CDA 4213/CIS 6930 CMOS VLSI

Fall 2024

Final Project

Due date(s)

Friday, 6th December 2024

Today's Date:	December 5th, 2024
Your Team Name:	*The Floral Princesses*
Team Members:	*Aidan Khalil – U9240-8495* *Sergio Flores – U9506-2088*
Work Distribution	***Explain in detail who has done what. Each team member's grade will be based on their overall contribution.*** ***1) Aidan Khalil*** *– nand, inverter, full adder, 4x4 multiplier* ***2) Sergio Flores*** *– and, mux, input / output registers, testing*
No. of Hours Spent:	50
Exercise Difficulty: (Easy, Average, Hard)	Average
Any Feedback:	N/A

(10 pts) Proposed Design – Bit slice design

(a) List all module bit-slices you have used for your design. ✓

- x1 4x4 multiplier (16 ANDs + 20 Full Adders)

- x9 MUXs *(for selections)*

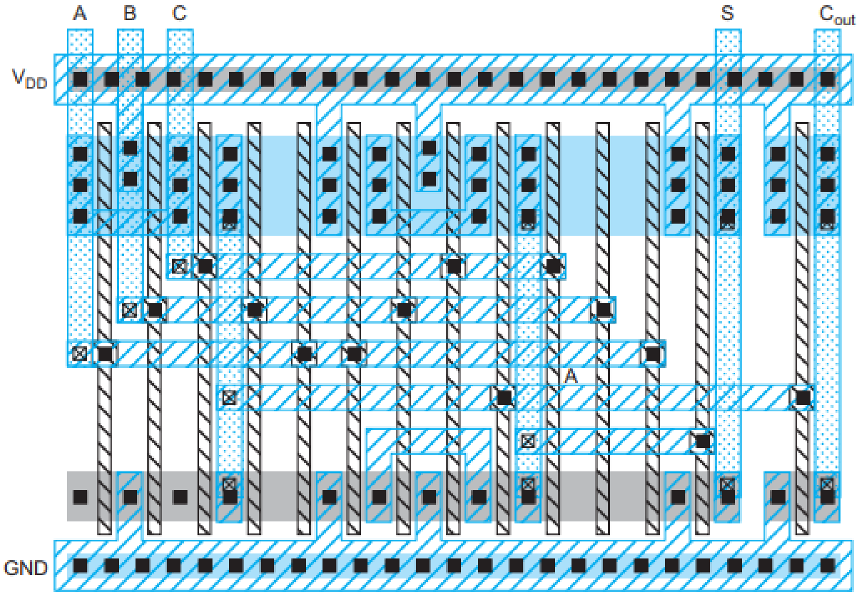
- x2 sets *of 4-bit shift registers (SIPO for input Q1-Q8)*

- x1 set *of 8-bit shift registers (PISO as output Q9-Q16(labeled P))*

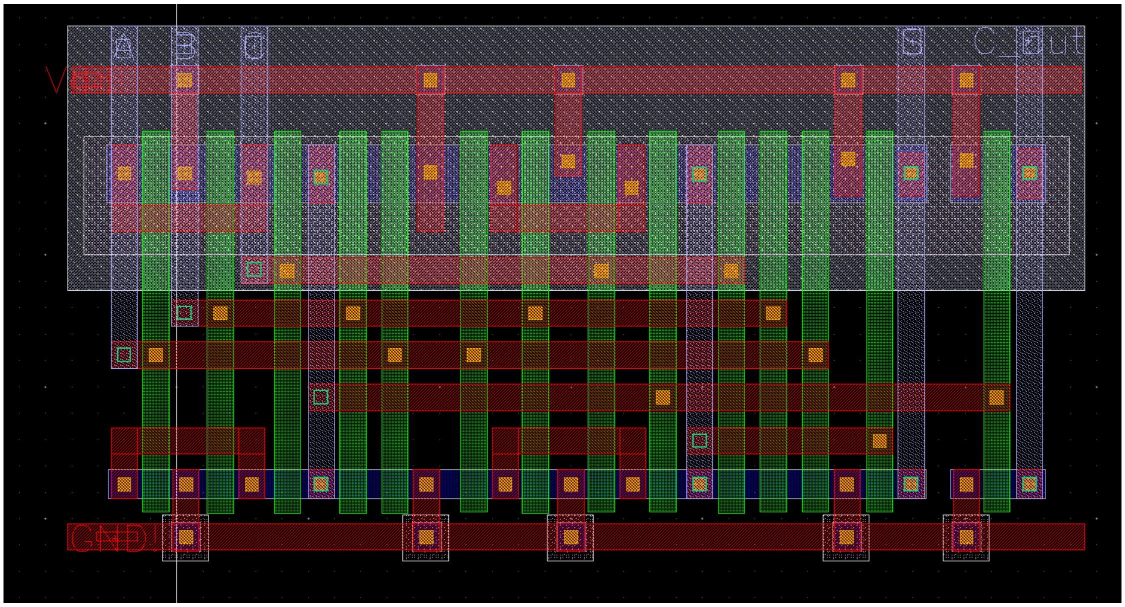
(b) For each bit slice, show the gate-level design and layout design. For layout, include the snapshot from Cadence Virtuoso. If you have used any other blocks, include them as well.

Full Adder:

