EEE/INSTR F313
Analog and Digital VLSI Design

8-bit parity checker

8-Bit Parity checker at 1GHz with load capacitance of 1pF

NAME OF STUDENT	I.D. NUMBER
Hariharan Venkat	2019A3PS0244P
Sashank Krishna S	2019A8PS0184P
Anirudh Subramanian	2019A3PS0274P

Submitted to Dr. Anu Gupta

INDEX

Introduction

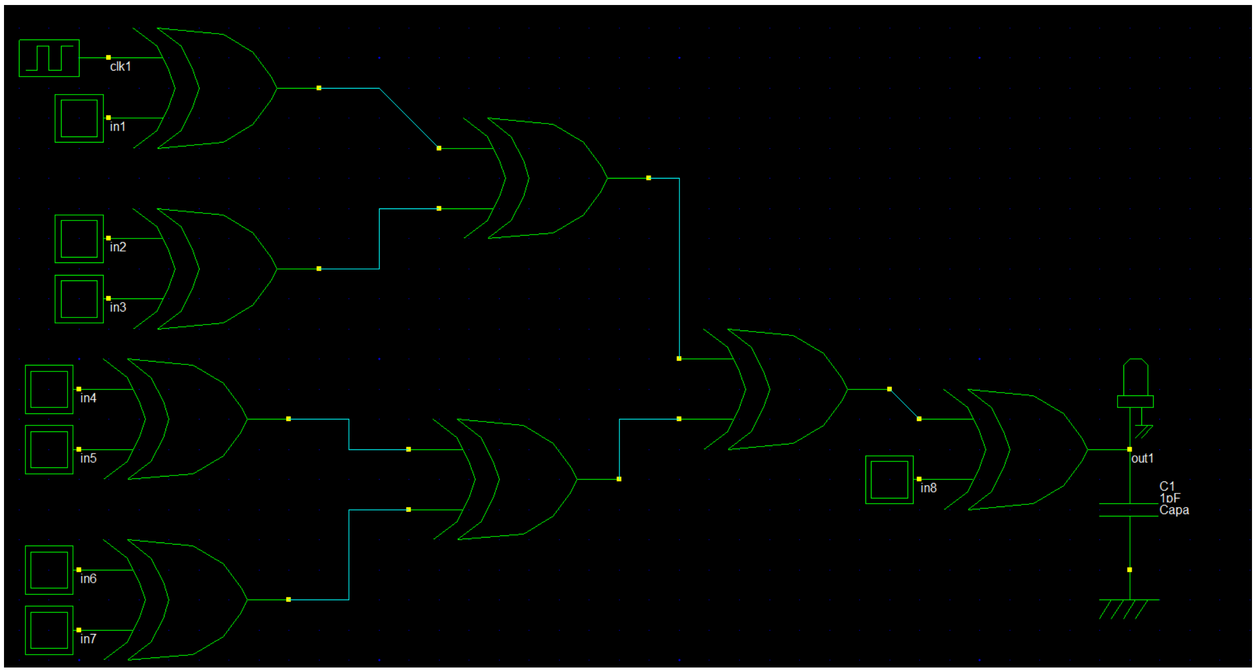
In digital communication , a parity bit is added to the word of data as a means of detecting random bit flips due to noise. The redundancy allows the receiver to check if everything tallies, and gauge whether or not an error occurred. The receiver counts the number of highs (1s) in the transmitted signal and if it does not equal the parity bit , there is an error in the transmission. Similarly, parity-based techniques are also widely employed in memory technology as well. For implementing such a scheme, hardware is required. The problem statement addressed in this section, is the design of such a circuit.

There are many ways to implement this . The XOR operation intrinsically performs a parity check. Hence, we usually use XOR gates for creating an 8-bit parity checker. Due to the nature of XOR gates there are multiple ways in which such a parity checker can be implemented , depending on the flexibility an XOR gate provides - such as varying the number of inputs of the gate, the number of levels, and the manner in which the gate itself is implemented.

Hence, we decoupled the problem statement as follows:

1. Selecting the number, and fan in of the XOR gates, and designing the circuit using the same
2. Deciding which XOR gate topology to use
3. Designing the unit inverter, the components of the XOR gate, and finally the XOR gate itself
4. Designing the full circuit, and translating it into a layout.

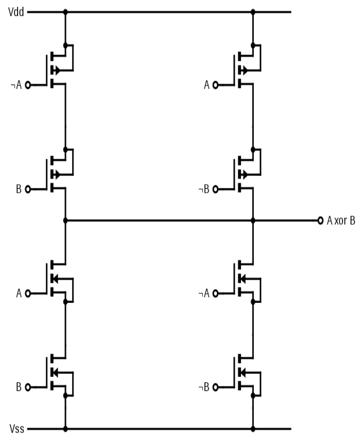
Step 1: Deciding number and fan in of XOR gates
Increasing the number of inputs of an XOR gate exponentially increases the number of transistors required. Even moving from 2 inputs to 3 would raise the number to more than double the original number. Hence we opted for the design shown below as it is simple and quite well optimized, and hence, serves our purpose.



An 8-bit parity checker (Implemented in DSCH)

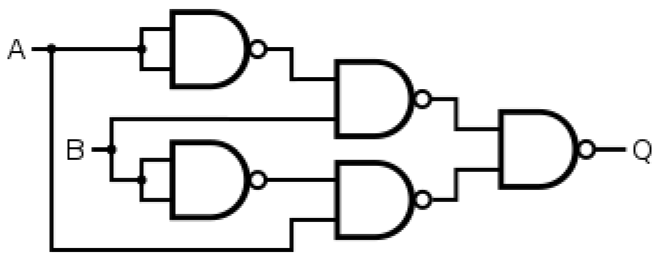
Step 2: XOR gate topology selection

Candidate 1: The typical implementation (12 transistors)



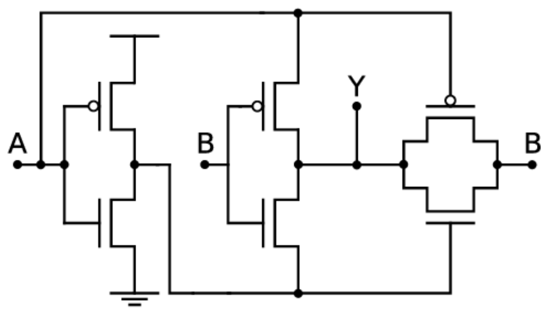
This is the naive implementation of the XOR gate. As complement inputs are not assumed to be at our disposal, we accounted for complementing them using a standard CMOS inverter (2 transistors per input) and calculated a total of 12 transistors. The implementation had a higher output capacitance at the output node as there are 4 transistors connected directly to it.

