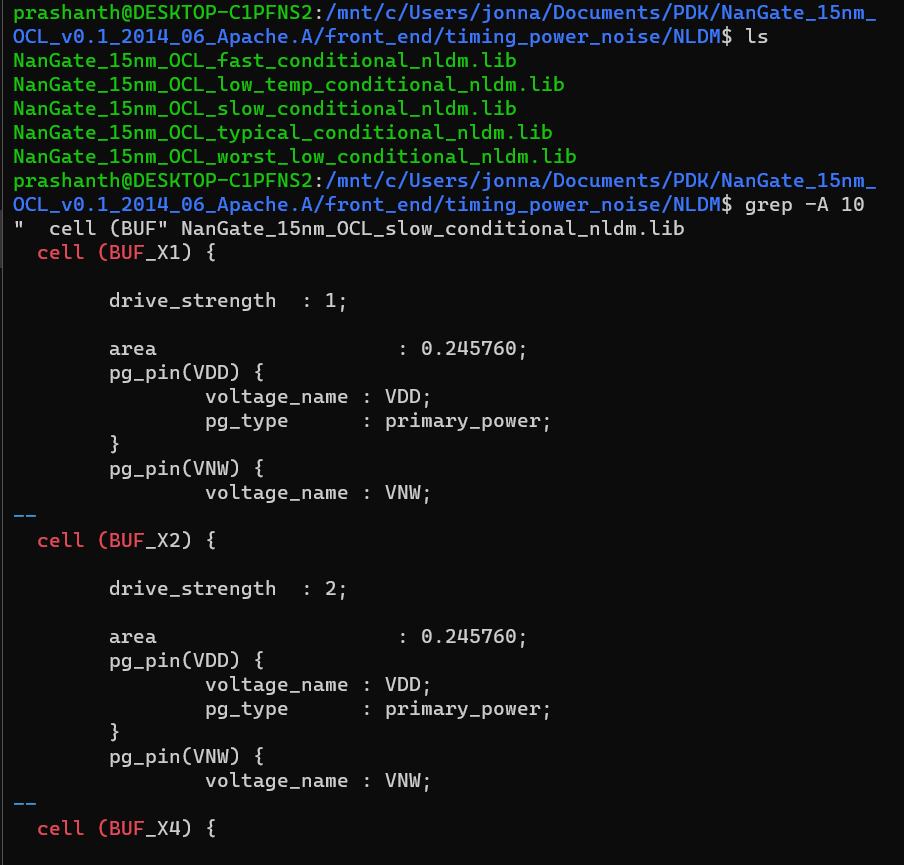
(B) Buffers in Standard Cell Library :

|  |  |  |
| --- | --- | --- |
| Cell Name | Name of Input Pin | Capacitance of Input Pin (pF) |
| BUFX2 (from Cadence RAK, /max/slow.lib) | A | 0.000601595 |
| BUF\_X2 (from Nangate\_15nm\_OCL, NanGate\_15nm\_OCL\_slow\_conditional\_nldm.lib) | I | 0.000839939 |
| BUF\_X2 (from Nangate\_15nm\_OCL, NanGate\_15nm\_OCL\_slow\_conditional\_ecsm.lib) | I | 0.000839939 |
| BUF\_X2 (from Nangate\_15nm\_OCL, NanGate\_15nm\_OCL\_slow\_conditional\_ccs.lib) | I | 0.000839939 |

Screenshots :

First, we identify the naming in the .lib files, after trial and error we find out that buffer files of strength “n” are named like BUF\_X{n}. So now we use the grep command to find out the capacitances.