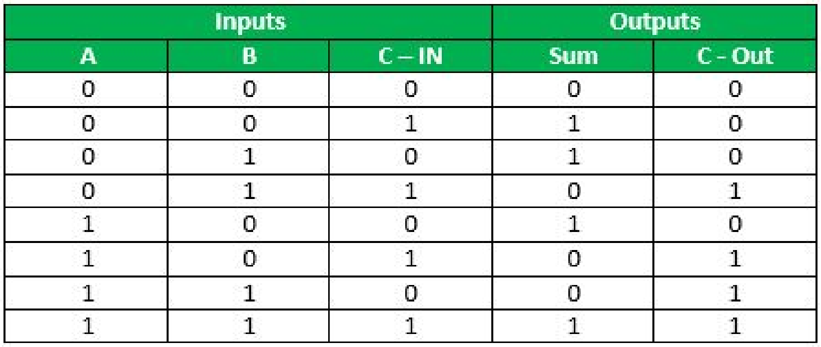
Full Adder (Textbook Reference): ✓



Full Adder in Cadence Virtuoso (Layout XL): ✓

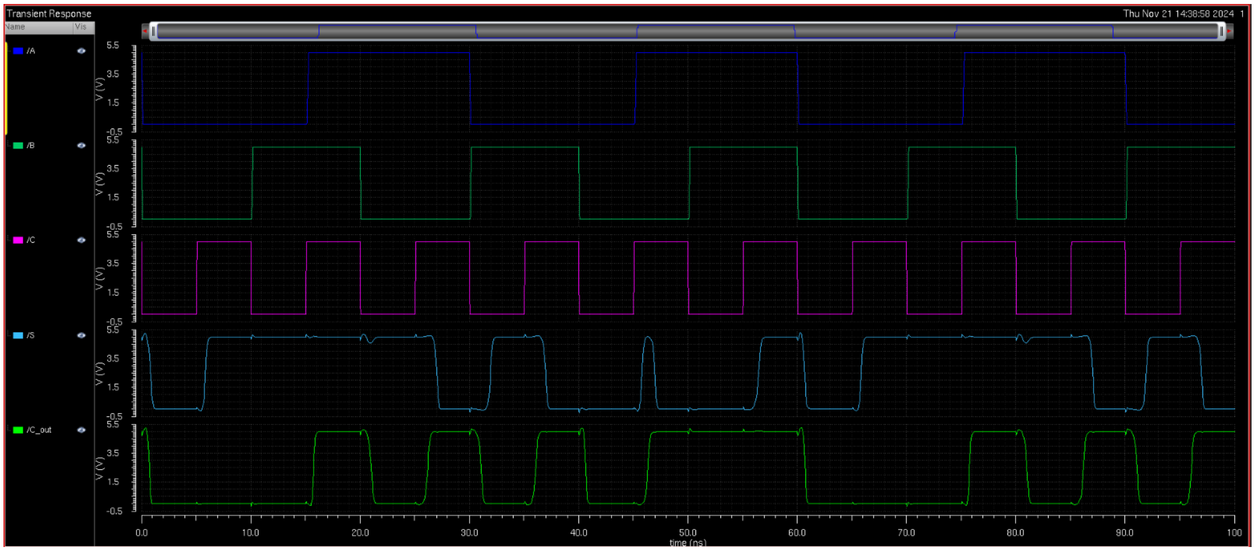

A
B
C
S
C_out
Vdd
GND

Full Adder Truth Table: ✓



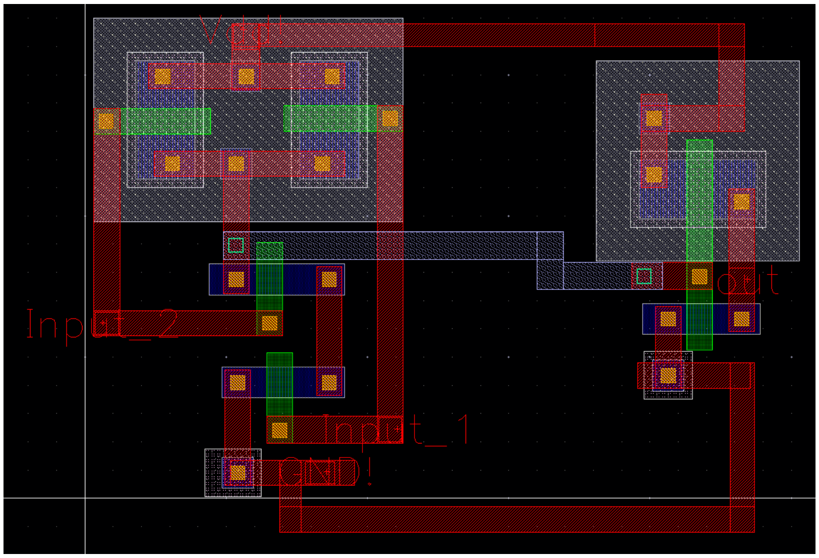
Inputs			Outputs	
A	B	C – IN	Sum	C - Out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Full Adder Waveform in Calibre (ADE L): ✓

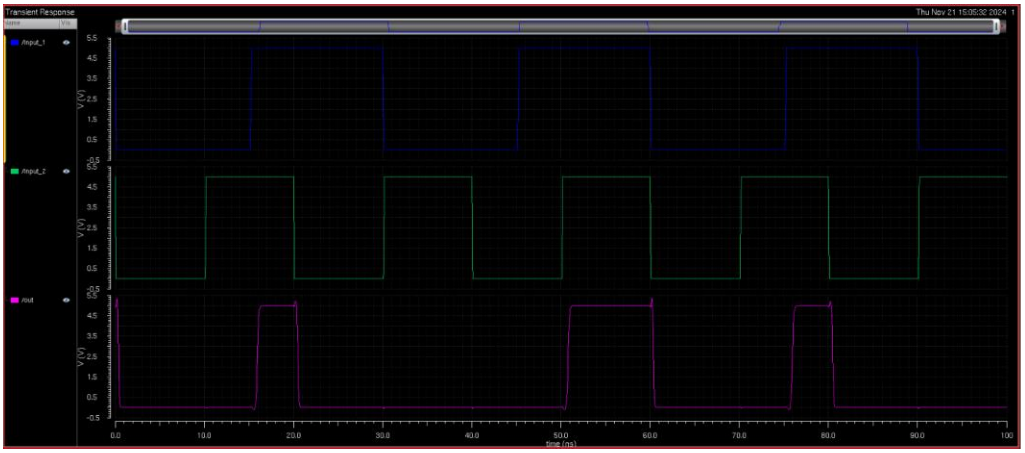


Full Adder with AND gate:

AND in Virtuoso (Layout XL, using NAND + Inverter): ✓

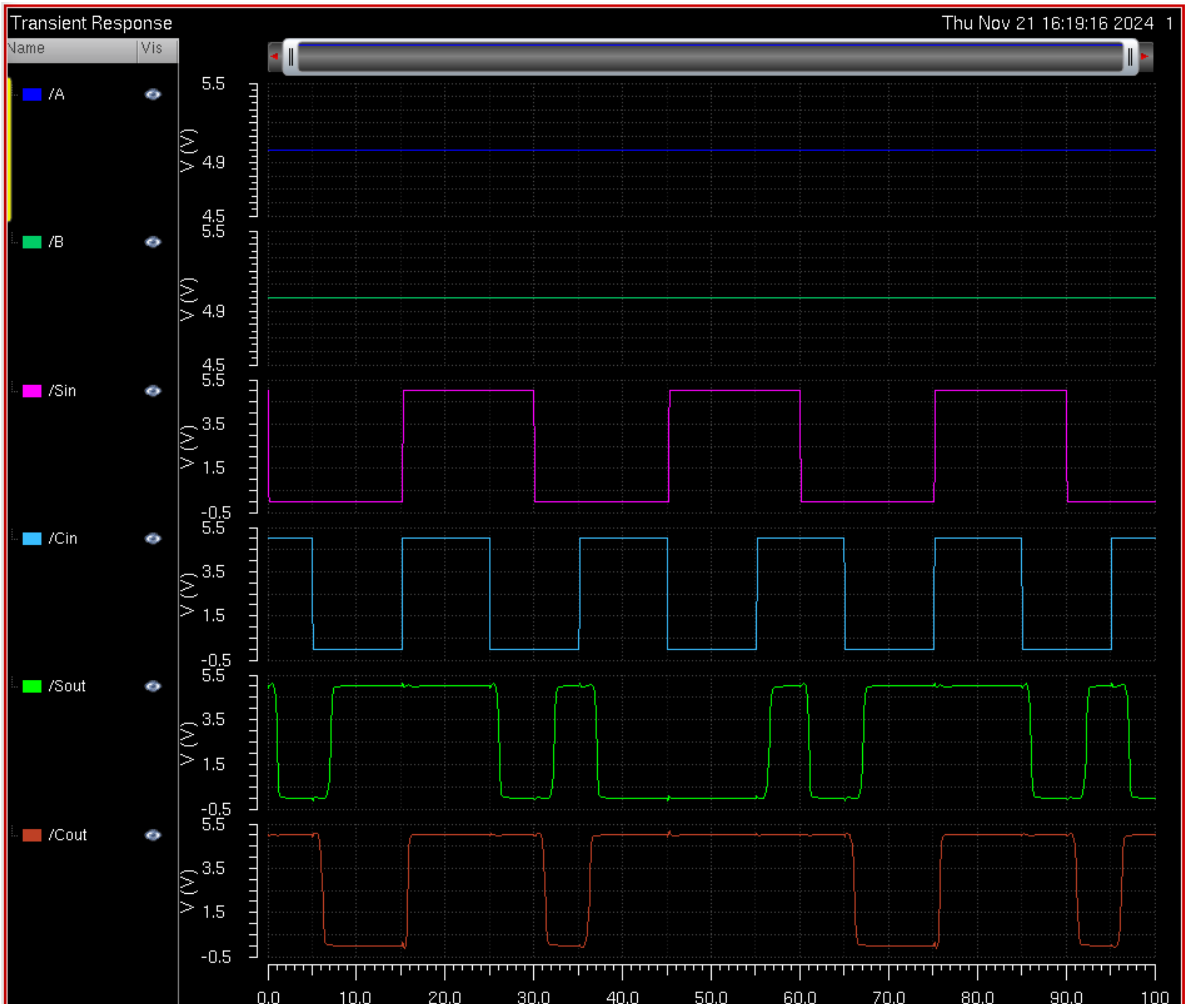


AND Waveform in Calibre (ADE L): ✓



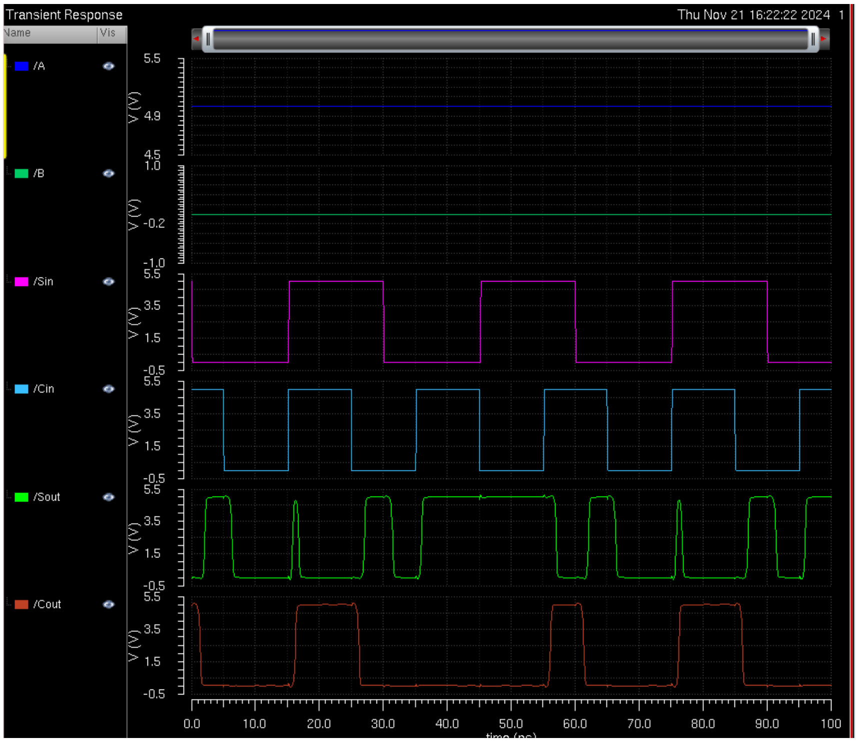
Full Adder w/ AND gate: ✓

A = high, B = high

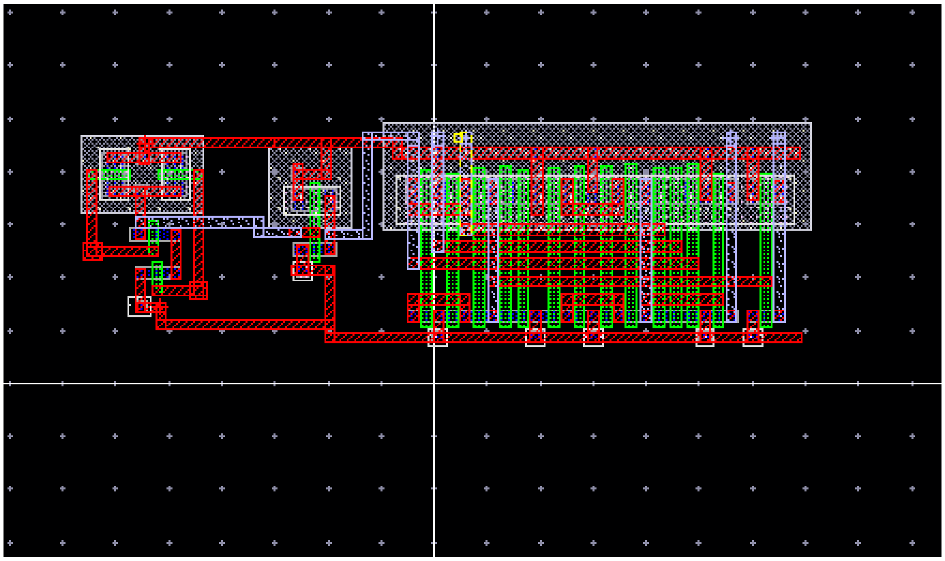


Full Adder w/ AND continued... ✓

A = high, B = low



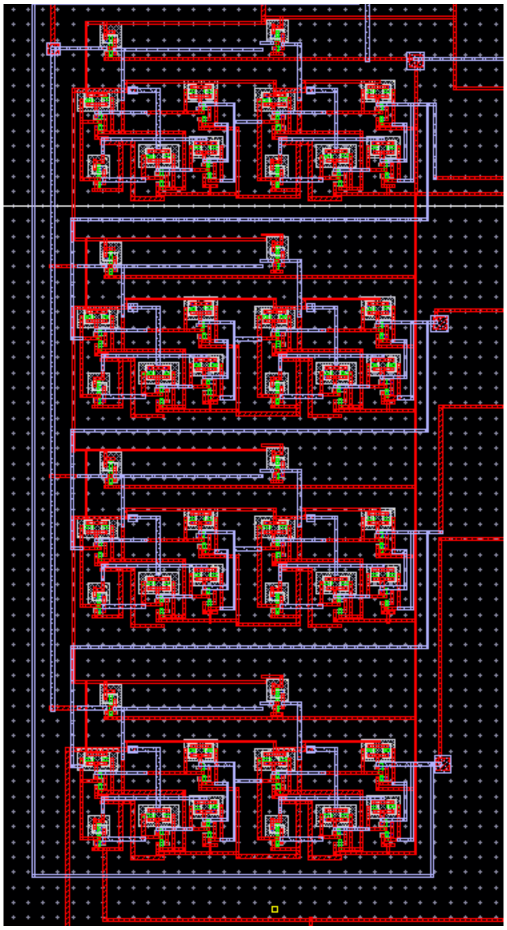
Full Adder w/ AND layout ✓



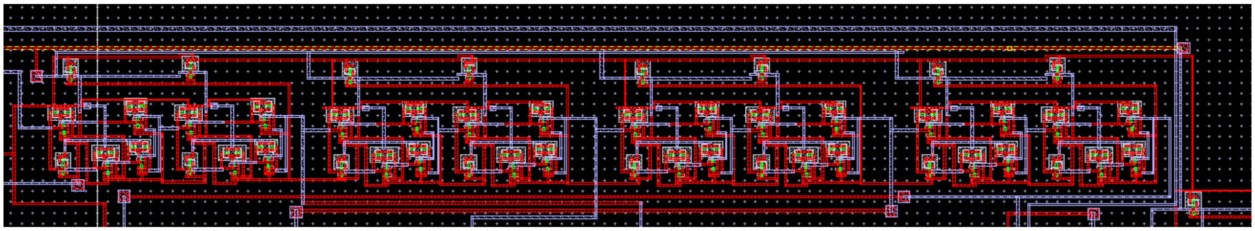
Registers (**Input** and **Output**):