Candidate 2: NAND Implementation (20 transistors)



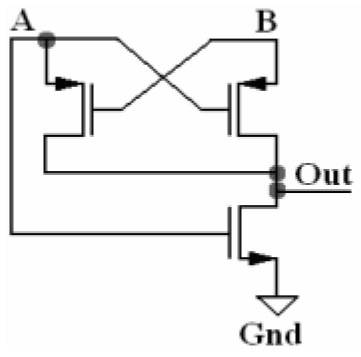
This is another comparatively naive implementation with 16 transistors. The two NAND gates in the first stage are performing an inverter's functions. This circuit is relatively easier to implement as compared to the previous one and has a lesser output capacitance.

Candidate 3: PTL Implementation (6 transistors)



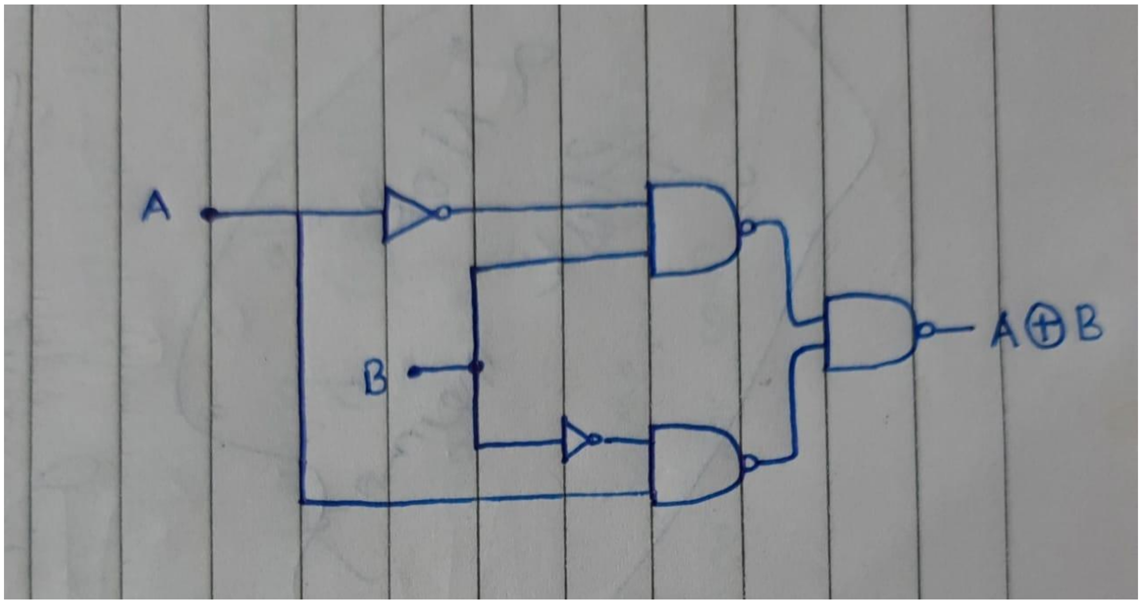
This implementation allows for a drastic reduction in the number of transistors required at each stage at the cost of deterioration of the output voltage levels. However, since we could not figure out how to compute the logical effort for such topologies, we were unable to optimize this method properly.

Candidate 4: 3 Transistor Implementation



This implementation was found during literature review. It consists of just 3 transistors and reduces our transistor count manyfold. However, the simple implementation of the circuit does not properly act as an XOR gate. The W values needed to be fine-tuned to obtain the needed XOR functionality**[CHECK THE REFERENCE PAGE NO.]**, we were unable to get the topology to work, and hence did not use this method.

Candidate 5: Modified NAND implementation (16 transistors)



This is similar to the candidate 3 topology but the first stage NAND gates are replaced with inverters which allows for a reduced transistor number while maintaining the simplicity of the topology. It was this topology which was chosen for this project.