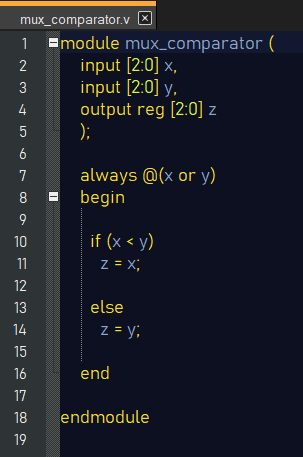




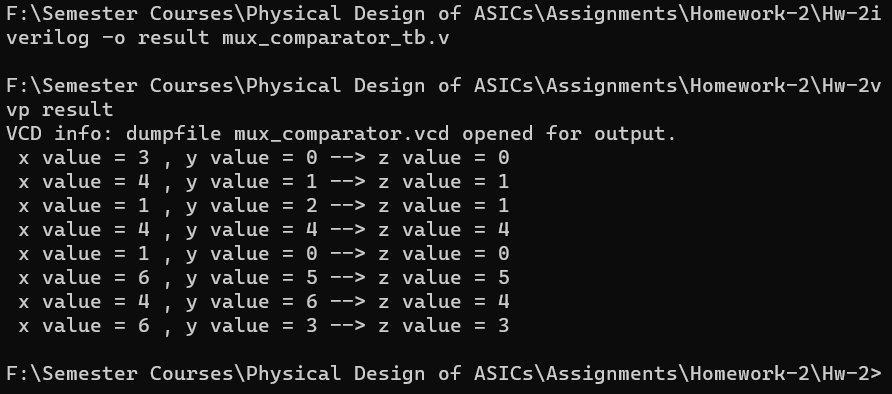


(C) mux\_comparator :

(1) The verilog code for mux\_comparator is given below :

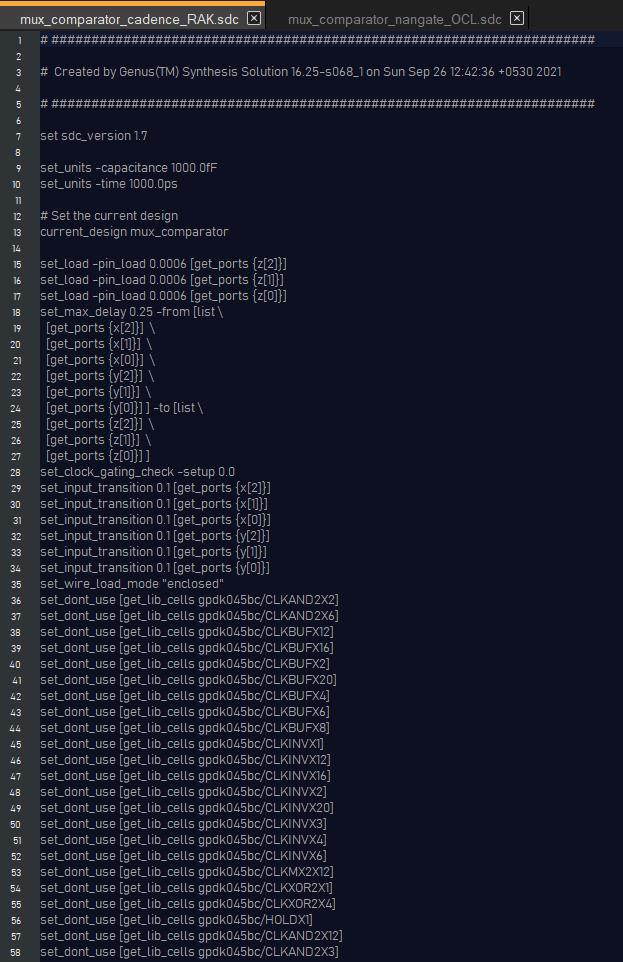


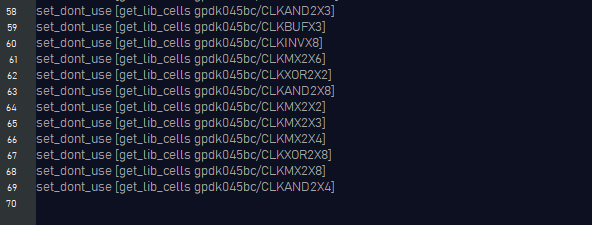
I have written the test bench for the above verilog file and obtained the following results to validate the working of the mux\_comparator by taking few random cases, (testbench file : ”mux\_comparator\_tb.v”)