Inputs: ✓

Shift registers SIPO X

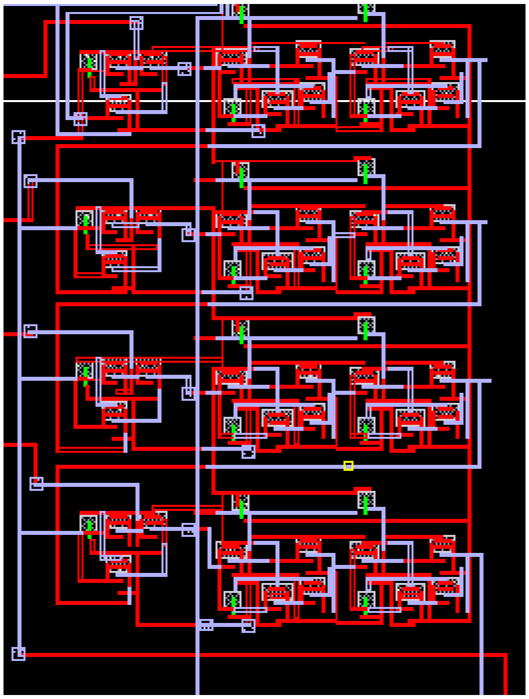


Shift registers SIPO Y

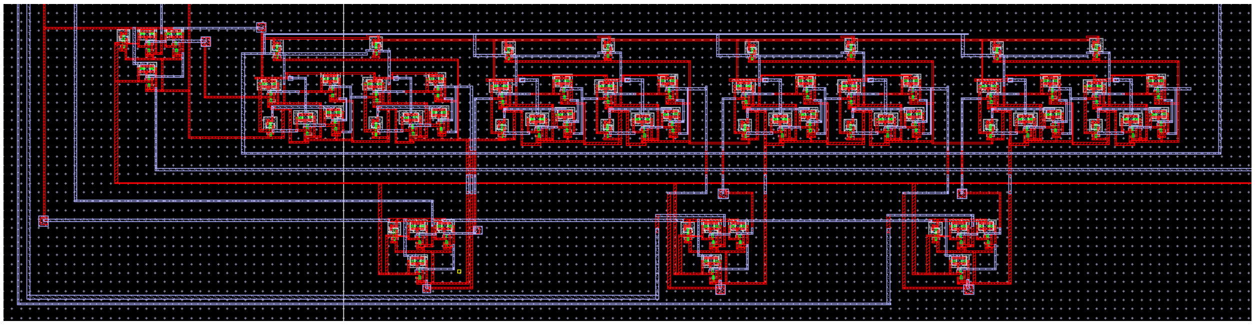


Outputs: ✓

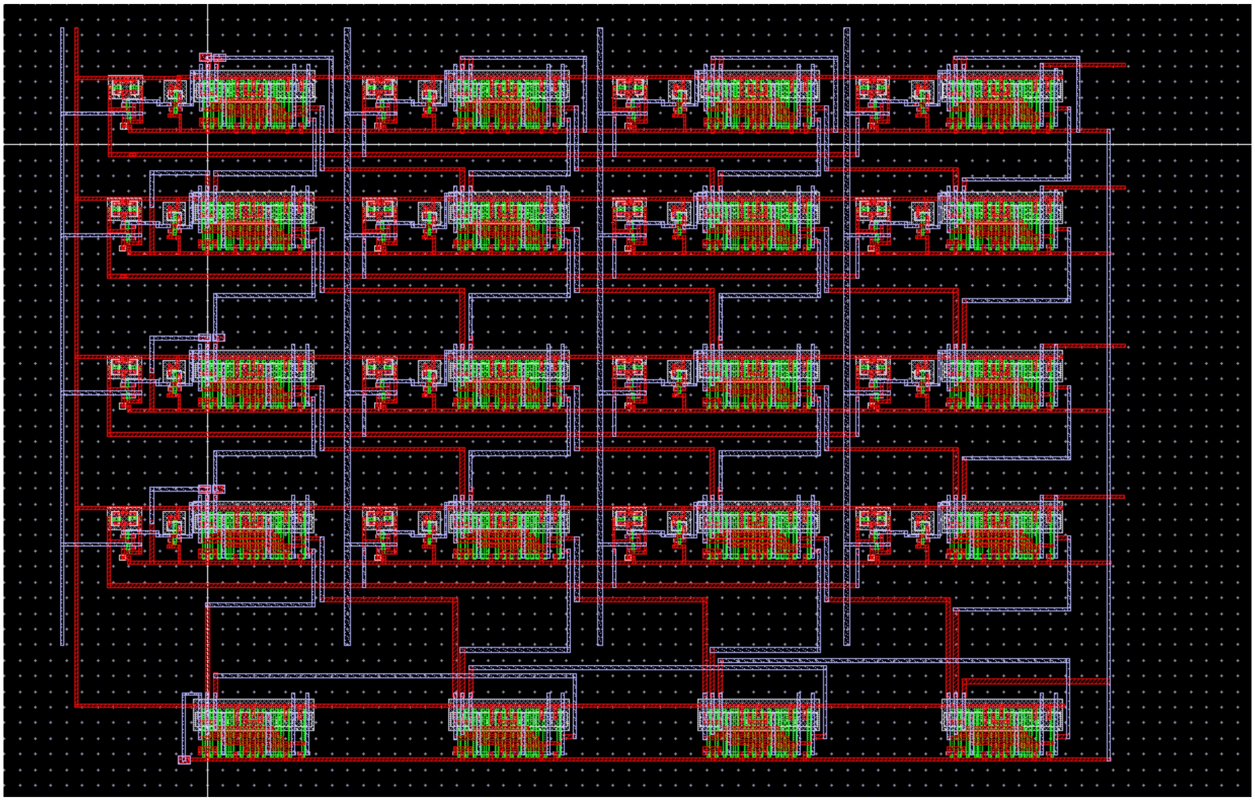
The first 4 (Far right PISO Outputs)



The second 4



4x4 array multiplier: ✓



Reference:

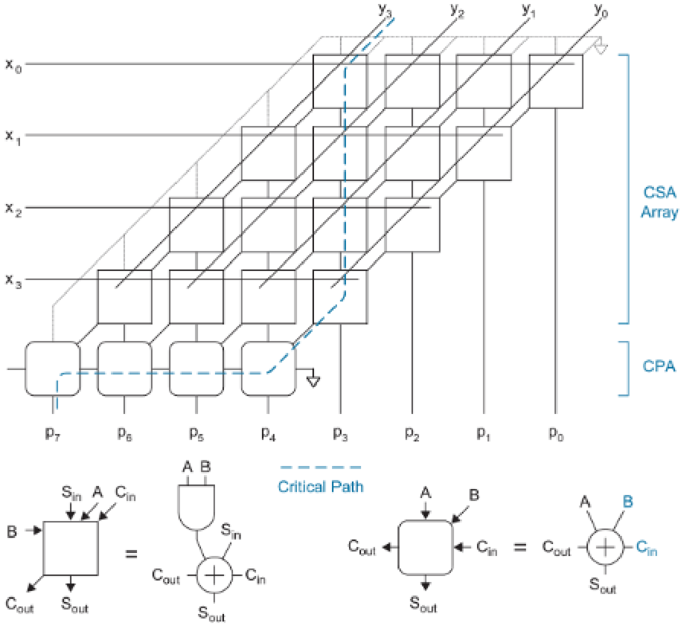
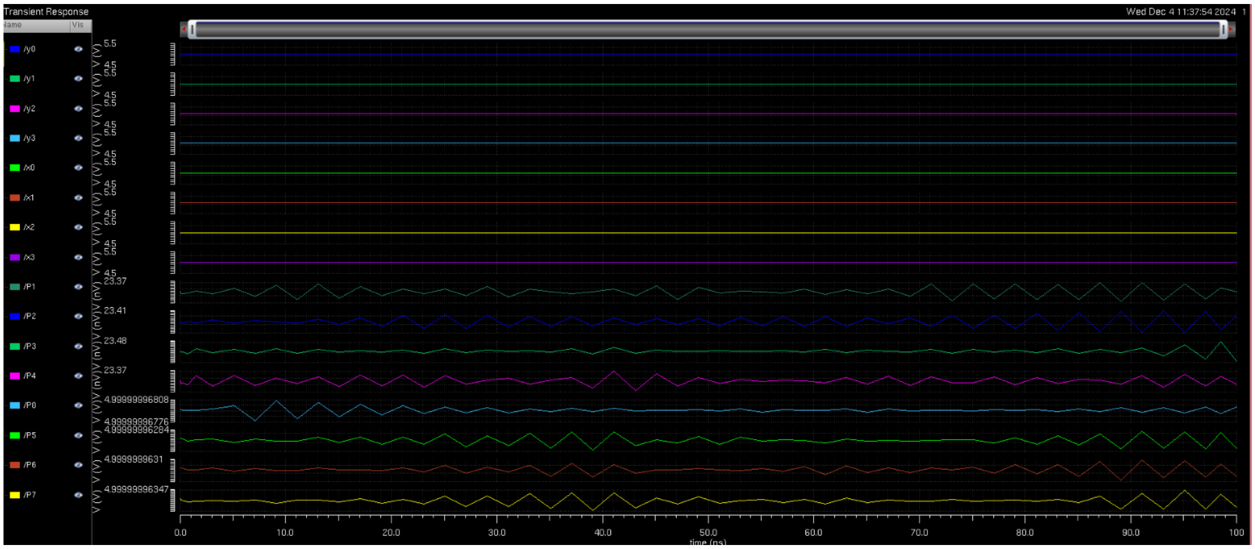


