Project Report

By: Matthew White, Chijioke Edeoga, Dayne Guy, Dung Tien Do

In this project we created a 5x5 multiplier. This design is made of 2 input shift registers, the multiplier, and 1 output parallel in serial out register. The way the design works is two inputs are fed into the 2 input registers serially, these inputs get turned into 5 inputs for each bit of X and Y. X and Y are fed through the combinational circuit that adds partial products and creates a 10 bit product. These bits get inputted into the output register. The output register inputs all 10 bits and outputs the final answer serially from one output made up of 10 bits.

Deep Dive:

Our Design 5x5 Multiplier is made into three main parts, the multiplier itself, 2 input SIPO Shift Registers, and an output PISO Shift Register. The Multiplier is the part that does the multiplying of the two numbers that are inputted into this system. The multiplier is made up of Full Adder and AND gates. This takes 10 inputs, half of them represent the first number you are multiplying, and the second half is the other number. When you follow the layout of the multiplier you will get the 10 outputs, each representing the digits of the final answer.

Next, the SIPO Shift Registers is the part that takes one of the numbers being multiplied as one serial input and turns it into 5 outputs. This register is made up of one AND gate and four D Flip-Flops. This register will output 5 values, representing each digit of the input serial number. These registers rely on a clock, so we AND the clock with an enable to activate and stop the D Flip Flops. Both of the SIPO register outputs will be inputted into the multiplier, just have to make sure that each output is inputted into the correct inputs because this can affect the result of the multiplier.

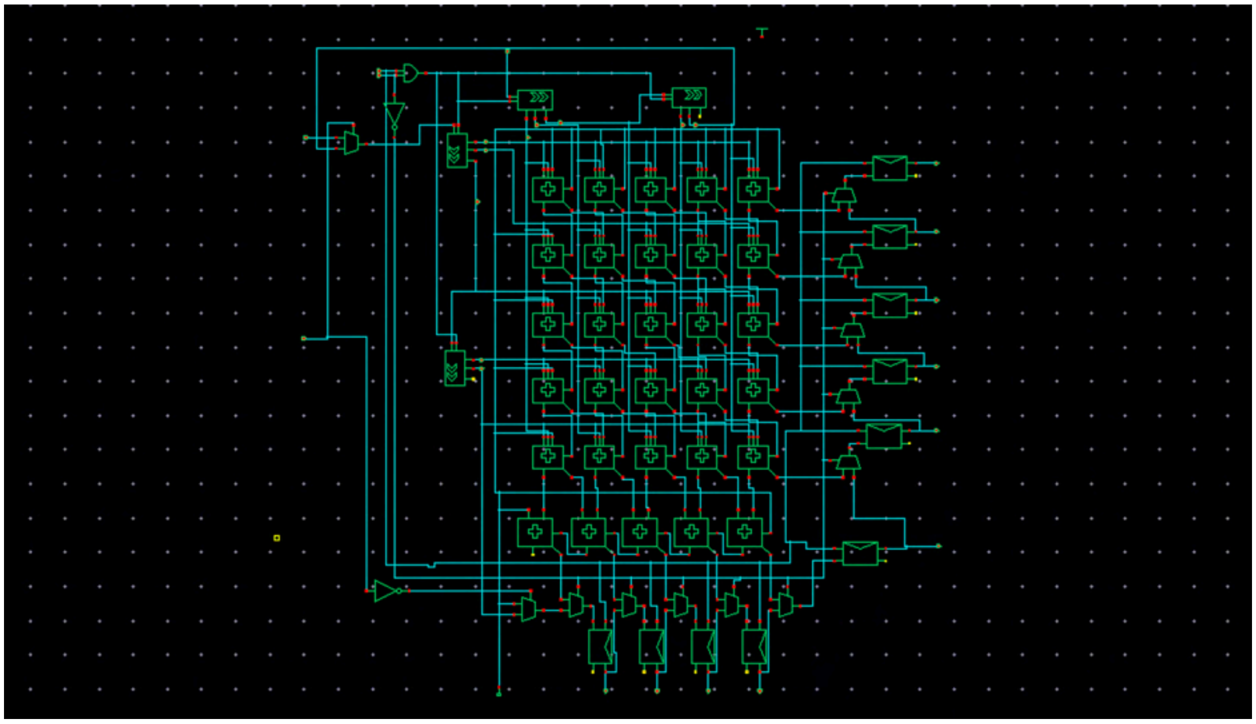
Finally, the PISO Shift Register takes the 10 outputs of the multiplier, each representing each digit of the final answer. The register takes these numbers and outputs them serially for the final output of this design. The register is made up of 8 2:1 MUX and 8 D Flip-Flops. A shift input for each MUX is necessary so that the D Flip-Flops can initially take in the inputs and then start to shift these numbers to the D Flip-Flop that is next to it and not take the inputs again. This design two numbers that you want to multiply can be inputted serially to two separate registers and the final answer be outputted from one register serially.

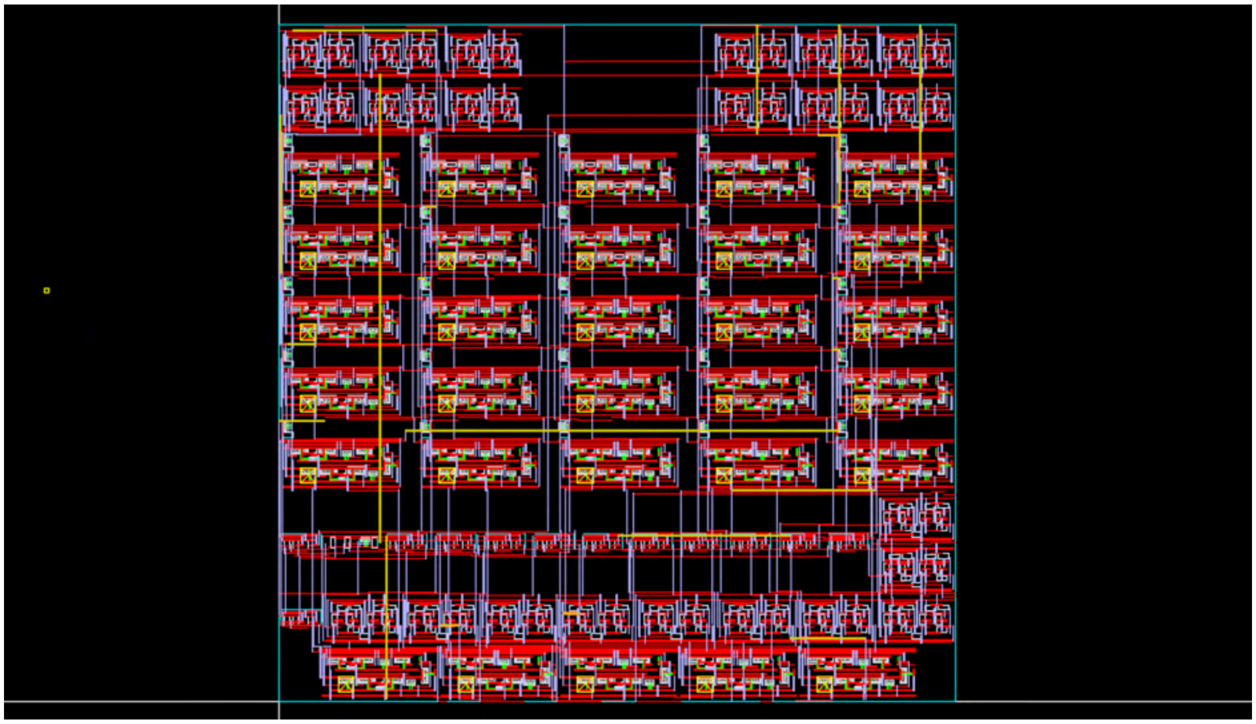
TEST Line - When on the test line activates a scan chain that allows input from Y to cascade to X and then to the output PISO register. This allows a single stream of output that includes both the output followed by the inputs X then Y.Of note our outputs are read MSB -First meaning that the highest place value will exit P0 first.

There are two main clocking elements:
CLK: A clock synchronizes all of our elements.
Load - Initially high to load the X and Y registers then shifts lower to begin shifting out the Multiplicand serially

The inputs are sensitive to both CLK and Load meaning that while LOAD is off, the input registers retain their values. The outputs are sensitive to Load in which load determines if the output registers are calculating a value or shifting out the output.

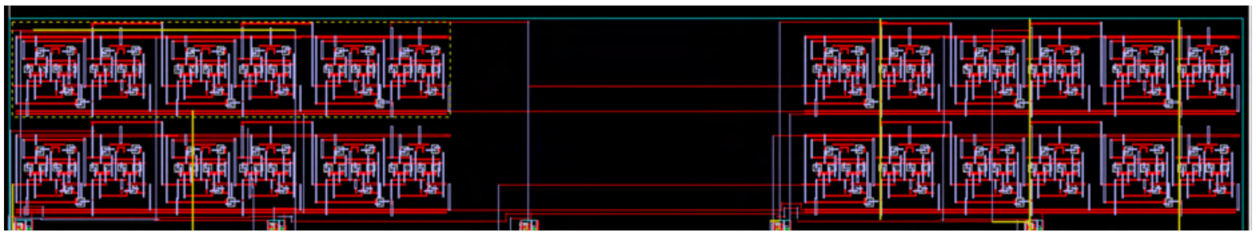
Design:



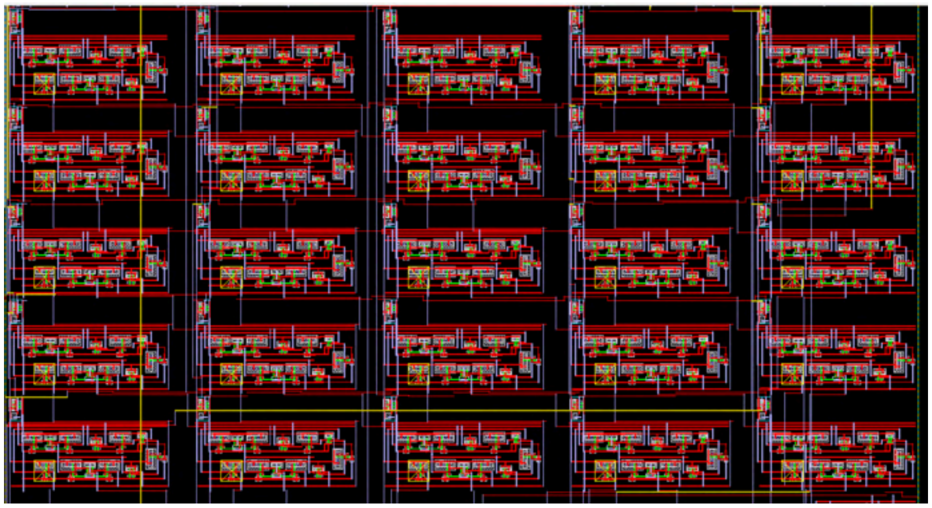


Video Link: https://youtu.be/Seni0iZFudQ

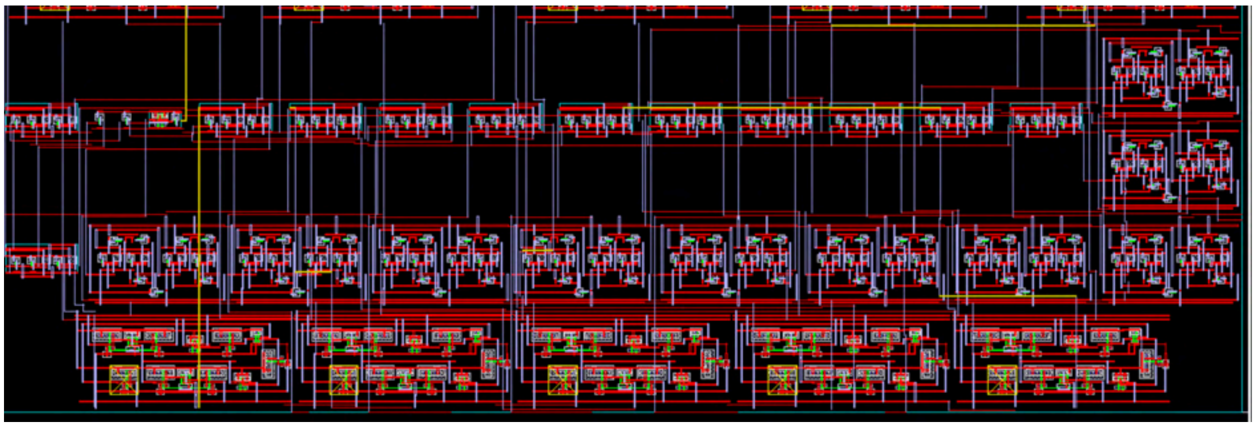
Input Register:



Multiplier:



Output Register:

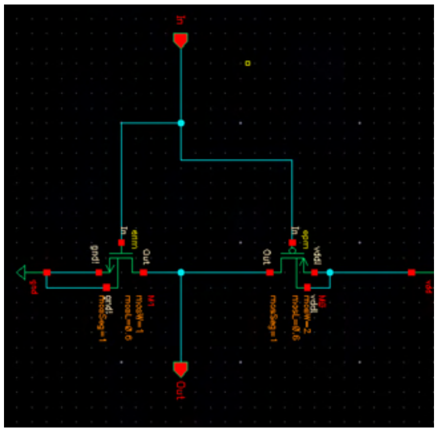


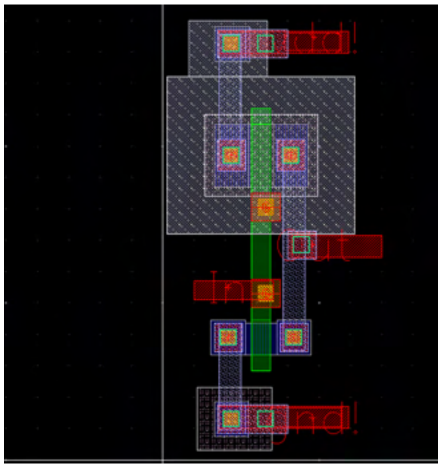
Design Decision:

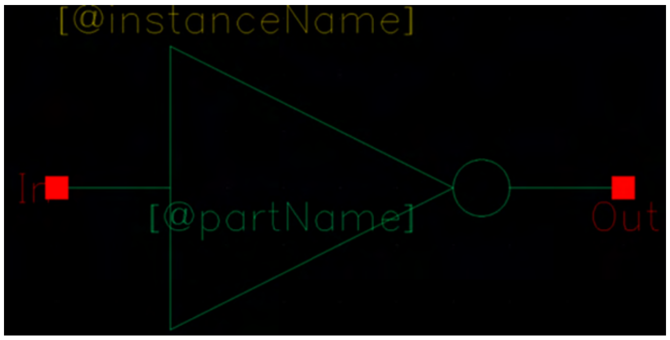
Deciding N: When designing our multiplier, we initially tried a 8x8 design. But when estimating our size of the multiplier, it would have been too large for the 900μm x 900μm area limit. We kept decreasing N until we found the right size for the 900µm x 900µm area limit, which ended up fitting perfectly for the 5x5.

Bit-slices:

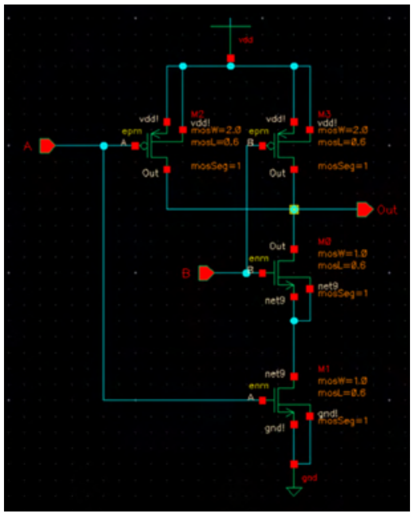
1. Inverter

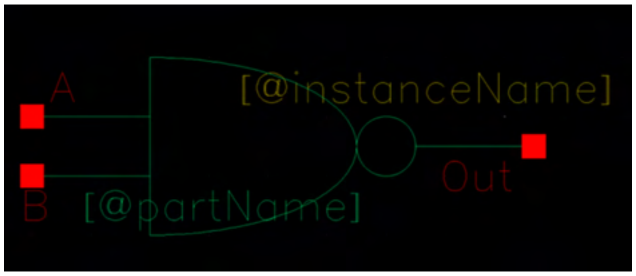


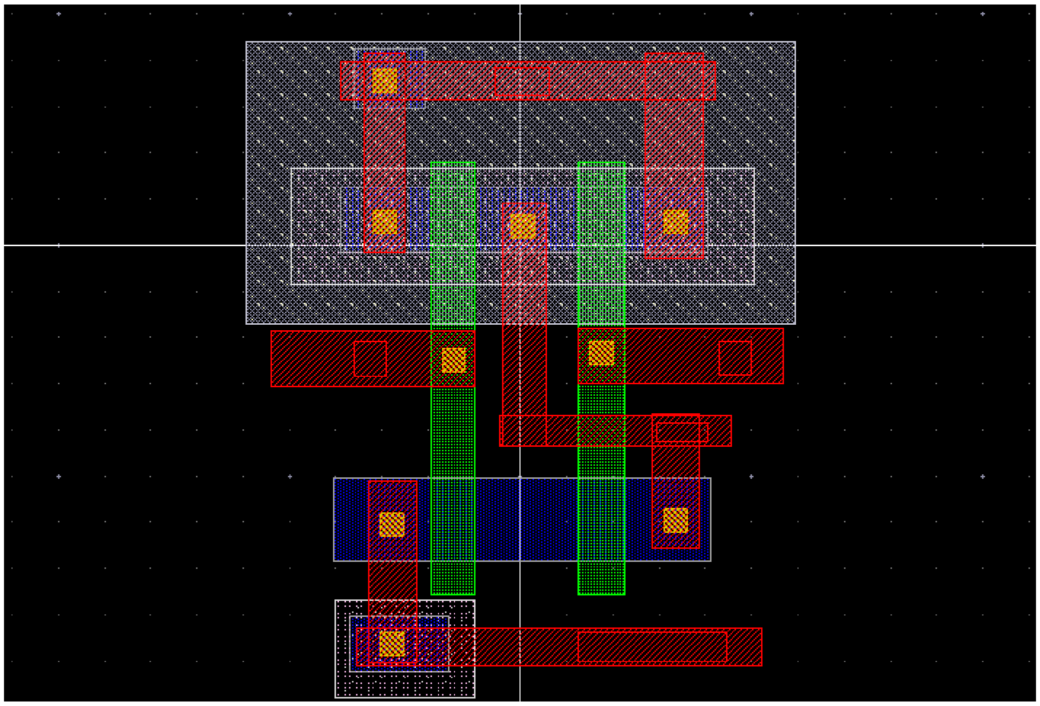




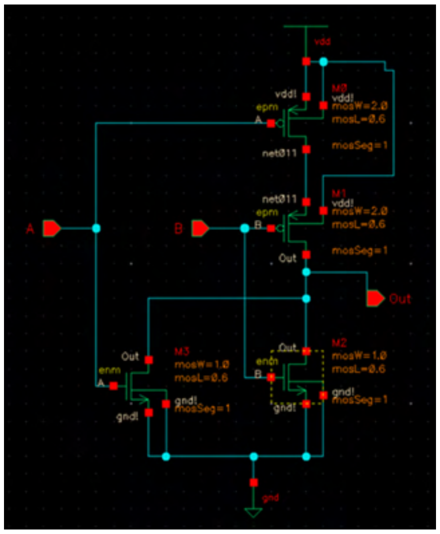
2. NAND

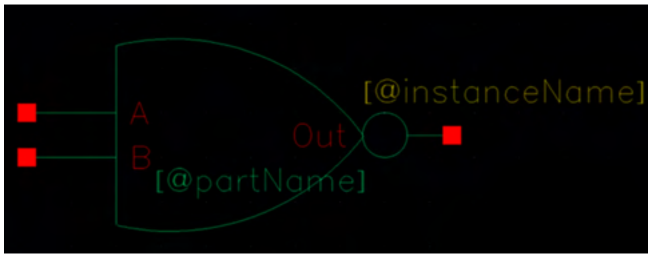

vdd
A
B
Out
M0
M3
M2
M1
vdd!
mosW=2.0
mosL=0.6
mosSeg=1
net9
mosW=1.0
gnd!
gnd

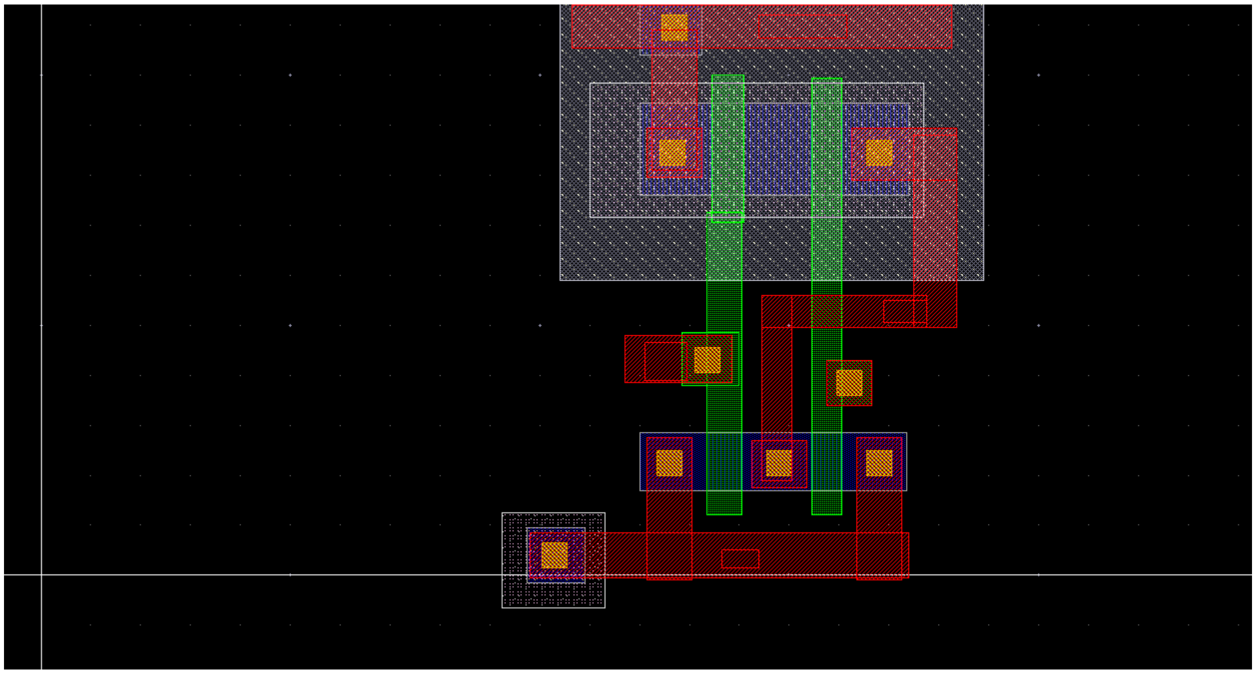

A
B
[@instanceName]
[@partName]
Out



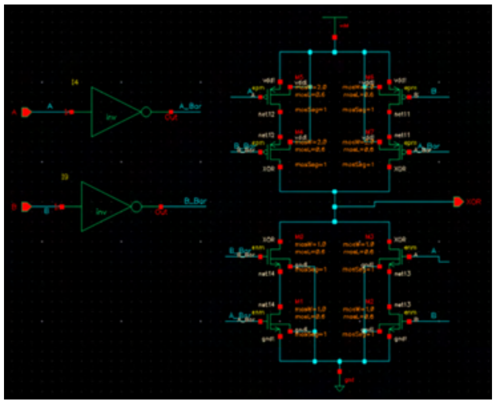
3. NOR

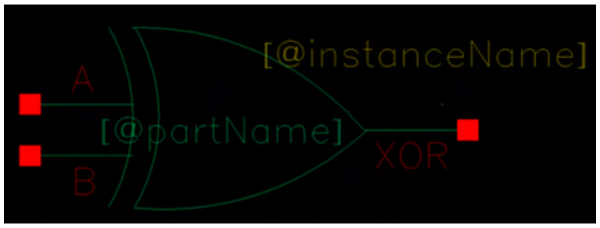

vdd
M0
vdd!
mosW=2.0
mosL=0.6
mosSeg=1
epm
A
net011
M1
B
Out
M3
mosW=1.0
enm
gnd!
M2
gnd

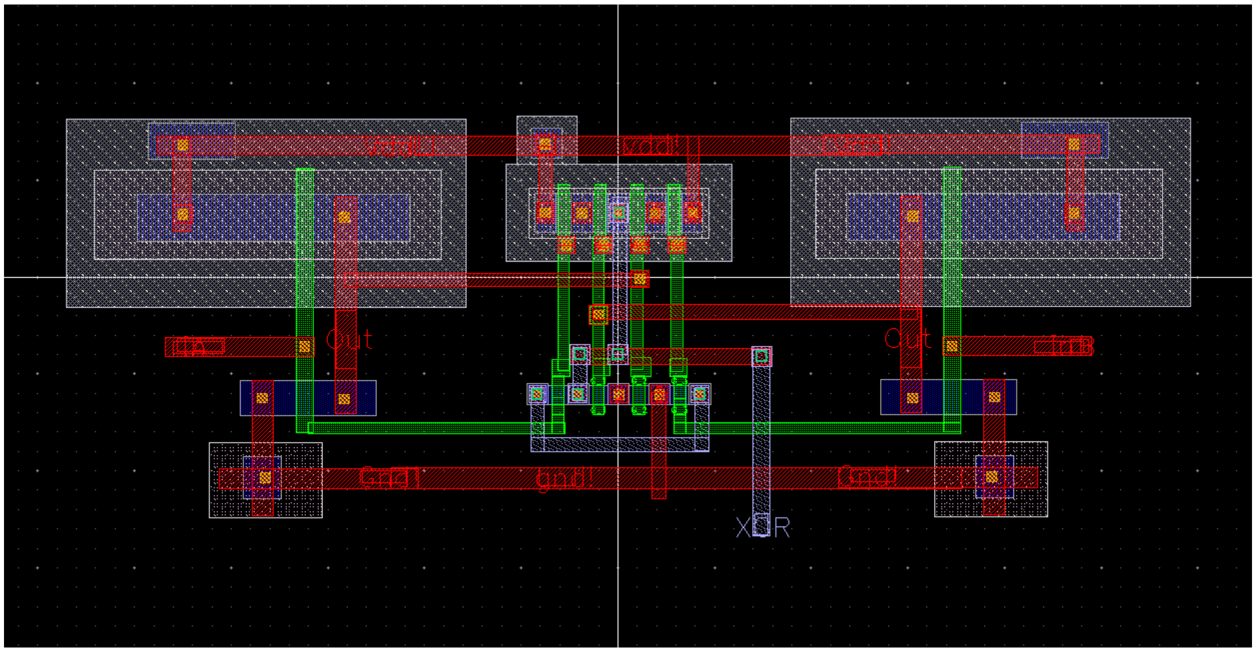

A
B
Out
[@instanceName]
[@partName]