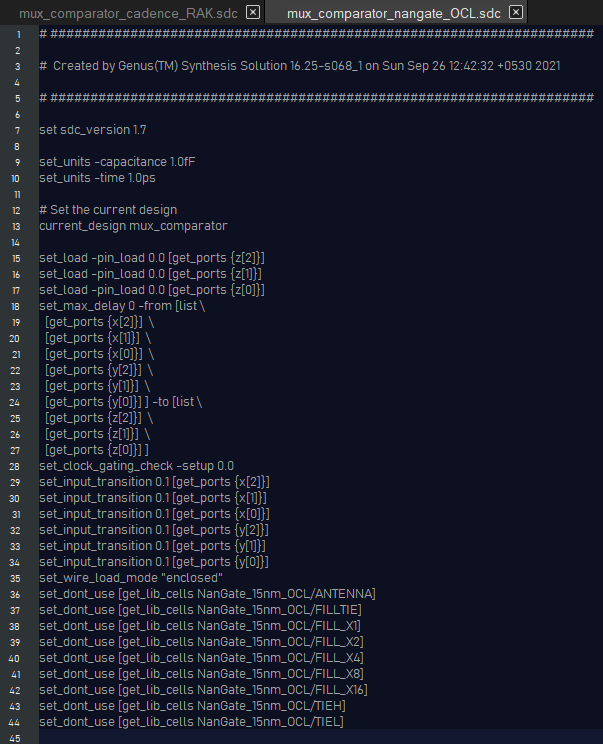
(2) .sdc files for,

Cadence RAK lib : (file : “mux\_comparator\_cadence\_RAK.sdc”)



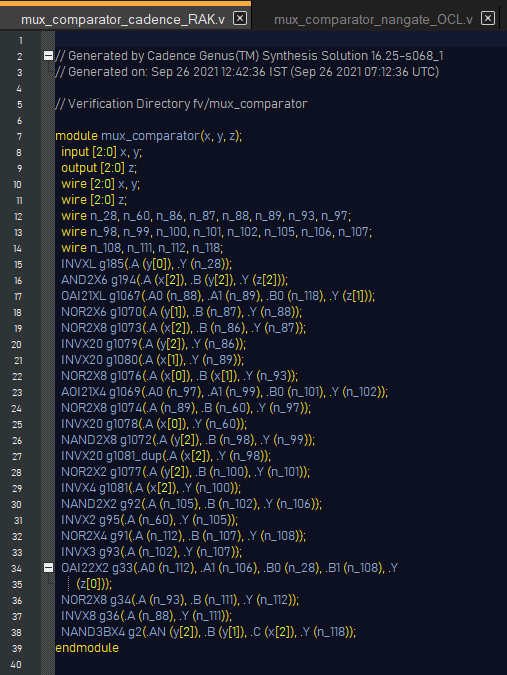


For Nangate OCL lib : (file : “mux\_comparator\_nangate\_OCL.sdc”)

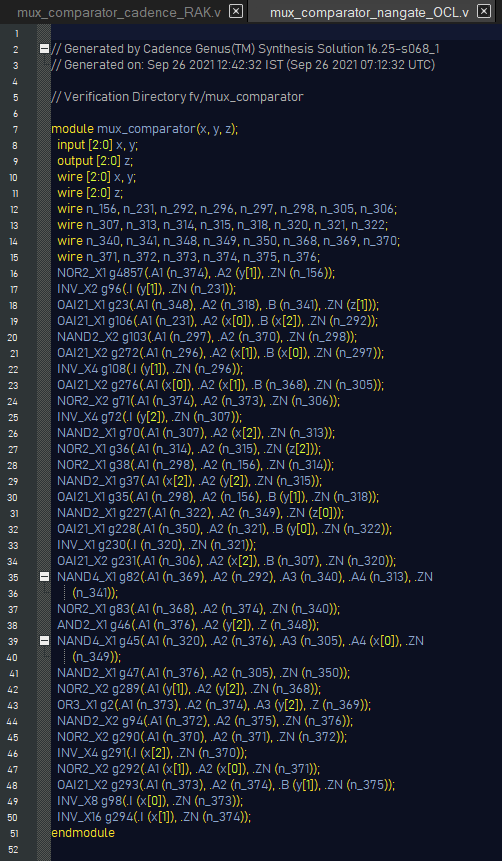


(3) The resulting netlists are,

For Cadence RAK library, (file : mux\_comparator\_cadence\_RAK.v)



For Nangate OCL library, (file : mux\_comparator\_nangate\_OCL.v)



For part (4) and (5), refer to,

“mux\_comparator\_RAK\_timing\_report.txt” file for Cadence RAK library

“mux\_comparator\_OCL\_timing\_report.txt” file for Nangate OCL library

(The script files used for generating these are “mux\_comparator\_RAK.tcl” and “mux\_comparator\_OCL.tcl”)

I haven’t pasted their screenshots here as there is a lot of data in those files and it would take 4-5 pages for a single library’s timing report file.

What it contains :

It contains timing reports for all the possible paths, (ie, x[0] -> z[0], x[0] -> z[2], y[2] -> z[1], etc.). By looking at each of the timing delays and other data, we can determine which of the paths have the maximum delay (critical path) and the minimum delay, and can hence list out the cells in those paths.