FIG 10.70 Array multiplier

Figure 2: Bit-sliced array multiplier example

Multiplier (4x4 array) waveform: ✓

Vector tested: 1111 * 1111

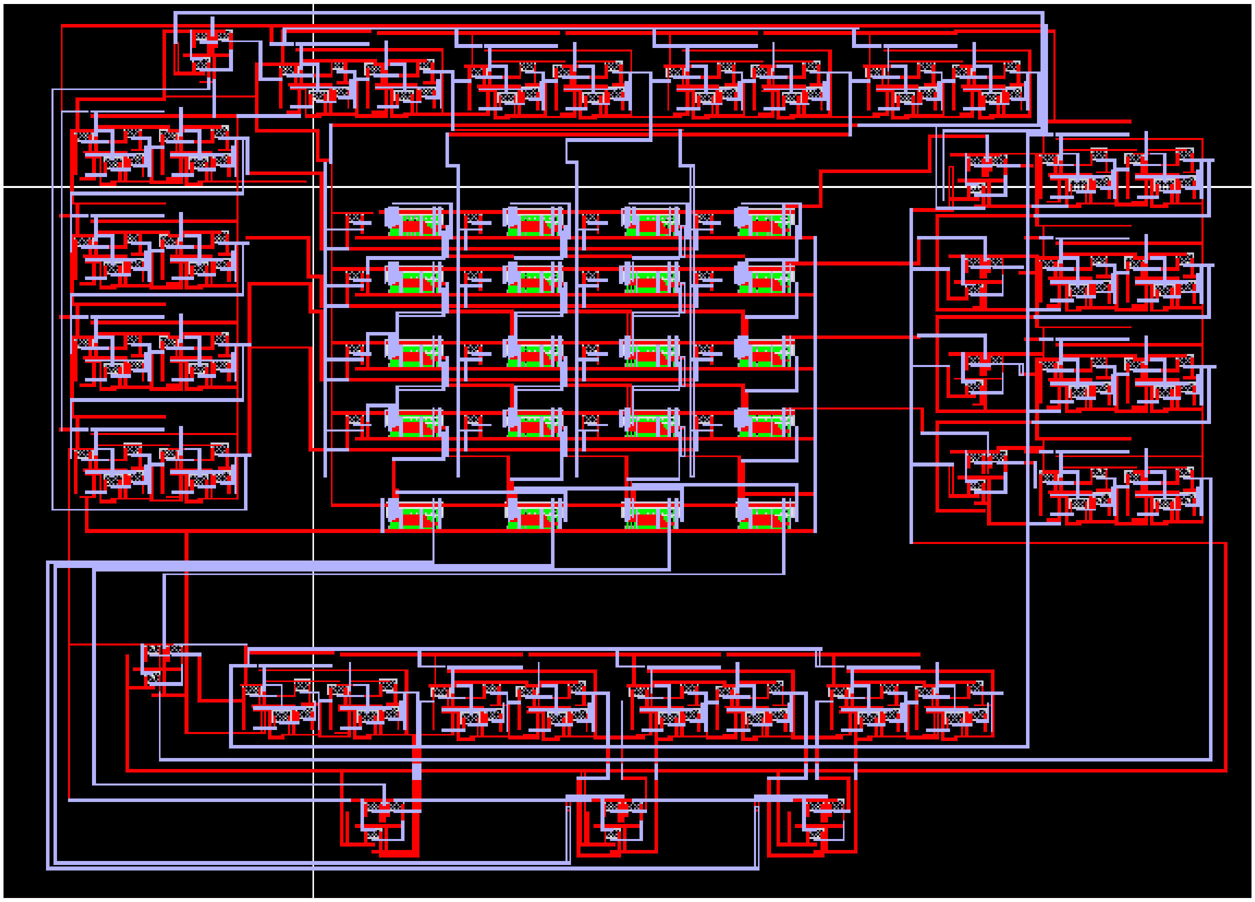


Ring Oscillator (If used):

Not applicable

(10 pts.) Show the layout of your multiplier with the registers.

Final Product: ✓



*note: bounding box area **fits** within the 900x900 constraint*

(10 pts.) Explain the design and functionality of your multiplier.

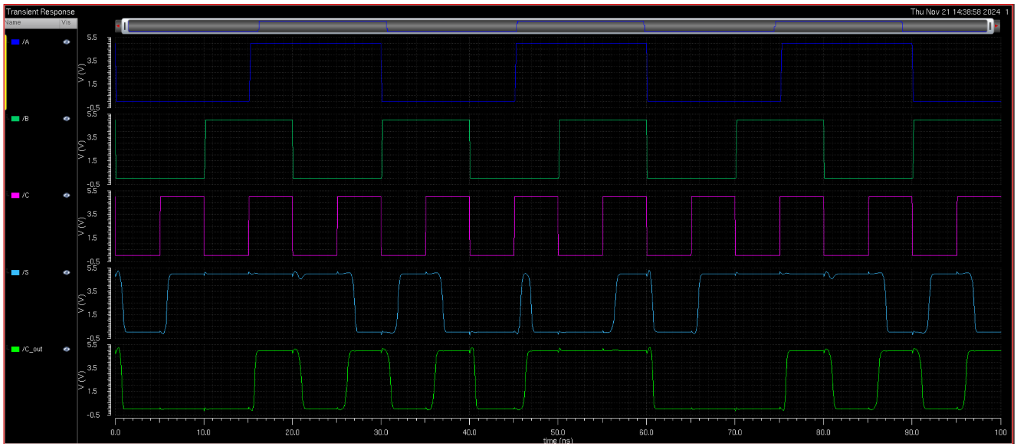
Our 4x4 bit-sliced array multiplier shown in the images of this pdf report were our hardware layout implementation of binary multiplication. It takes two 4-bit binary numbers, X (x3x2x1x0) and Y (y3y2y1y0), as inputs and produces an 8-bit product as the output, using SIPO input registers and PISO output registers. The design creates partial products for each bit of X multiplied by each bit of Y. These partial products are then summed using Carry Save Adder.

The inputs are taken in serially from X and Y pins in the design. In the input of Y, there is a MUX that determines whether the design is in test mode or normal mode. If in normal mode, the X and Y take a serial in input, make it into a parallel output, and input those values into the multiplier. The result of the multiplication is the input into a parallel in serial out shift register, where the parallel inputs is the 8-bit results, and the output is the 8 serial bits. Before each shift register, there is a also a MUX that uses the same TEST signal to determine whether the output registers should take the inputs from the multiplier, or input from the input registers X and Y. Once in test mode, the X register takes a serial input, the last register in X is connected the first in Y, and the last in Y is connected to the first of the output register. This makes a large 16-bit shift register that is tested to ensure the full functionality of all of the registers.

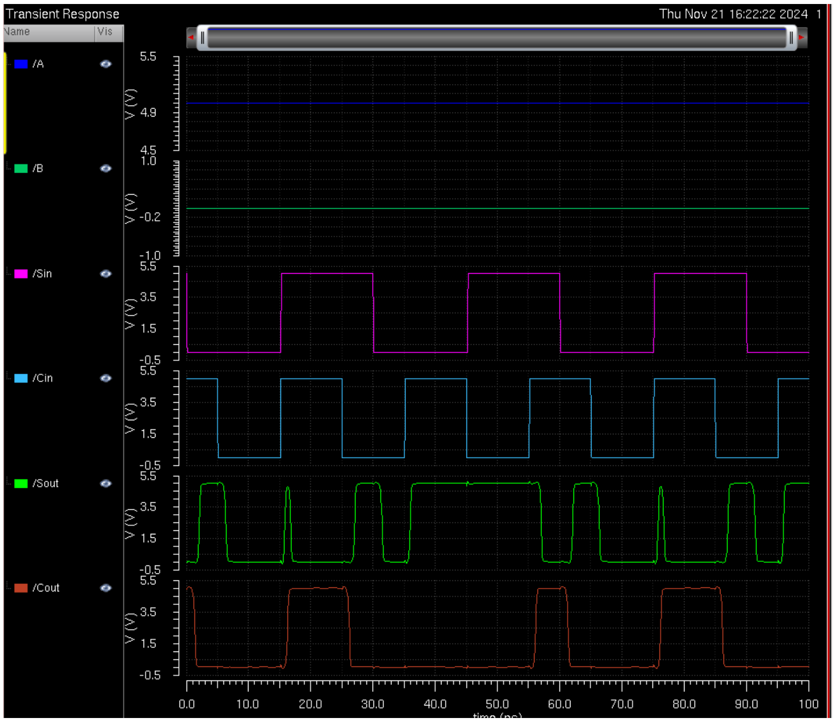
(20pts) Simulation Results:

(10 pts total) Individual cells:

Full Adder: ✓

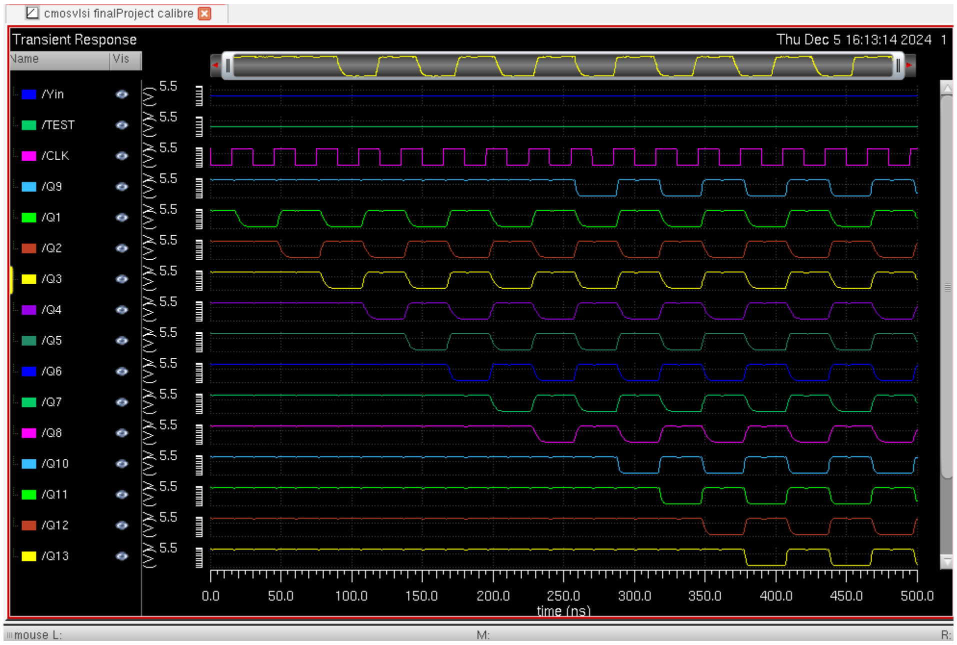


Full Adder with AND gate: ✓

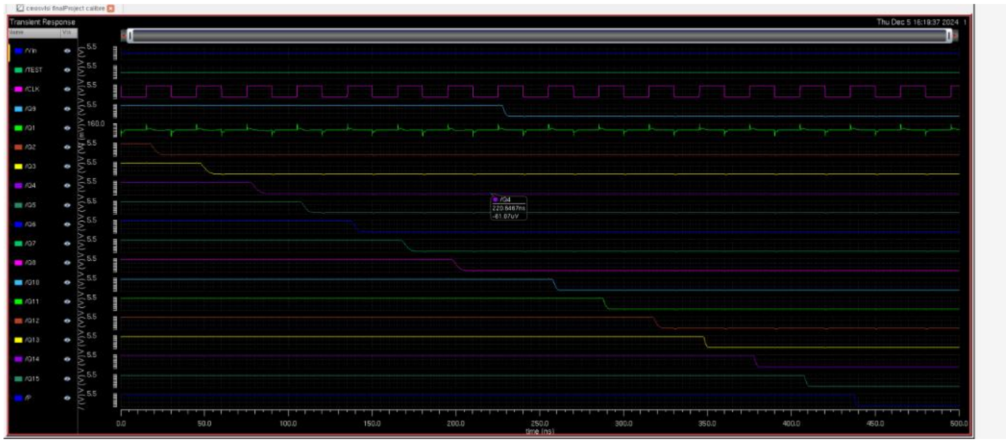


Registers (Test Mode):

11111111 ✓



00000000 ✓



Ring Oscillator (If used):

N/A

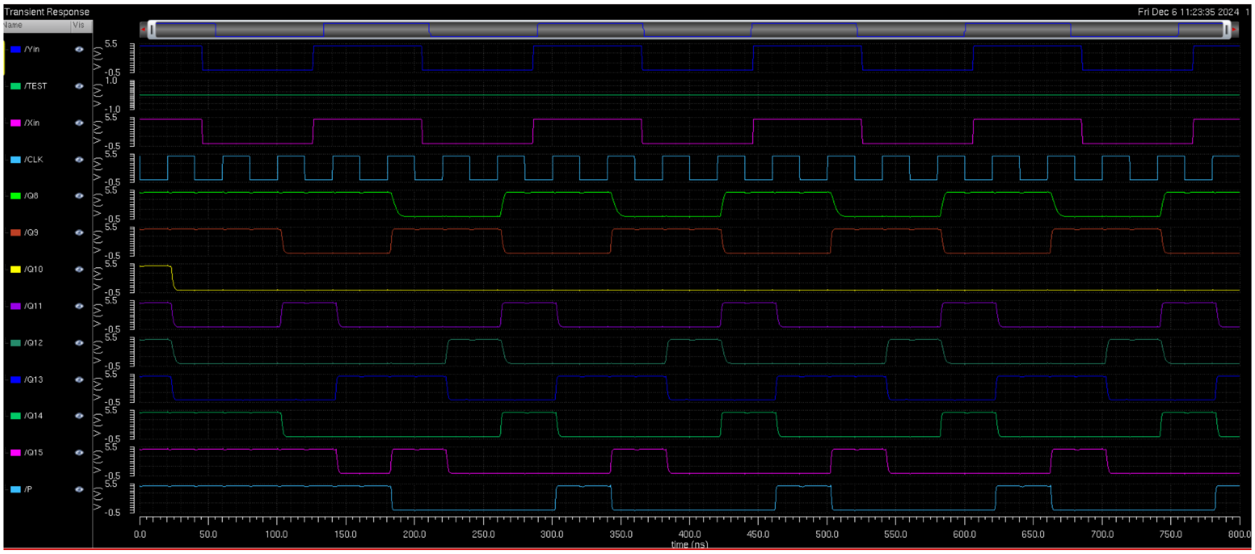
(10 pts) The final multiplier in test and normal modes:

8 Test vectors (Normal modes)

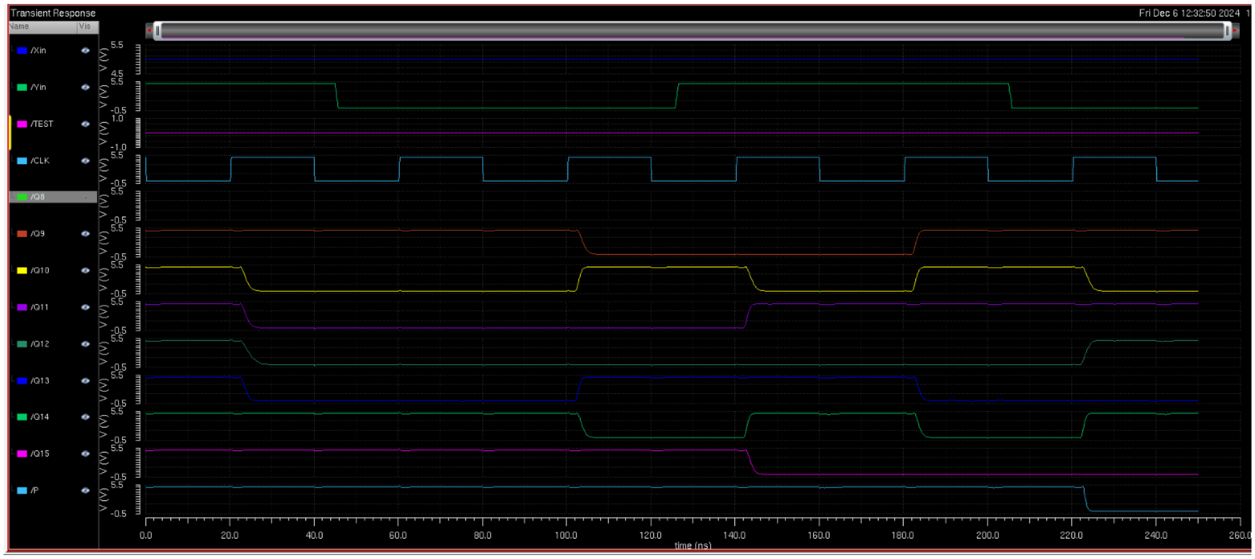
Note: Outputs are read from Q16 (labeled P) to Q9.

Q8 was an input bit that was left on when plotting waveforms and can be disregarded in the following screenshots.