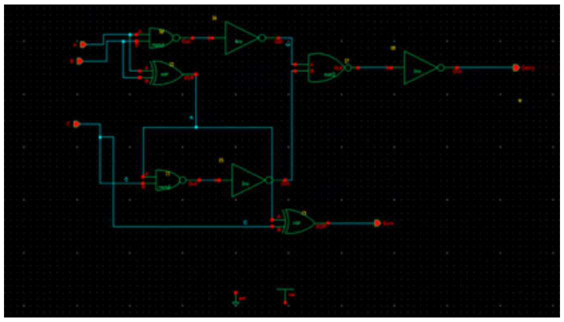
4. XOR

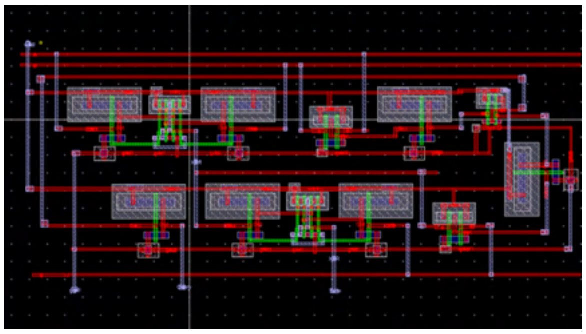


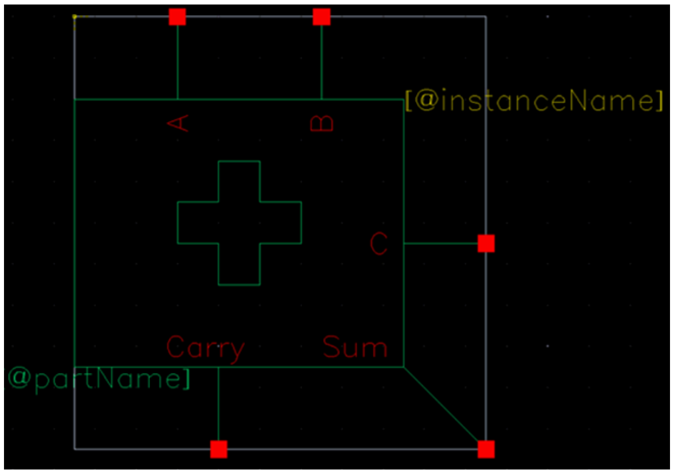




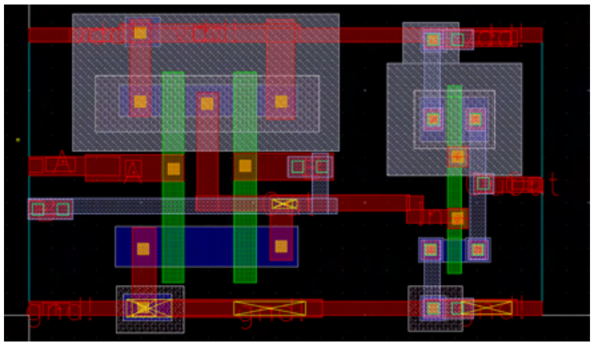
5. Full Adder

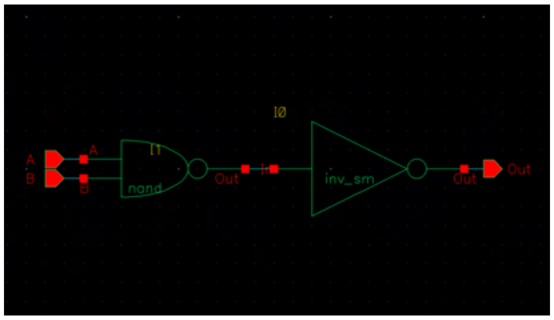




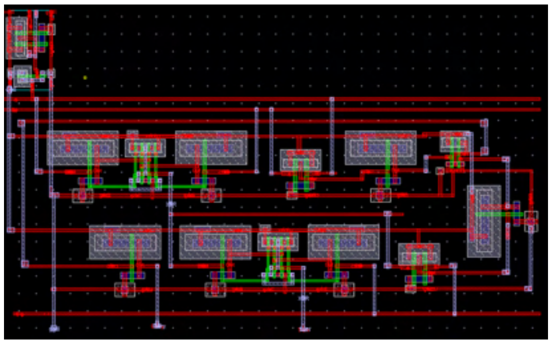


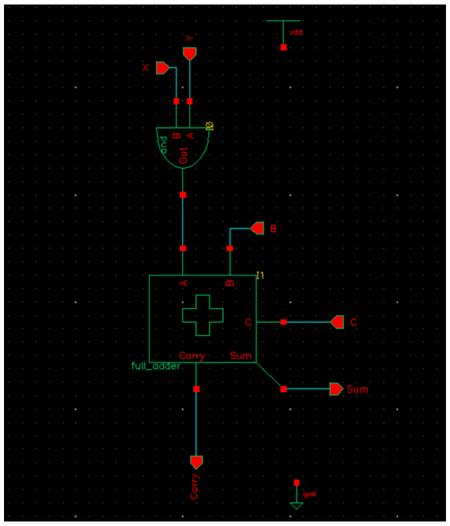
6. AND



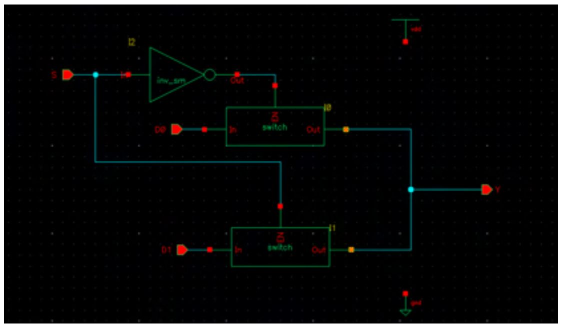


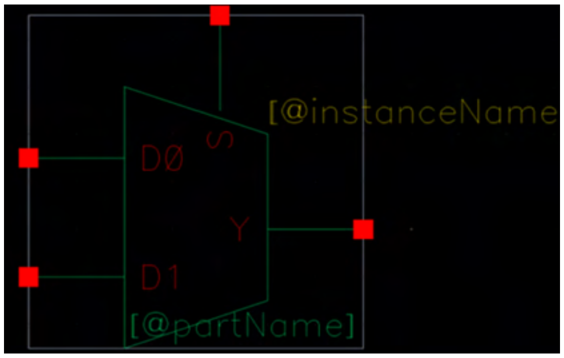
7. Full Adder and AND



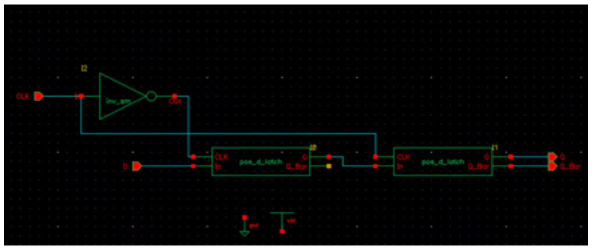


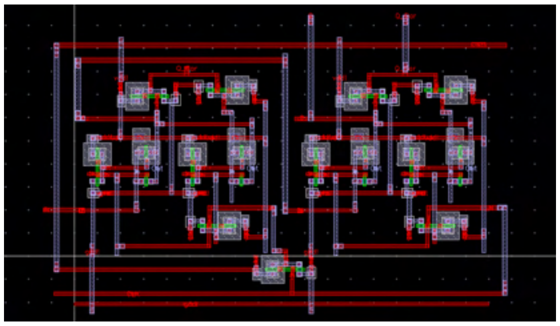
8. Multiplexor



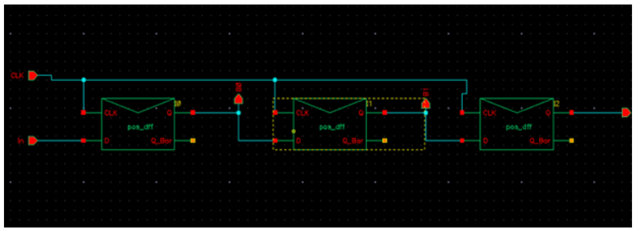


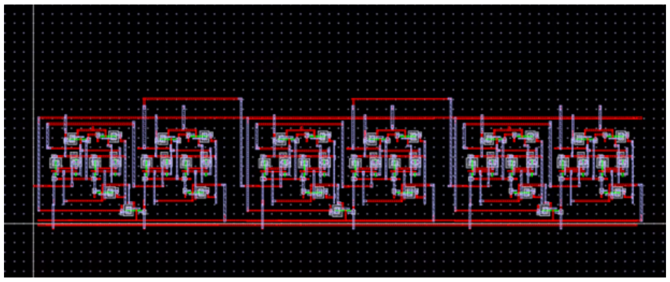
9. D Flip Flop

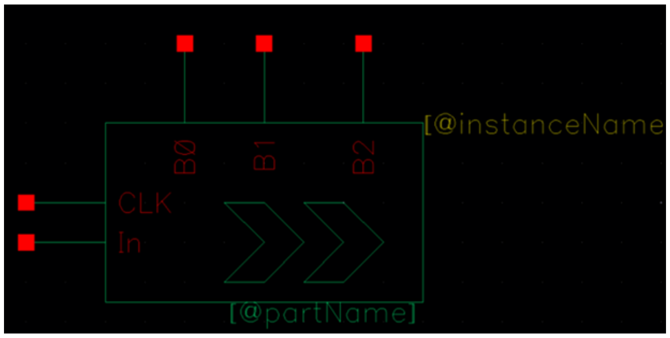




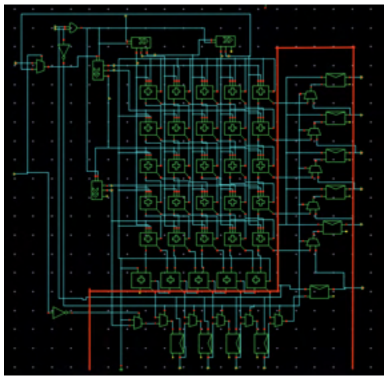
10. 3-bit Shift Register/Input Register

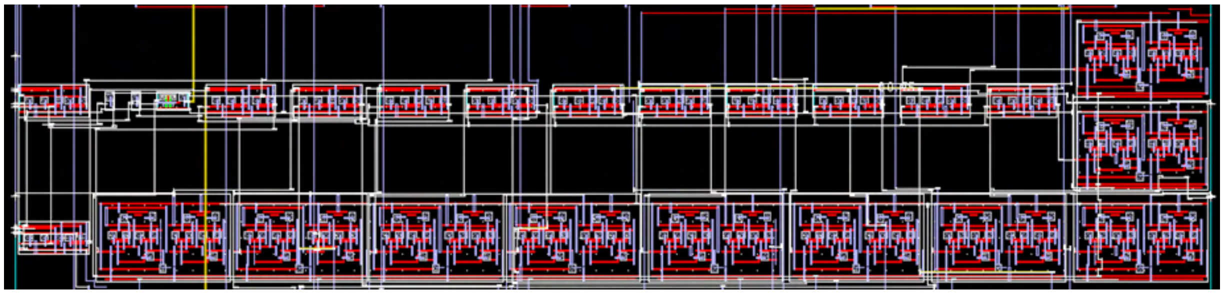






11. Output Register



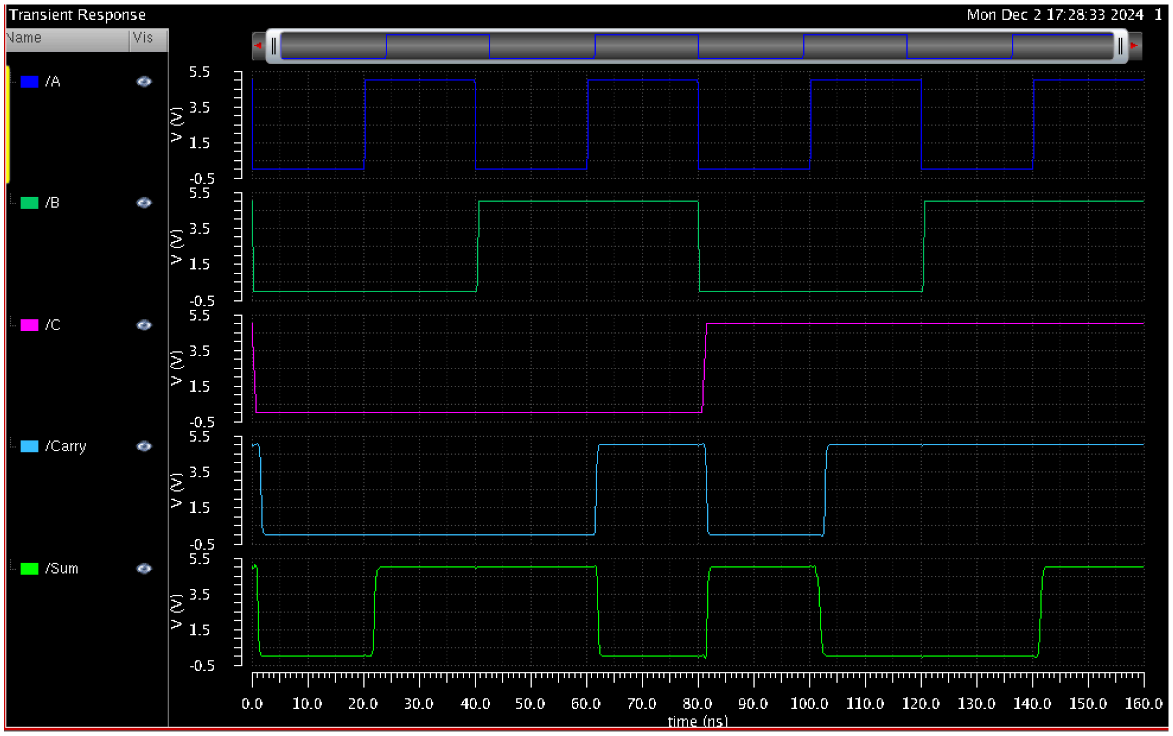


(1) **(20pts)** Simulation Results:

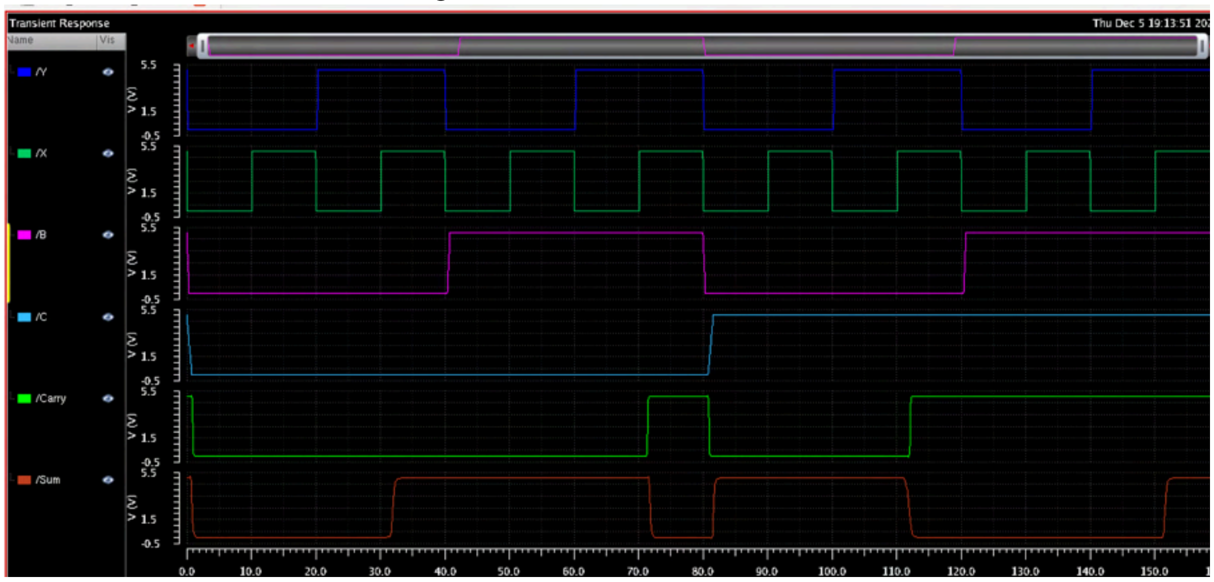
(a) (10 pts total) Individual cells:

I. Full Adder:

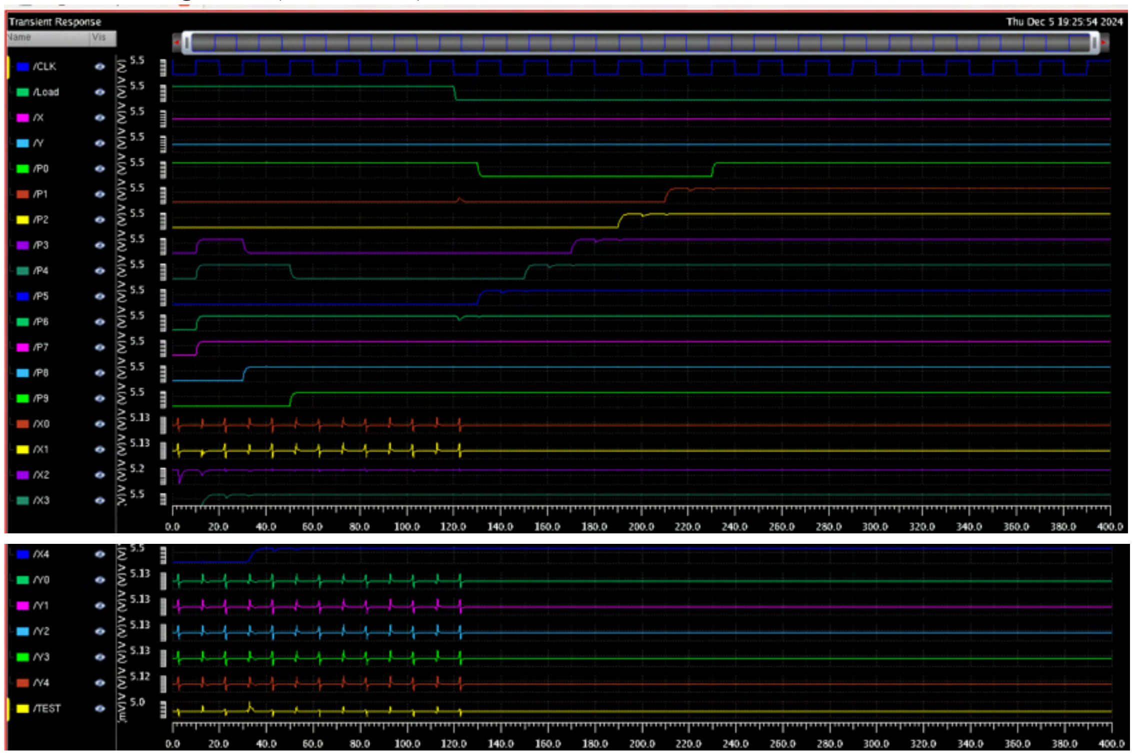
Simulation



II. Full Adder with AND gate:

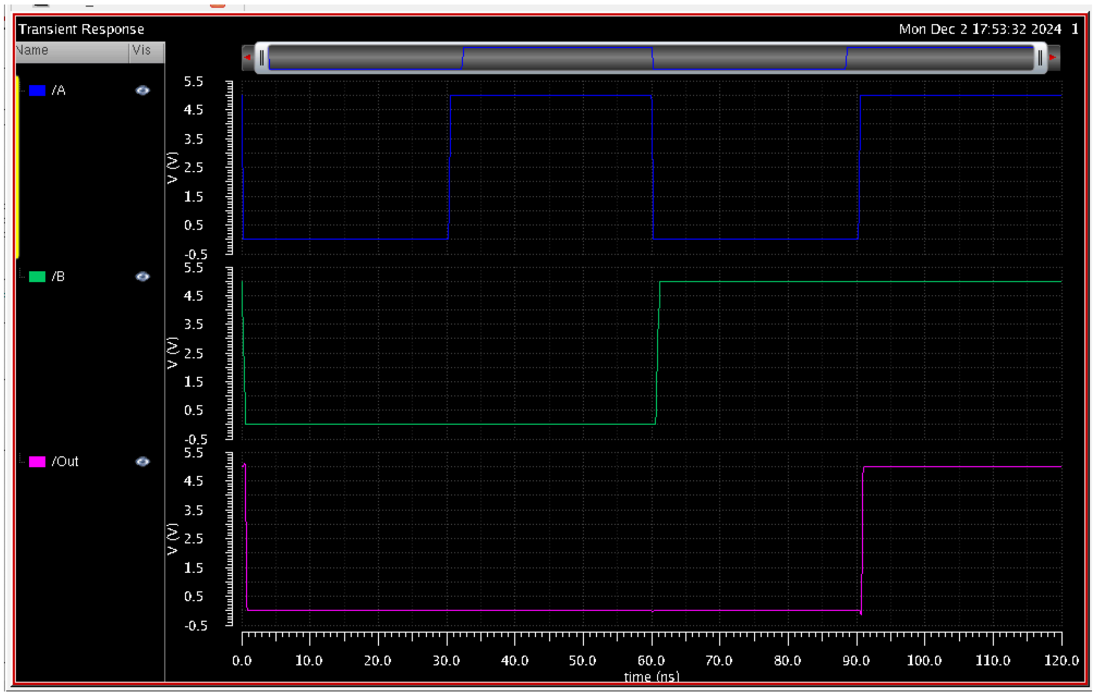


III. Registers (Test Mode)

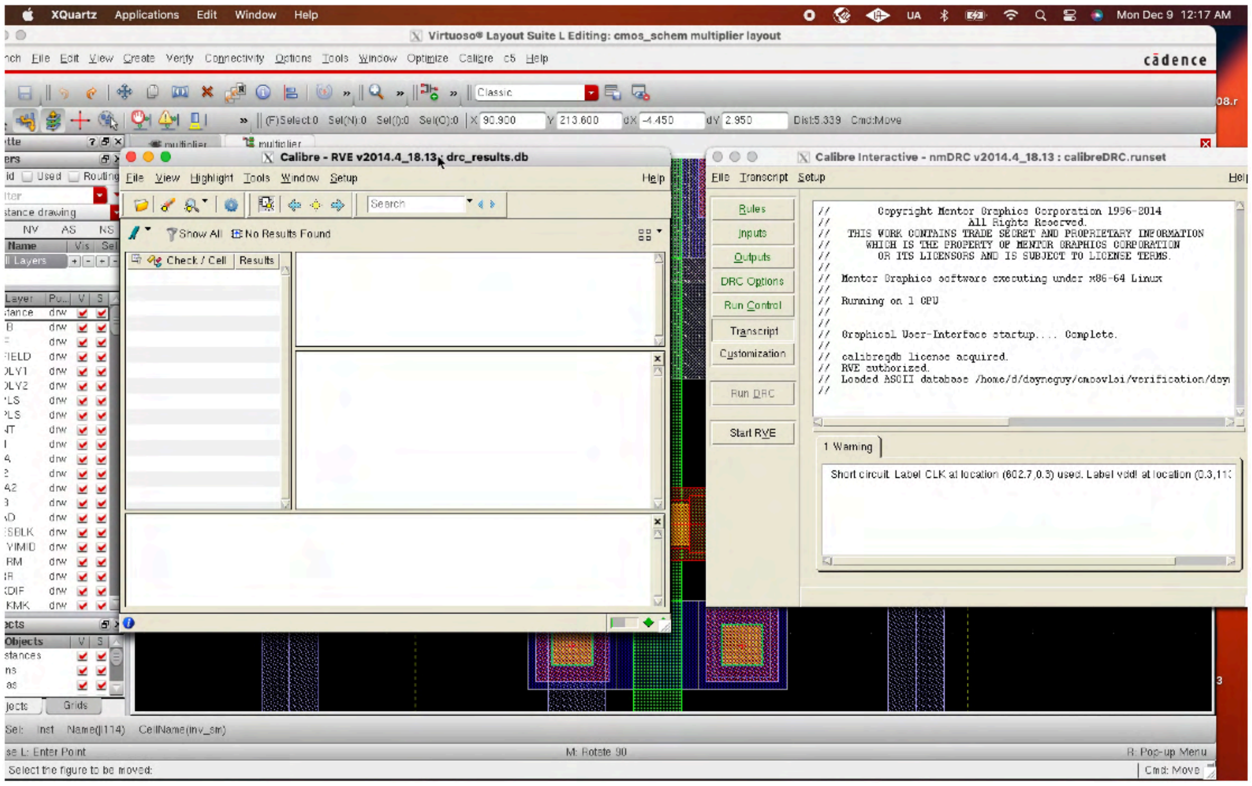


All input Fed through Y
11111 * 11111 = 1111000001
Output Stream is at P0

And Gate:



Design Rule Check:

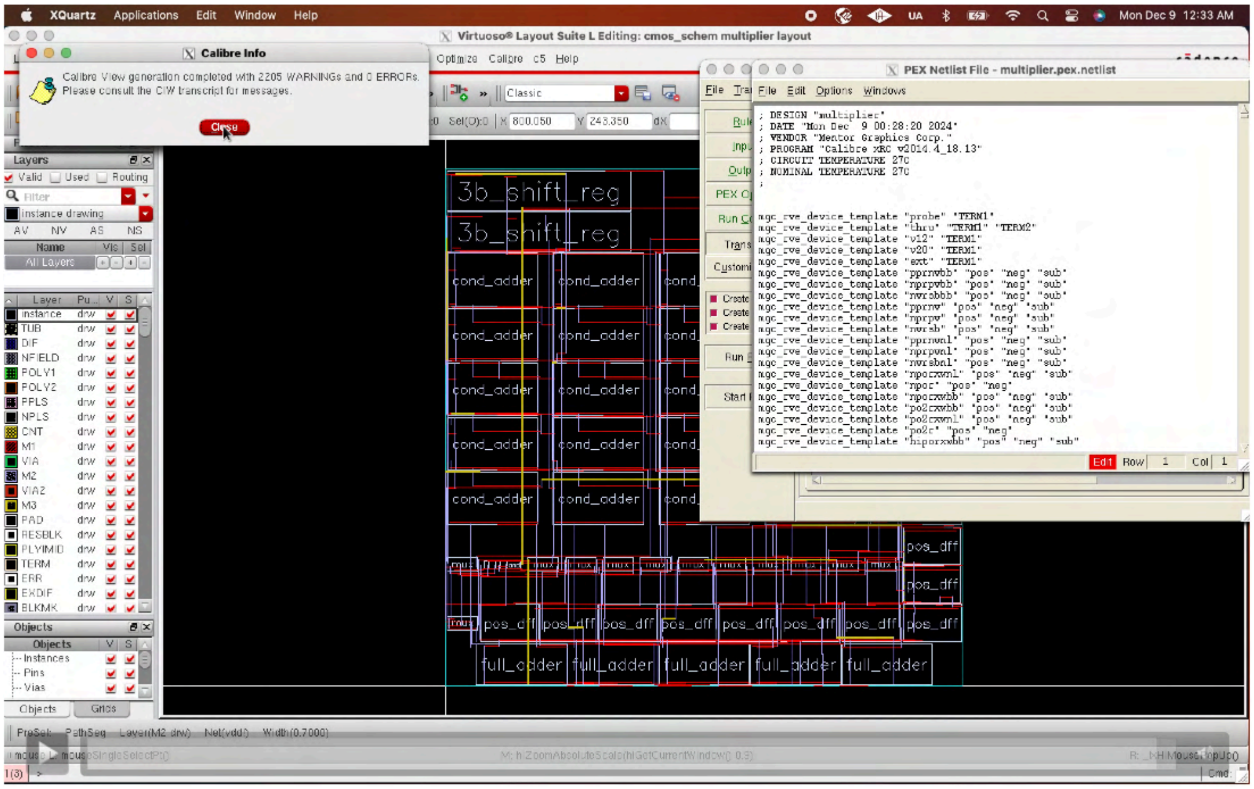

Virtuoso® Layout Suite L Editing: cmos_schem multiplier layout
Calibre - RVE v2014.4_18.13 : drc_results.db
Show All
No Results Found
Check / Cell
Results
Calibre Interactive - nmDRC v2014.4_18.13 : calibreDRC.runset
Rules
Inputs
Outputs
DRC Options
Run Control
Transcript
Customization
Run DRC
Start RVE

```
//          Copyright Mentor Graphics Corporation 1996-2014
//                     All Rights Reserved.
//   THIS WORK CONTAINS TRADE SECRET AND PROPRIETARY INFORMATION
//      WHICH IS THE PROPERTY OF MENTOR GRAPHICS CORPORATION
//         OR ITS LICENSORS AND IS SUBJECT TO LICENSE TERMS.
//
//  Mentor Graphics software executing under x86-64 Linux
//
//  Running on 1 CPU
//
//
//  Graphical User-Interface startup.... Complete.
//
//  calibreqdb license acquired.
//  RVE authorized.
//  Loaded ASCII database /home/d/dayneguy/cmosvlsi/verification/day
//
```

1 Warning

```
Short circuit. Label CLK at location (602.7,0.3) used. Label vdd! at location (0.3,11
```

Parasitic Extraction:


Calibre Info
Close
Virtuoso® Layout Suite L Editing: cmos_schem multiplier layout
PEX Netlist File - multiplier.pex.netlist
3b_shift_reg
cond_adder
pos_dff
full_adder

Calibre View generation completed with 2205 WARNINGs and 0 ERRORs.
Please consult the CIW transcript for messages.

```
; DESIGN "multiplier"
; DATE "Mon Dec  9 00:28:20 2024"
; VENDOR "Mentor Graphics Corp."
; PROGRAM "Calibre xRC v2014.4_18.13"
; CIRCUIT TEMPERATURE 27C
; NOMINAL TEMPERATURE 27C
;


mgc_rve_device_template "probe" "TERM1"
mgc_rve_device_template "thru" "TERM1" "TERM2"
mgc_rve_device_template "v12" "TERM1"
mgc_rve_device_template "v20" "TERM1"
mgc_rve_device_template "ext" "TERM1"
mgc_rve_device_template "pprnvbb" "pos" "neg" "sub"
mgc_rve_device_template "nprpvbb" "pos" "neg" "sub"
mgc_rve_device_template "nvrsbbb" "pos" "neg" "sub"
mgc_rve_device_template "pprnv" "pos" "neg" "sub"
mgc_rve_device_template "nprpv" "pos" "neg" "sub"
mgc_rve_device_template "nvrsb" "pos" "neg" "sub"
mgc_rve_device_template "pprnvnl" "pos" "neg" "sub"
mgc_rve_device_template "nprpvnl" "pos" "neg" "sub"
mgc_rve_device_template "nvrsbnl" "pos" "neg" "sub"
mgc_rve_device_template "nporxwnl" "pos" "neg" "sub"
mgc_rve_device_template "npor" "pos" "neg"
mgc_rve_device_template "nporxwbb" "pos" "neg" "sub"
mgc_rve_device_template "po2rxwbb" "pos" "neg" "sub"
mgc_rve_device_template "po2rxwnl" "pos" "neg" "sub"
mgc_rve_device_template "po2r" "pos" "neg"
mgc_rve_device_template "hiporxwbb" "pos" "neg" "sub"
```

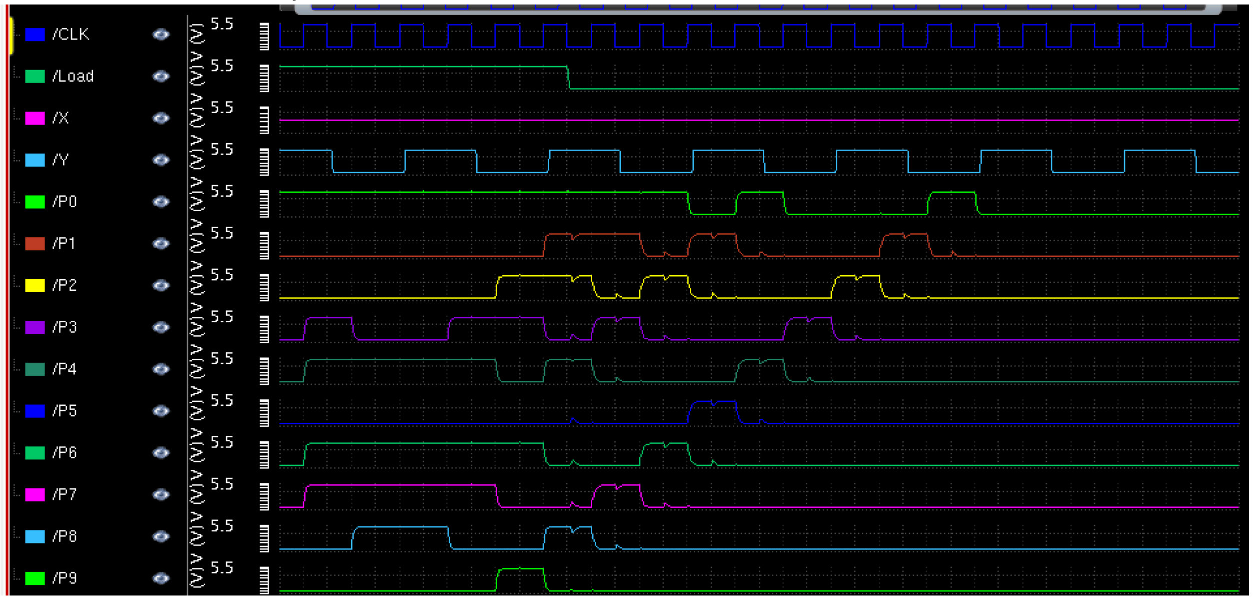
Testing:

Testing was conducted at a 50 MHz frequency (a clock period of 20ns)

Normal mode:

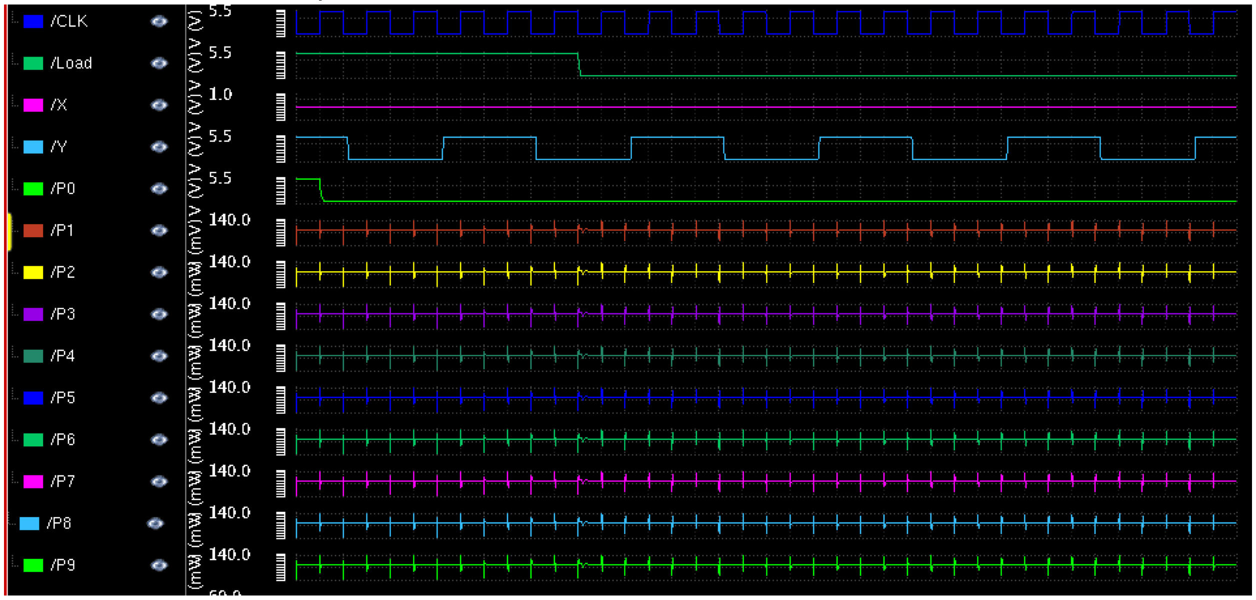
1. Test 1

   X=11111, Y=01001, Expected: 0100010111

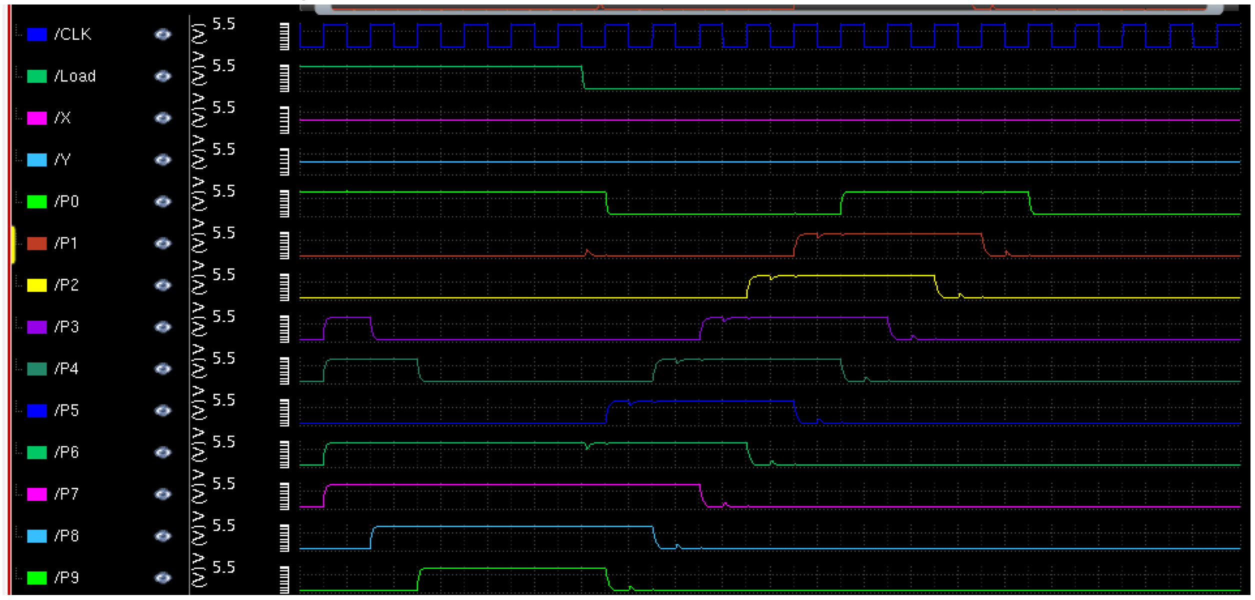


2. Test 2

   X= 00000, Y= 11001, Expected: 0000000000

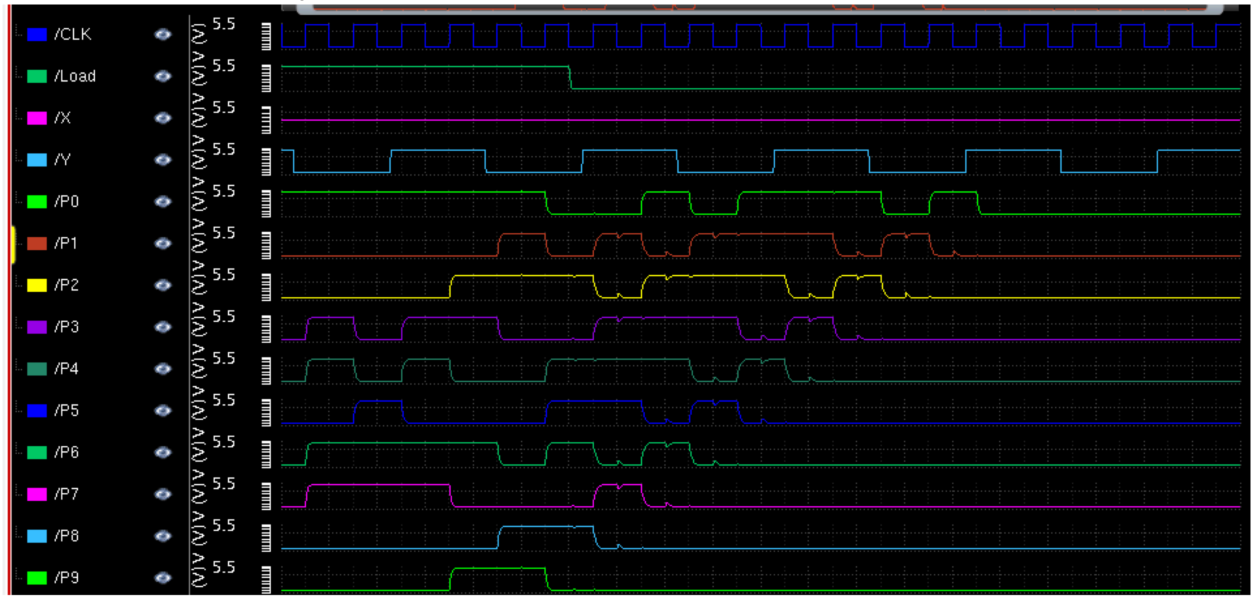


3. Test 3
X = 11111, Y=11111, Expected: 1111000001



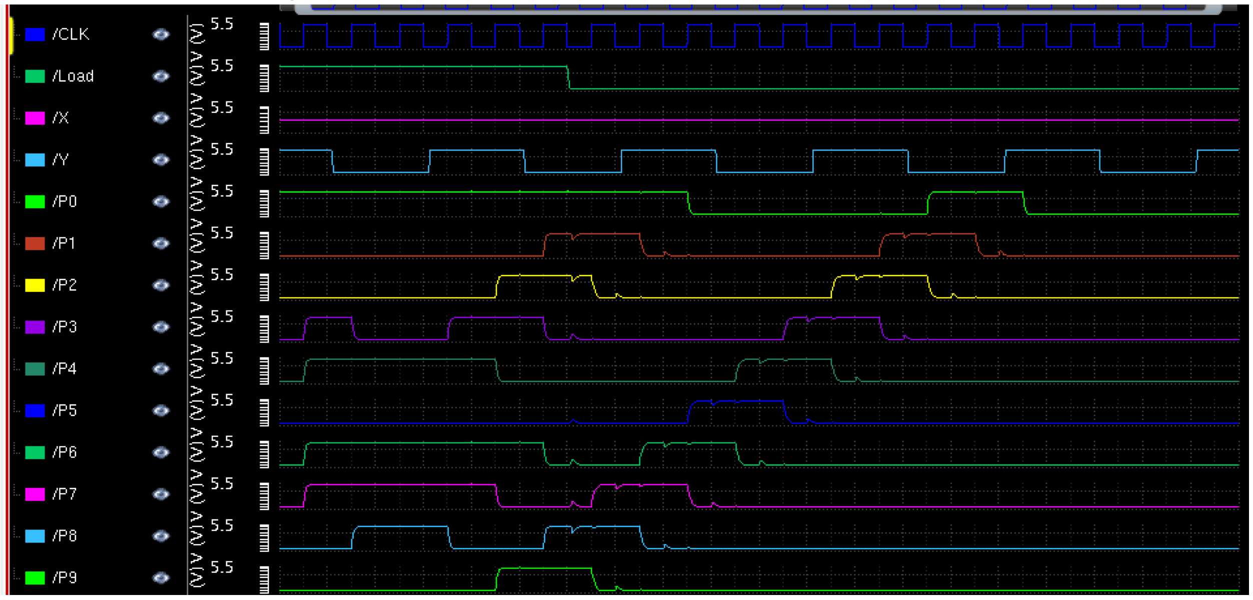
Value can be read from P0-P9 with P0 being the serial output 1000001111 in Least Significant bit or 1111000001 in standard binary form