(4) The critical paths are the first generated timing reports in each of the above mentioned .txt files as “report\_timing” in general gives max\_delay (critical) path.

For Nangate OCL : max delay = 14 ps

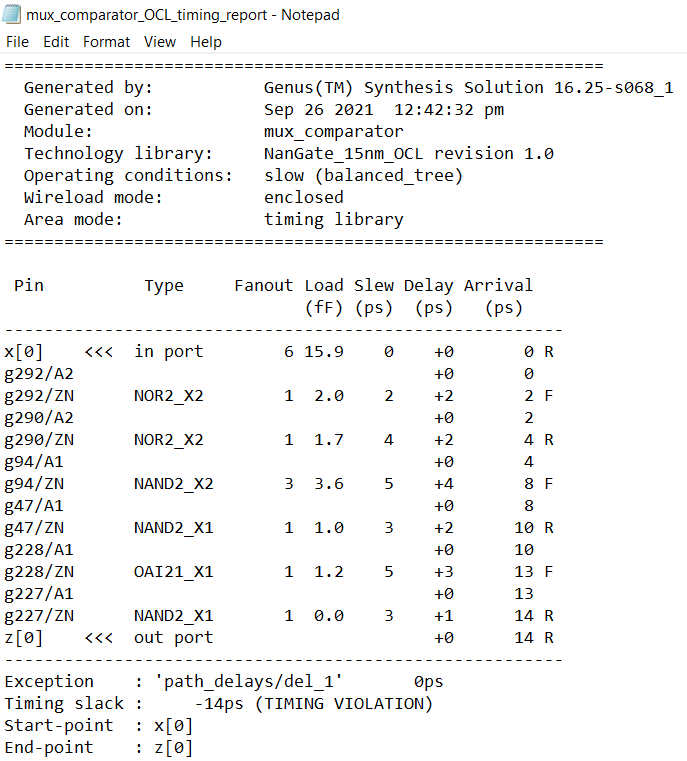
Number of cells = 6 cells

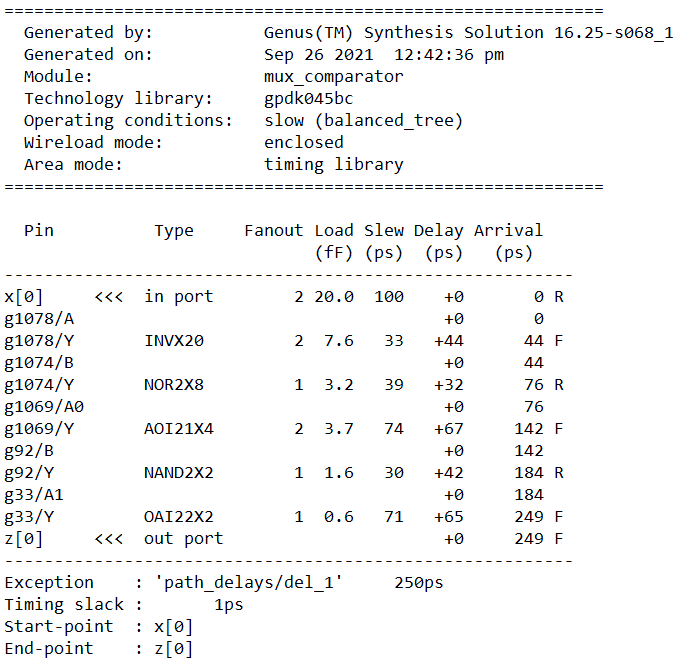
For Cadence RAK : max delay = 249 ps

Number of cells = 5 cells

(the screenshots below show the timings and cells in the critical path)