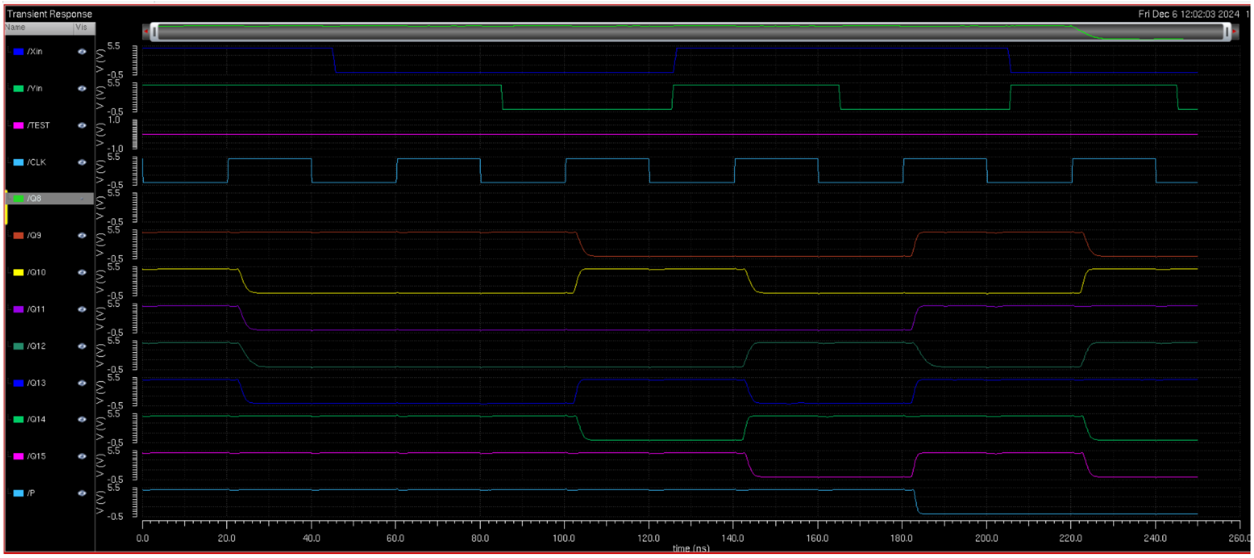
Test vector 1) 1001*1001 = 0101 0001



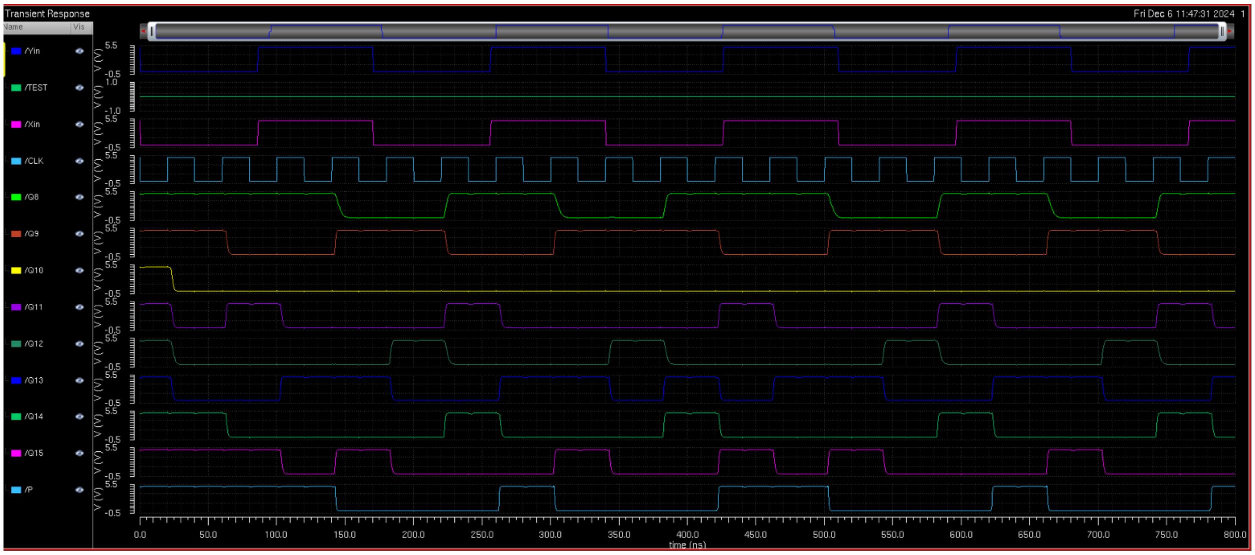
Test vector 2) 1111* 0011 = 0010 1101



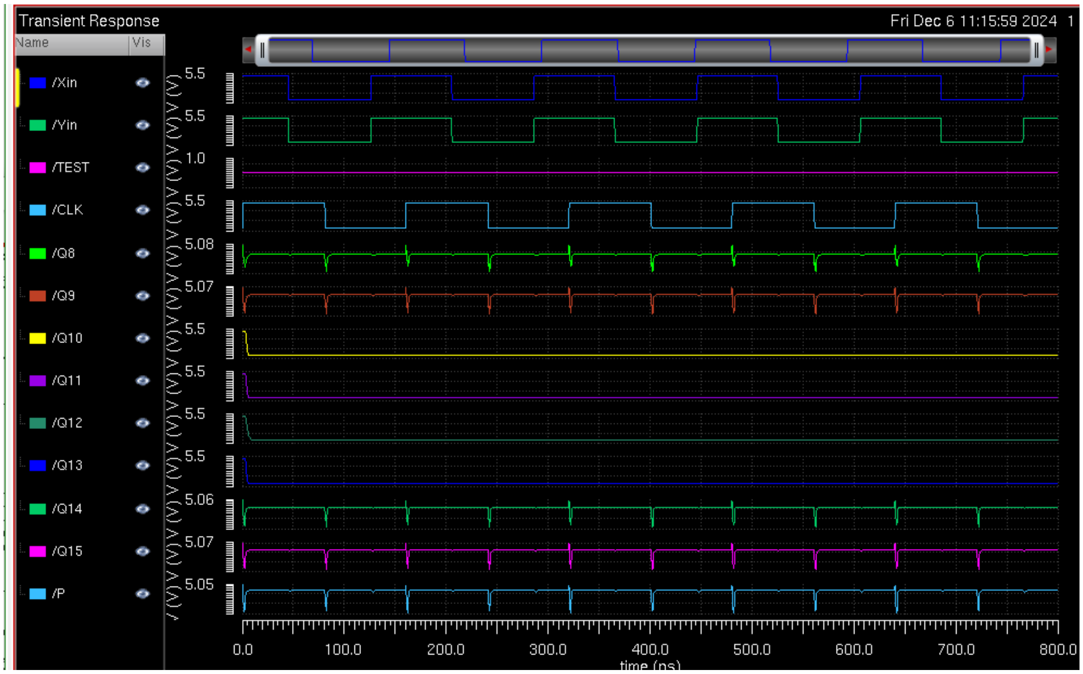
Test vector 3) 1001 * 1011 = 0111 0101



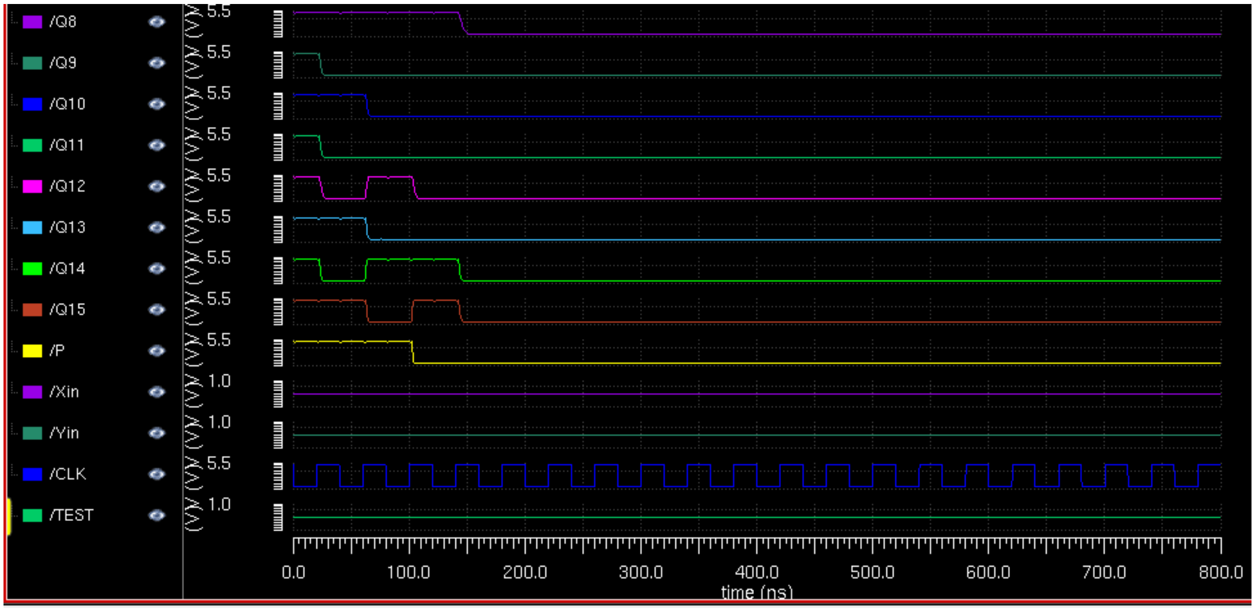
Test vector 4) 0011 * 0011 = 0000 1001



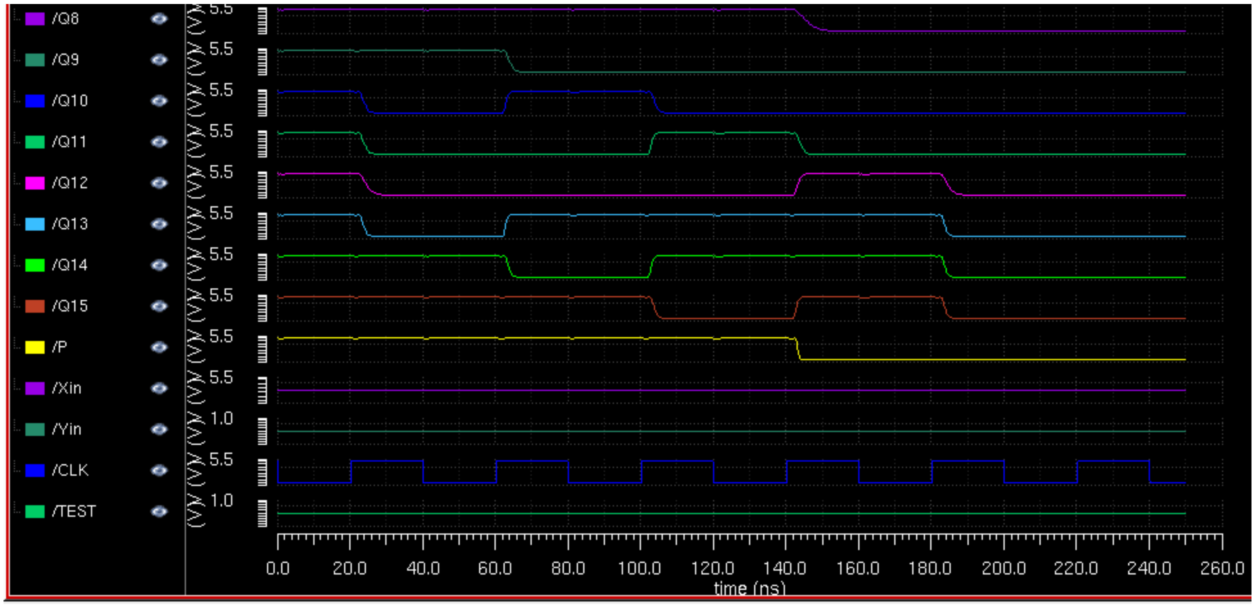