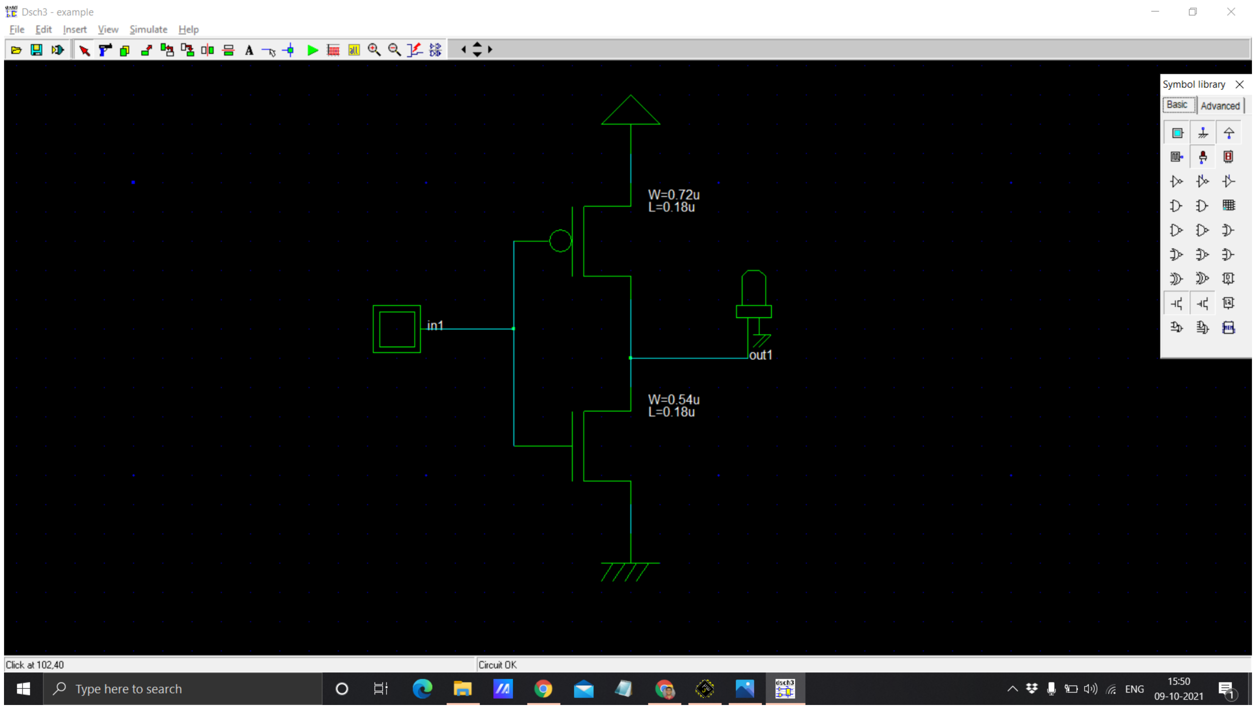
Gate Level Circuit

As mentioned above, the plan for creating the 8-bit parity checker is straightforward. We create the XOR gate and then cascade them as shown above to achieve the required functionality. Our first step of creating the XOR gate involves using a NAND gate and an inverter as well. As mentioned above, candidate topology 5 **(INSERT FIGURE NO)** was chosen for this project. We will now see the design of the inverter, the nand gate and then the XOR gate, mentioning the caveats and the thought processes employed along the way.

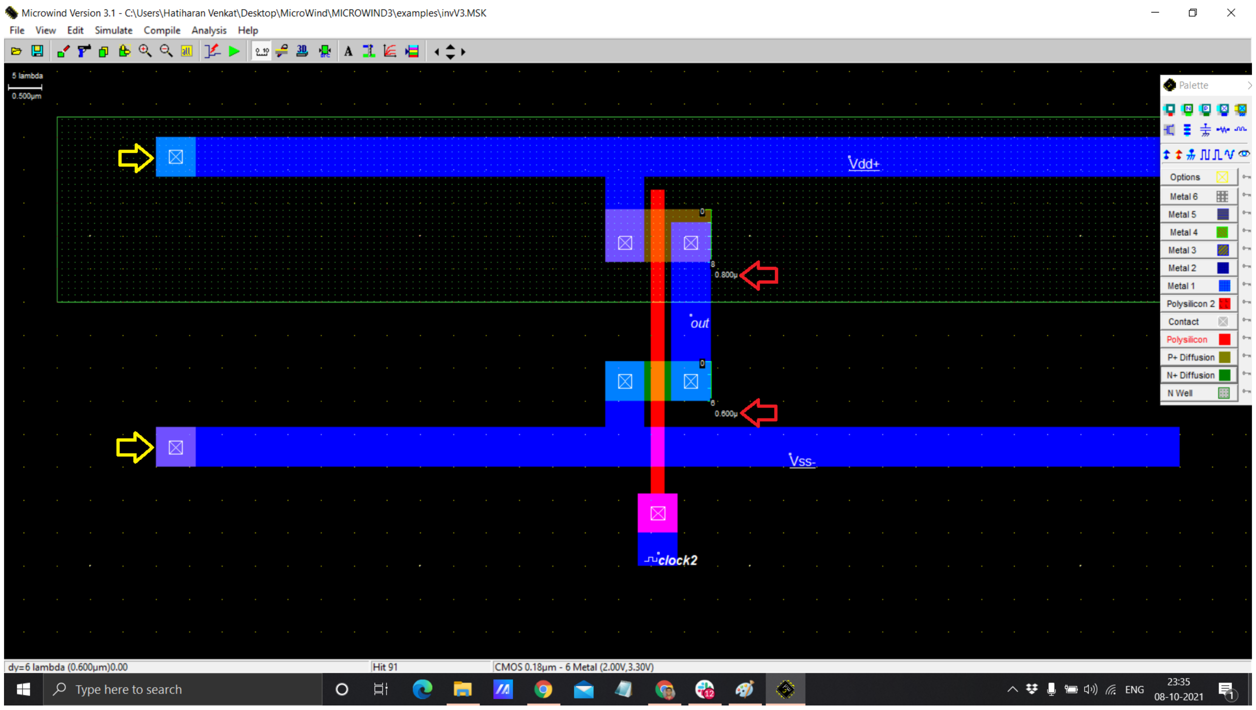
1. Inverter Design

The first step in the design was to design an inverter of equal charge and discharge cycles. Since our technology node was fixed at 180nm ,the lengths of both transistors were taken as Lmin while the widths were varied (in accordance to the standard design rules). Equality of charge and discharge cycles was determined by examining the simulations and the difference between the tplh and tphl values.

After trial and error and checking many different W values, the closest to an optimal solution came at Wn = 6λ and Wp = 8λ. The below images show the CMOS Inverter design at the transistor level and the layout levels. One important thing we learnt from making this layout was the importance of grounding the N and P wells as well. We have highlighted those contacts, as well as the measured Wp and Wn values in the layout diagram given below. Since λ = 90nm, Wp = 8λ = 720 nm and Wn = 540 nm.