For Nangate OCL library,

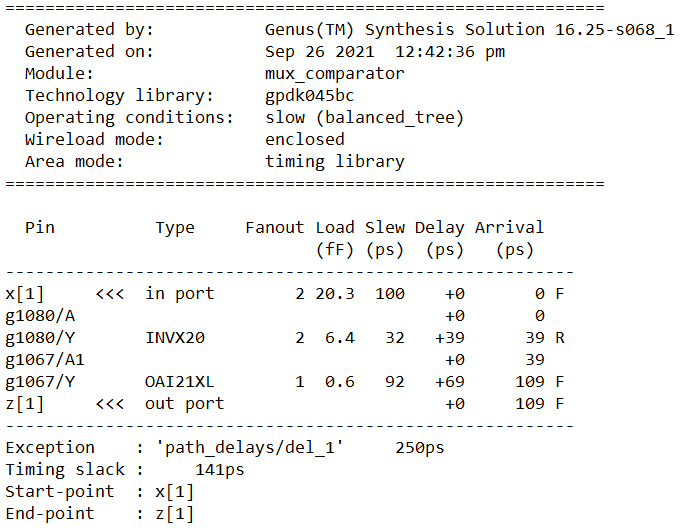


For Cadence RAK library,

(5) For the min delay paths,

|  |  |
| --- | --- |
| For Cadence RAK, input -> output | Timing delay (in pico seconds) |
| x[0] –> z[0] | 249 ps |
| x[1] –> z[0] | 256 ps |
| x[1] –> z[1] | 109 ps |
| x[2] –> z[0] | 256 ps |
| x[2] –> z[1] | 189 ps |
| x[2] –> z[2] | 128 ps |
| y[0] –> z[0] | 131 ps |
| y[1] –> z[0] | 217 ps |
| y[1] –> z[1] | 161 ps |
| y[2] –> z[0] | 254 ps |
| y[2] –> z[1] | 199 ps |
| y[2] –> z[2] | 122 ps |

The remaining paths do not exist in the circuit. The min delay path is from x[1] -> z[1] as per the table. The number of standard cells are 2.



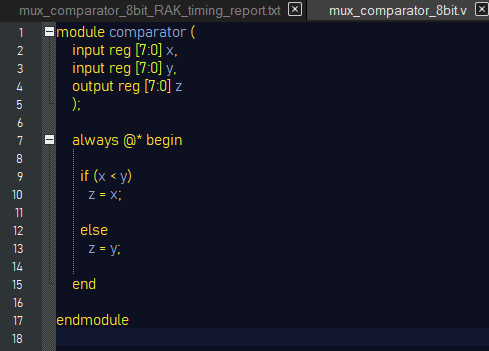
|  |  |
| --- | --- |
| For Nangate OCL, input -> output | Timing delay (in pico seconds) |
| x[0] –> z[0] | 14 ps |
| x[0] -> z[1] | 13 ps |
| x[0] -> z[2] | 9 ps |
| x[1] –> z[0] | 14 ps |
| x[1] –> z[1] | 13 ps |
| x[1] -> z[2] | 10 ps |
| x[2] –> z[0] | 14 ps |
| x[2] –> z[1] | 12 ps |
| x[2] –> z[2] | 8 ps |
| y[0] –> z[0] | 4 ps |
| y[1] –> z[0] | 14 ps |
| y[1] –> z[1] | 12 ps |
| y[1] –> z[2] | 10 ps |
| y[2] –> z[0] | 14 ps |
| y[2] –> z[1] | 12 ps |
| Y[2] -> z[2] | 4 ps |

The remaining paths do not exist in the circuit. There are two min delay paths,

y[0] -> z[0] and y[2] -> z[2],

both of which have 4 ps timing delay. There are 2 cells in both the paths.

(6) For the 8-bit mux\_comparator, the following Verilog file has been created, (file : “mux\_comparator\_8bit.v”)

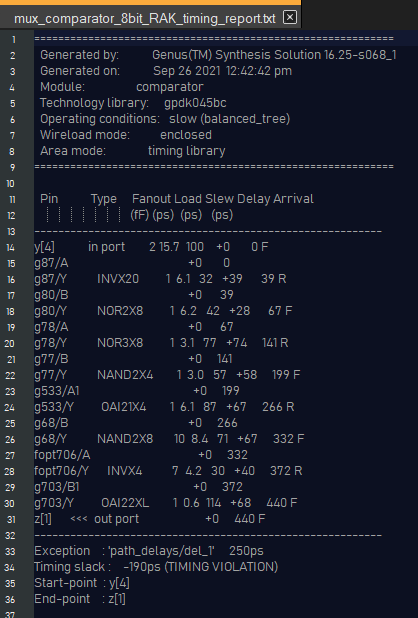


The Tcl file used to generate critical path info is, (file : “mux\_comparator\_8bit\_RAK.tcl”)



The generated timing report is,

(file : “mux\_comparator\_8bit\_RAK\_timing\_report.txt”)



So the path from y[4] to z[1] is the critical path and the cells used in that path are INVX20, NOR2X8, NOR3X8, NAND2X4, OAI21X4, NAND2X8, INVX4, OAI22XL. Number of cells used in critical path are 8.