4. Test 4
X=11111, Y =00110, Expected: 010111010

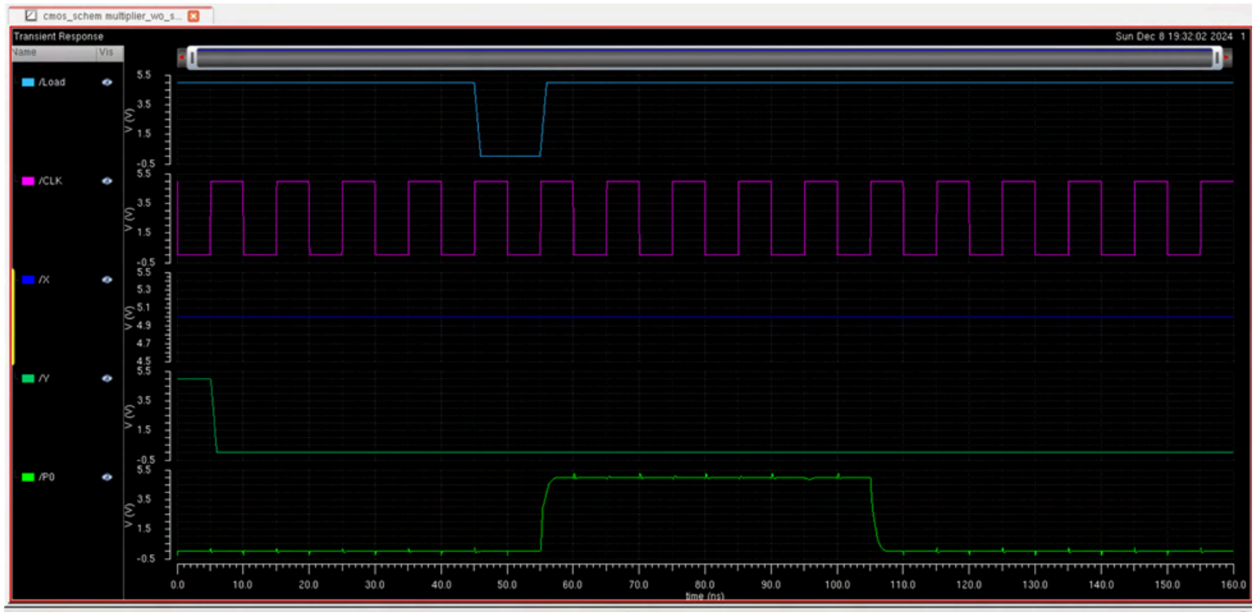


Read P0 - P9 0101110100 in LSB Form

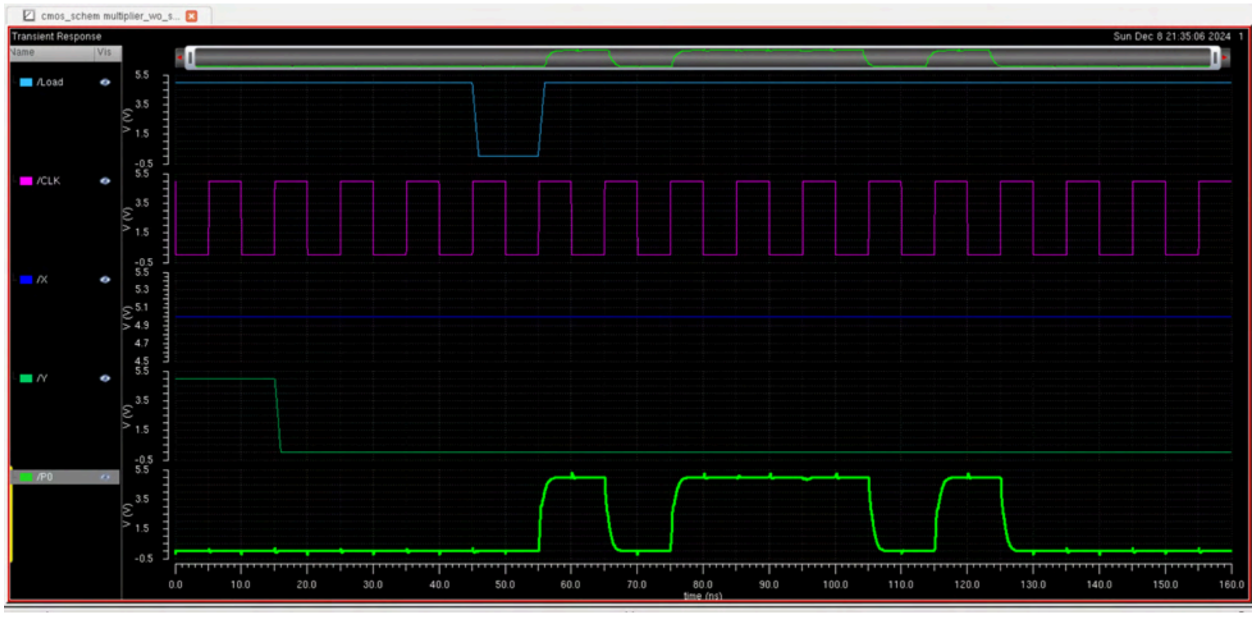
5. Test 5
X=11111, Y=11001, Expected: 1110000011



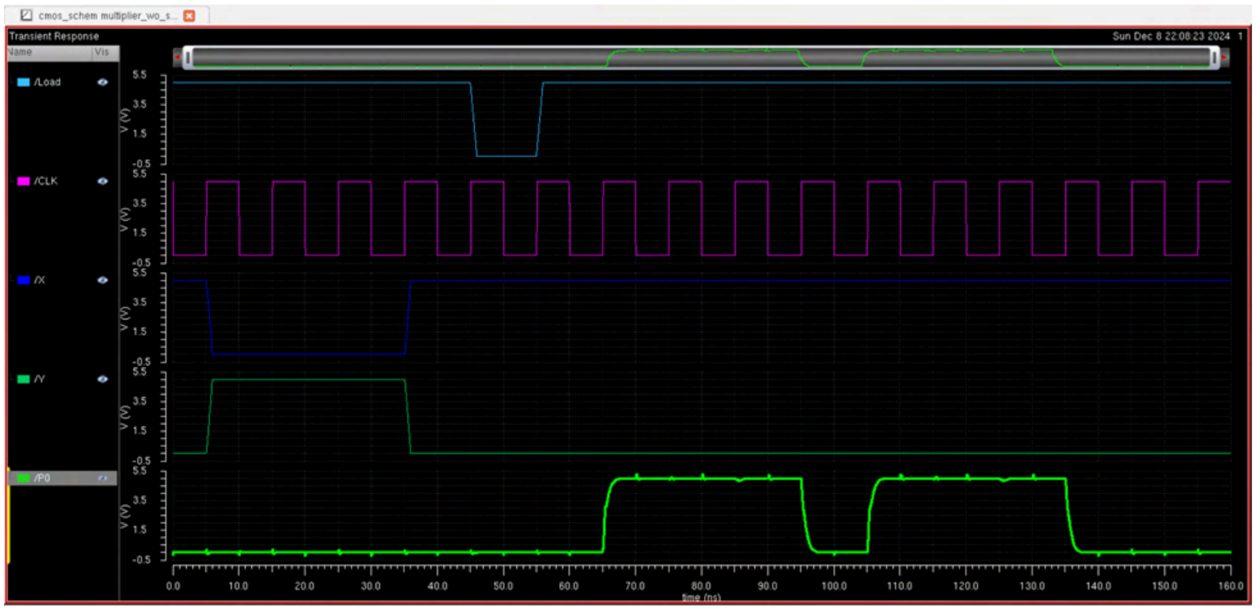
6. Test 6
X=00001, Y=11111, Expected: 0000011111



7. Test 7
X=11111, Y=00011, Expected: 0001011101

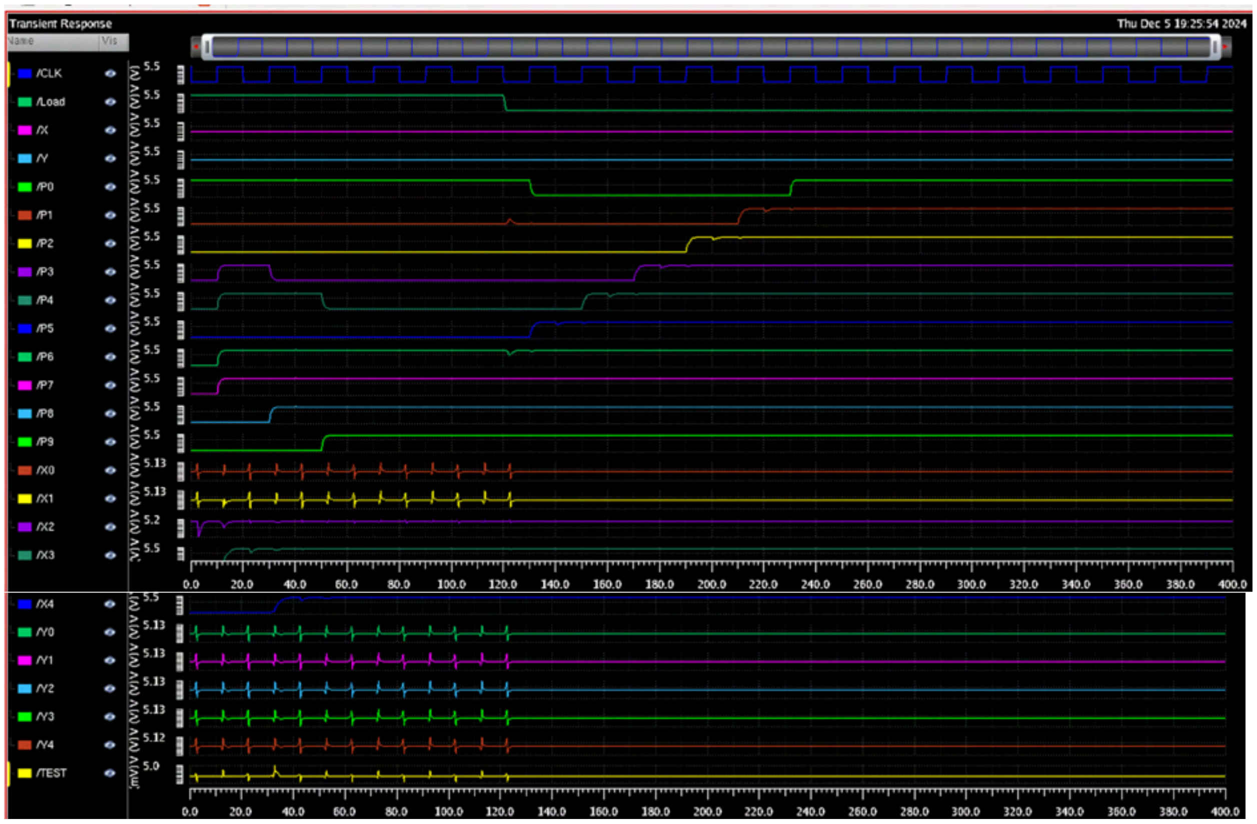


8. Test 8
X=10001, Y=01110, Expected: 011101110

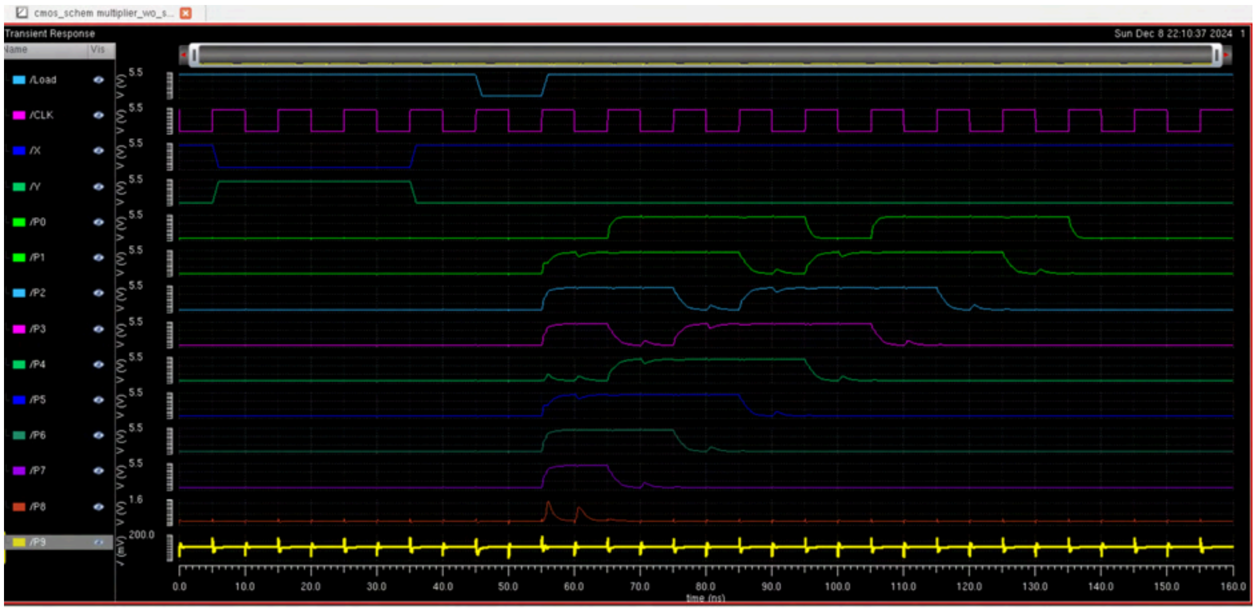


Test Mode:

1. 11111 * 11111 = 1111000001



2. 10001 x 01110 = 011101110



Future Improvements

1. A major limiting factor of our design was our earlier design -notably our inverter. While we did create a smaller version of the inverter, other layouts, like the XOR and full adder already had the larger design embedded into it. By recreating other core bit slices with smaller area, we could increase the size of N.
2. Replace D Flip Flops with SRAM, due to its smaller transistor count, it would be possible to reduce the size of our design by implementing as SRAM.
3. Add an additional register for each input and output (or add registers throughout the combinational circuit). This would reduce available space in the limited area but would reduce the impact of metastability and decrease the critical path.
4. SIPO and PISO registers to be implemented as bits. This would've sped up our design process.