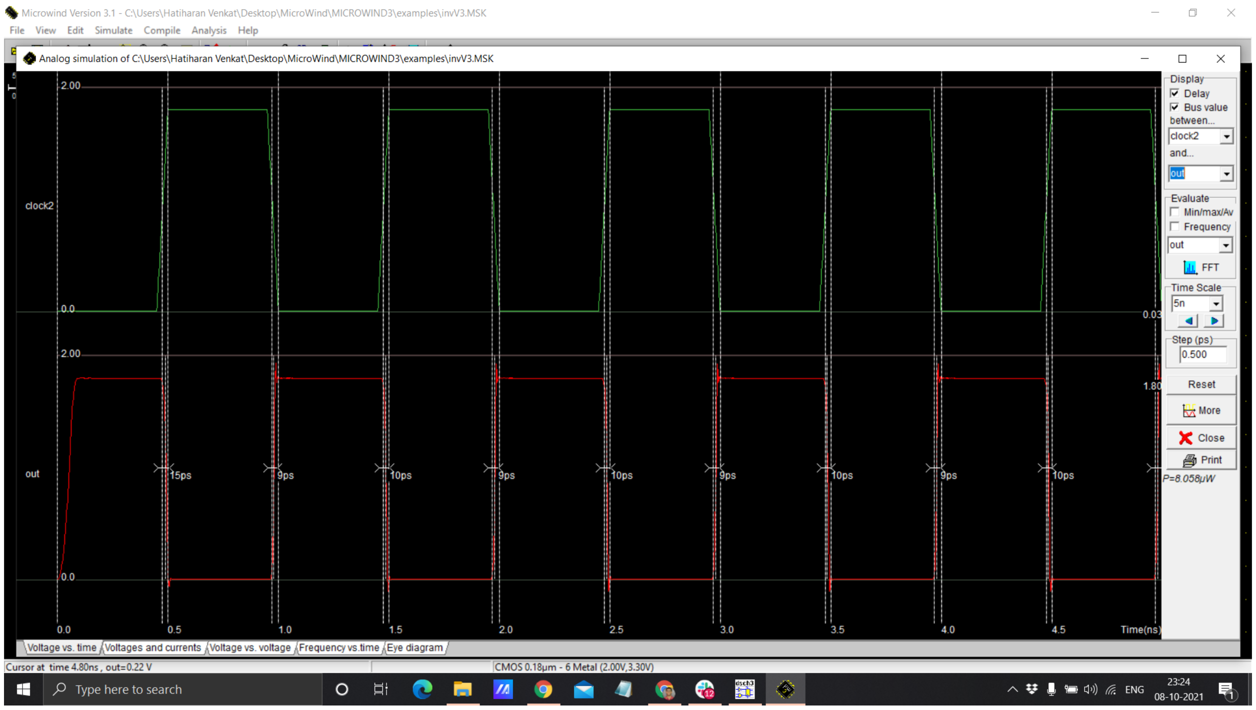
CMOS implementation of a Inverter in DSCH



Inverter Layout (Note the Well to ground contacts and the Wn and Wp values)

On using the above mentioned Wn and Wp values, we simulated the circuit on MicroWind to test the results. We observed a tphl = 10 ps and tplh = 9 ps on no load. This was however, the most optimal result we could achieve as getting them exactly equal was not possible. The graph below summarizes the results.

Later, we went on to realize that not using a load here was a major blunder.


Microwind Version 3.1 - C:\Users\Hatiharan Venkat\Desktop\MicroWind\MICROWIND3\examples\invV3.MSK
Analog simulation of C:\Users\Hatiharan Venkat\Desktop\MicroWind\MICROWIND3\examples\invV3.MSK
clock2
out
15ps
9ps
10ps
Time(ns)
Delay
Bus value
FFT
Reset
More
Close
Print
P=8.058µW
Voltage vs. time
Voltages and currents
Voltage vs. voltage
Frequency vs.time
Eye diagram

Cursor at time 4.80ns , out=0.22 V

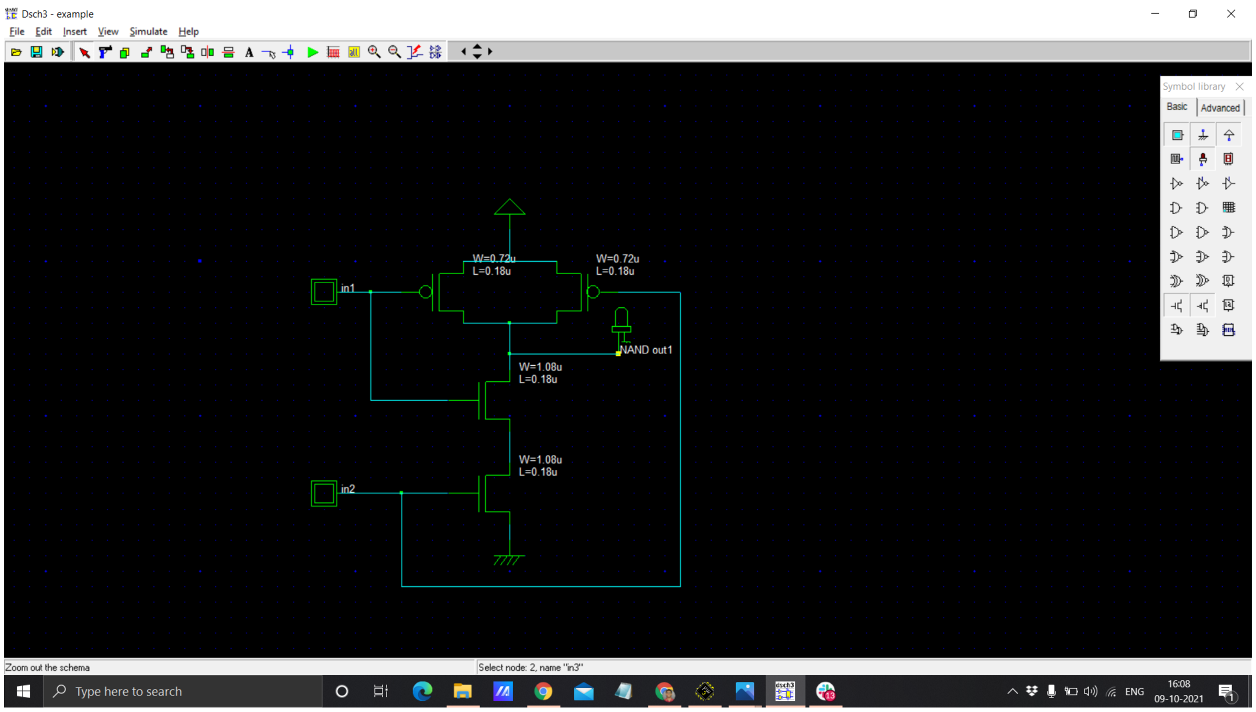
CMOS 0.18µm - 6 Metal (2.00V,3.30V)