Test vector 5) 1111 x 1111 = 1110 0001



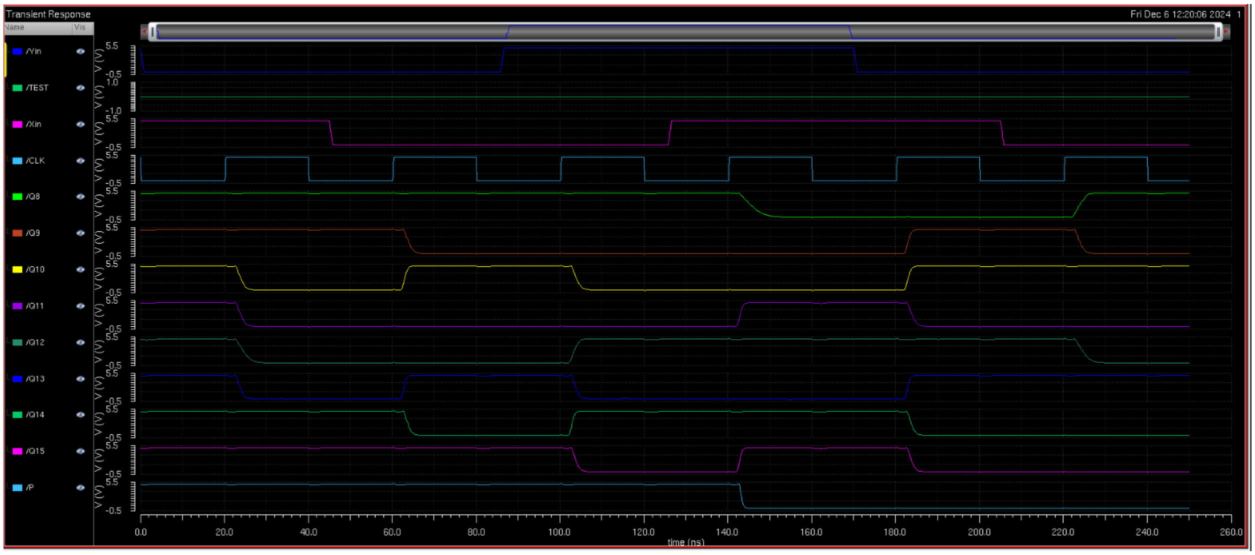
Test vector 6) 0000*0000 = 0000 0000



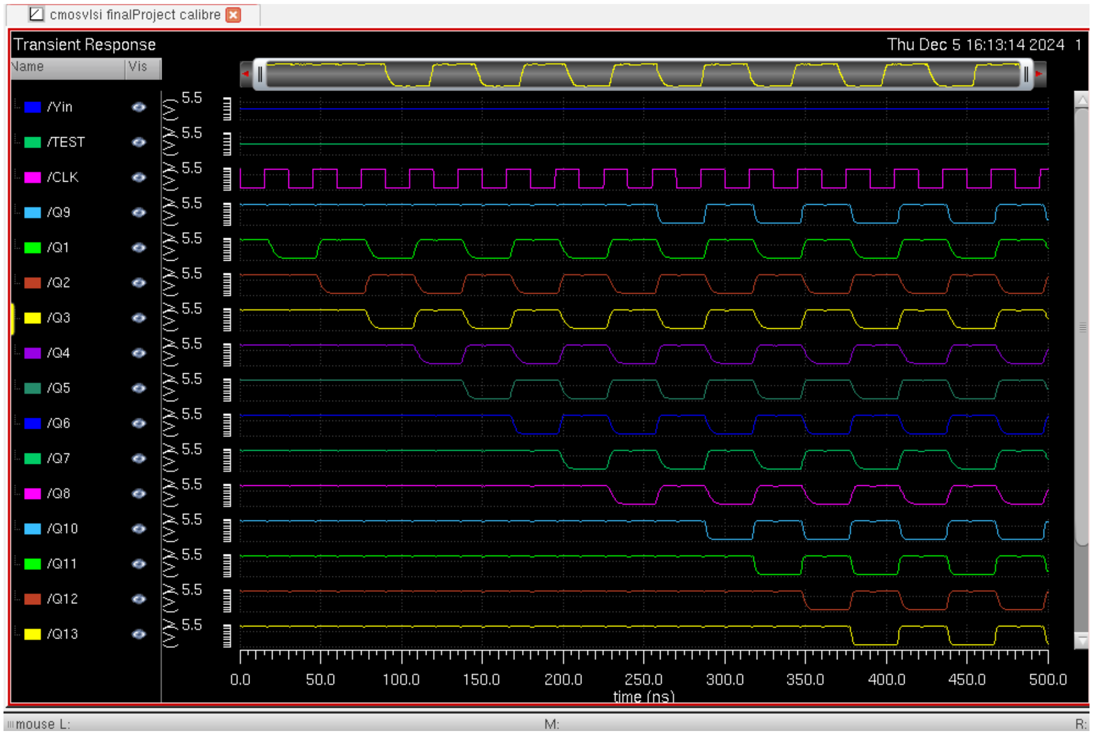
Test vector 7) 1111*0000 = 0000 0000



Test vector 8) 1001 * 0011 = 0001 1011



Test Mode (TEST=High) ✓



above image (TEST=1) shows parallel shift through registers, creating the honeycomb-like waveform output.