Inverter Output v/s time graph of input clock (green) and output waveform (red) illustrating the rise and fall times (MicroWind)

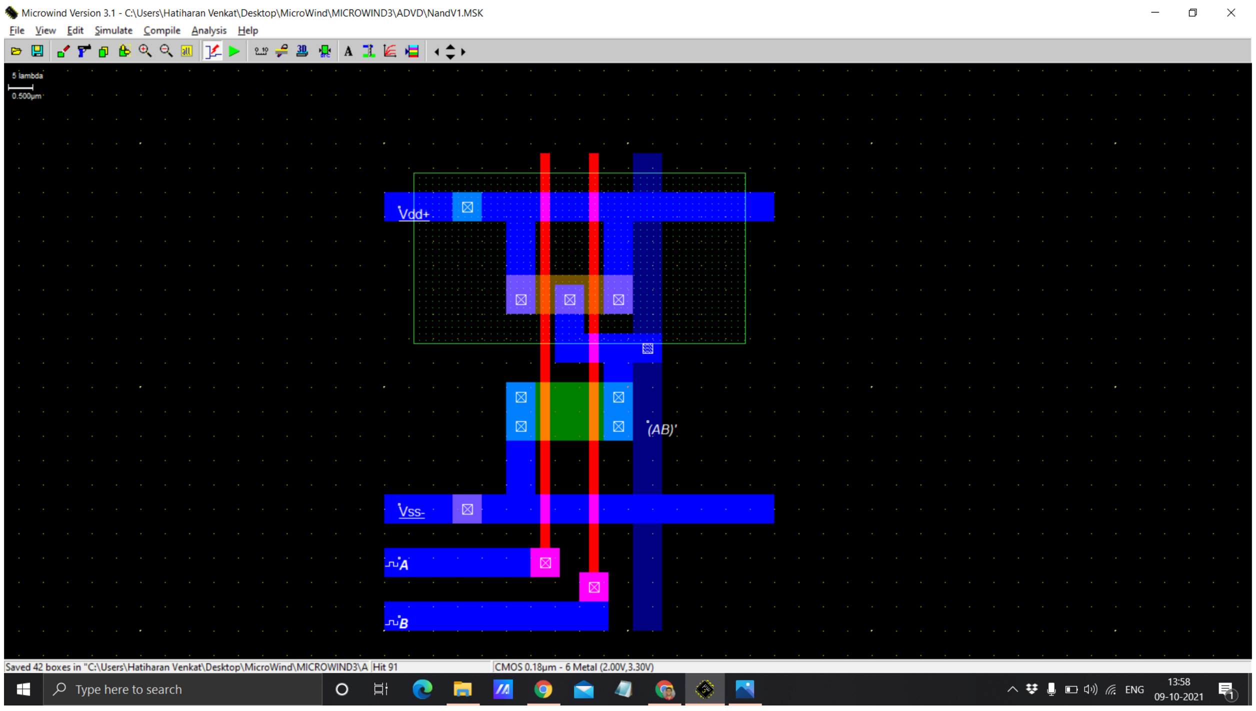
With that, we concluded the design of our unit inverter. Note that the inverter was designed using the level 3 mosfet model, since microwind 3 did not allow us to use the BSIM4 model. At a later stage, we had switched to using microwind 2, when we heard that it had superior features. Time did not permit us to switch to the BSIM4 model, since it would require the redesigning of the inverter, as well as the NAND and XOR that was designed at that point.

2. NAND Design

The next step we took towards the completion of our assignment was the design of the nand gate. We used the typical CMOS logic to design the NAND with two PMOS transistors in parallel connected to 2 series NMOS transistors. The Wp and Wn values were selected so as to retain the same proportions as that of the unit inverter designed earlier.



CMOS implementation of NAND Gate (DSCH)

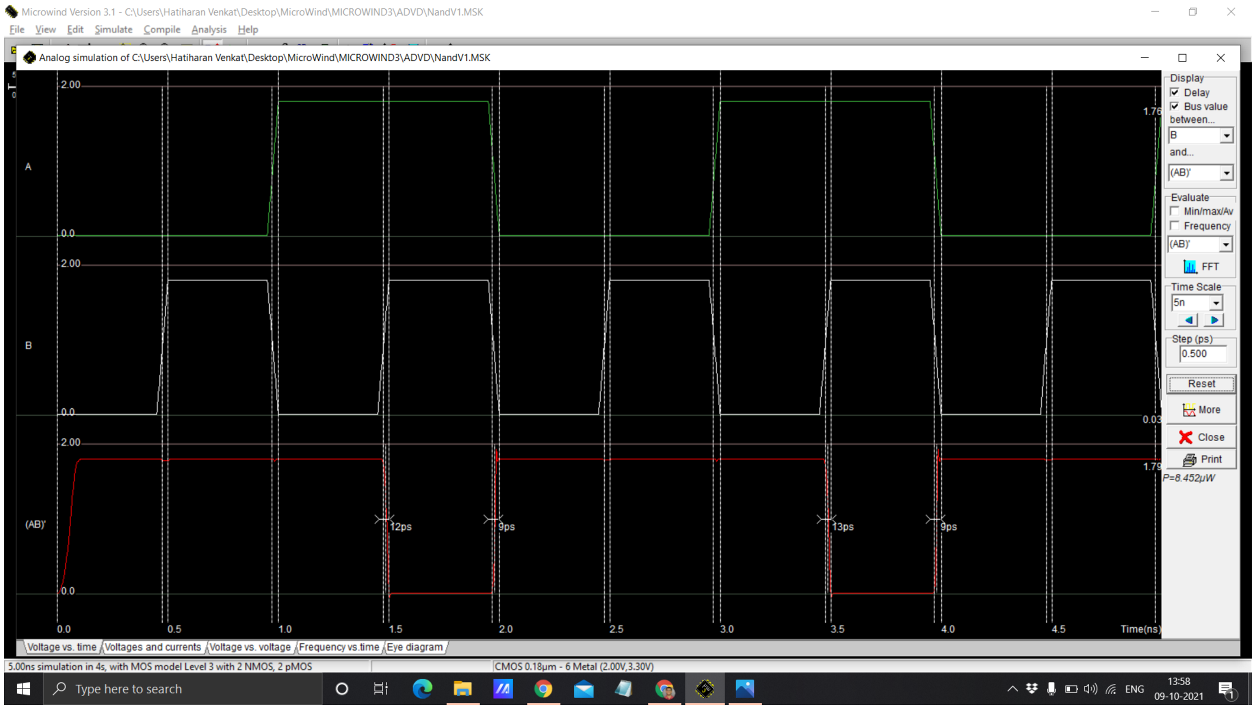


Microwind Version 3.1 - C:\Users\Hatiharan Venkat\Desktop\MicroWind\MICROWIND3\ADVD\NandV1.MSK

Saved 42 boxes in "C:\Users\Hatiharan Venkat\Desktop\MicroWind\MICROWIND3\A Hit 91

CMOS 0.18µm - 6 Metal (2.00V,3.30V)

NAND Layout



Microwind Version 3.1 - C:\Users\Hatiharan Venkat\Desktop\MicroWind\MICROWIND3\ADVD\NandV1.MSK

Analog simulation of C:\Users\Hatiharan Venkat\Desktop\MicroWind\MICROWIND3\ADVD\NandV1.MSK

5.00ns simulation in 4s, with MOS model Level 3 with 2 NMOS, 2 pMOS

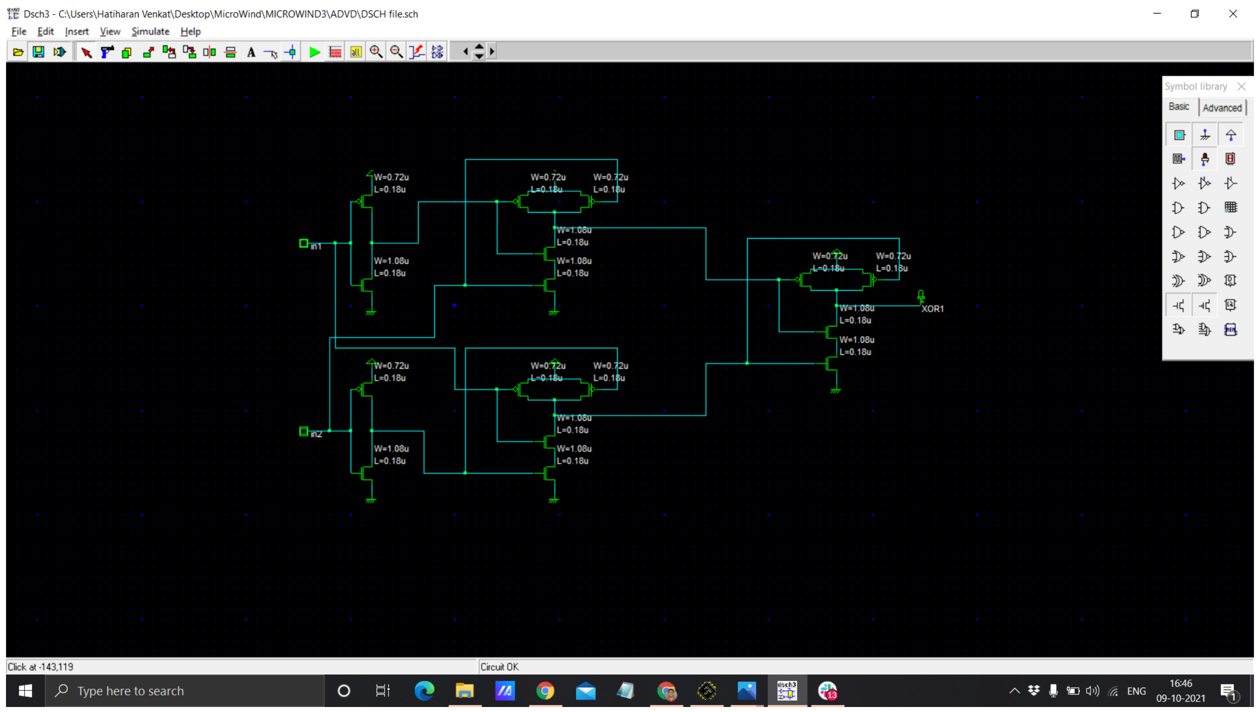
CMOS 0.18µm - 6 Metal (2.00V,3.30V)

NAND Inputs A & B (green and white) and Output waveform (AB)' (red) v/s time

3. XOR Design

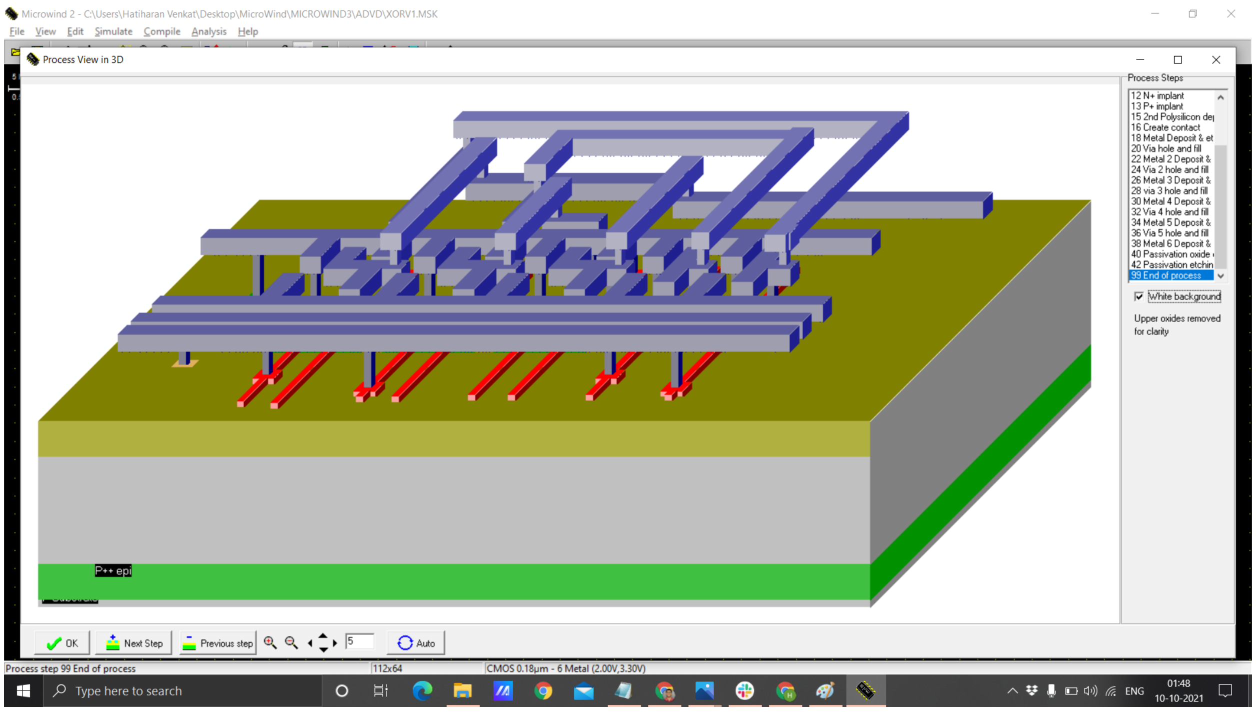
Now that the inverter and NAND gate have been implemented, we can now put them together as shown in figure **<u>(NAME OF FIGURE).</u>** Once again, the Wp and Wn values of each transistor have been kept as per those inferred from the inverter design.

Before we proceed further, however , we must elaborate upon the various other XOR layouts that we considered, and how we finally arrived at the below model that we used in the final design-

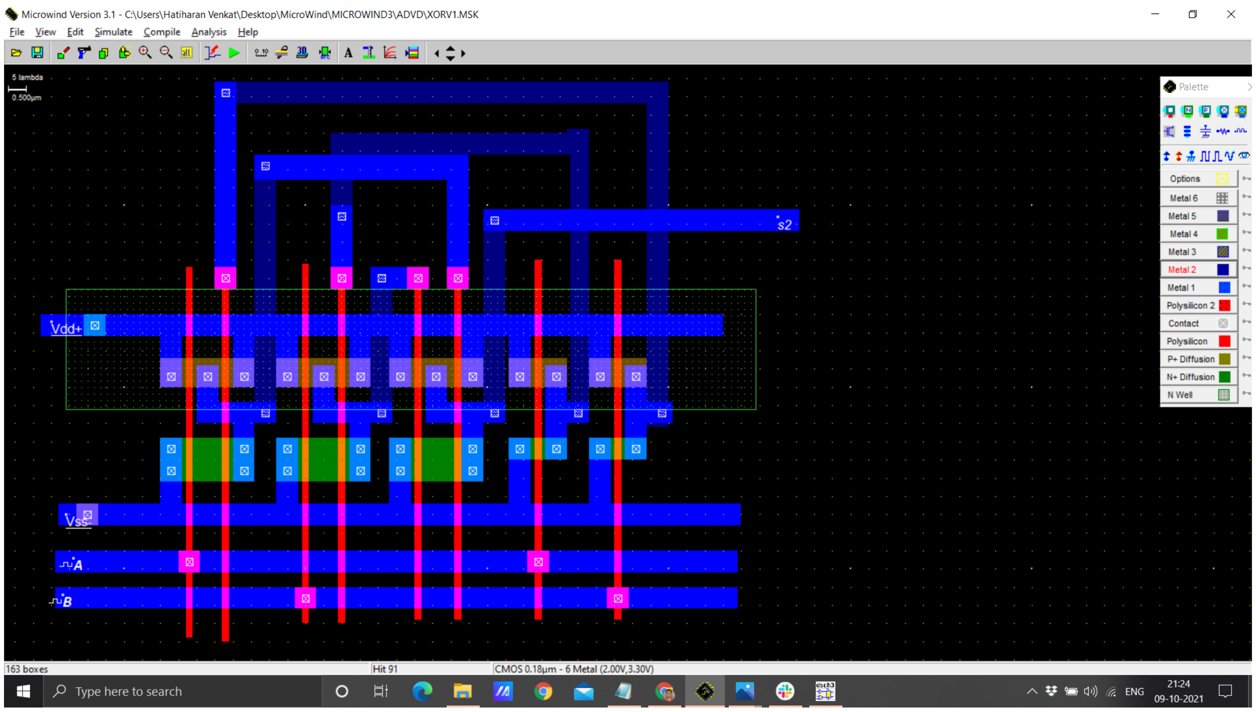


XOR in CMOS Logic (DSCH)

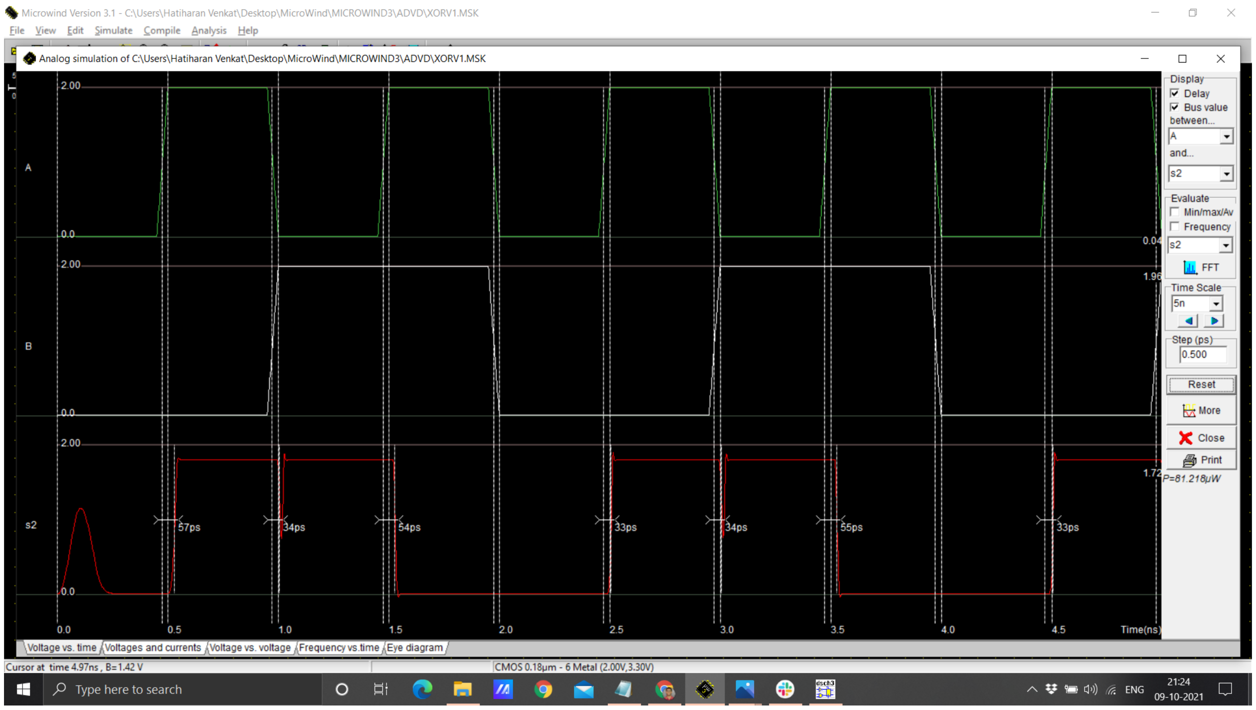
We chose the XOR design with the intent of minimizing the number of transistors while still retaining it's functionality . Below are the configurations we considered -



Process View of XOR in 3d (Microwind)



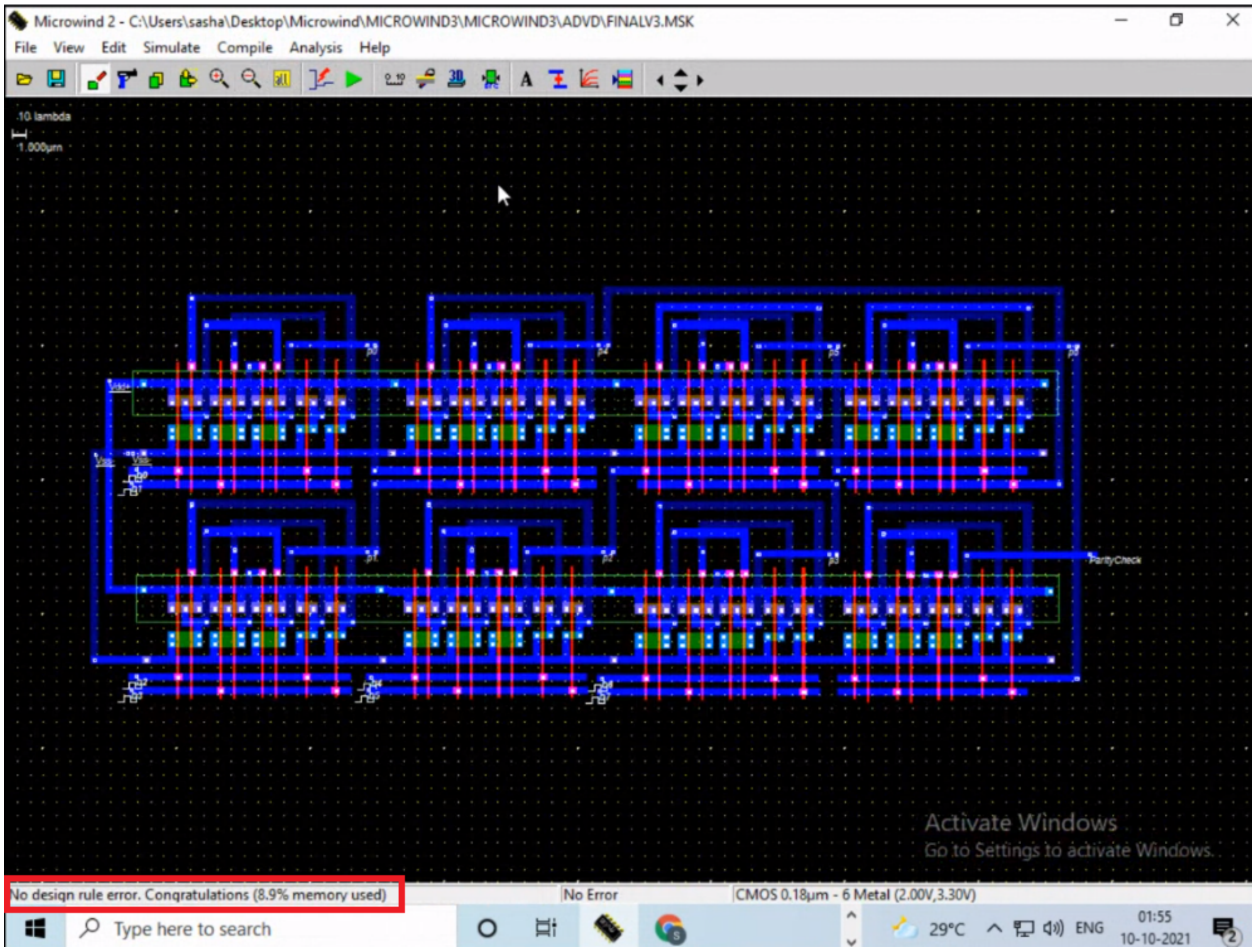
XOR Layout



Inputs (Green and White) and Output (Red) v/s time

4. Parity Checker Design

Once the individual XOR gate was designed, all that was left was for us to cascade the gates as shown in fig **[INSERT FIGNO].** Figure **[INSERT FIG NO]** shows the final layout of the 8 bit parity checker we created.


Microwind 2 - C:\Users\sasha\Desktop\Microwind\MICROWIND3\MICROWIND3\ADVD\FINALV3.MSK
File View Edit Simulate Compile Analysis Help
10 lambda
1.000µm
PartyCheck
Activate Windows
Go to Settings to activate Windows.
Type here to search
29°C ENG 01:55 10-10-2021

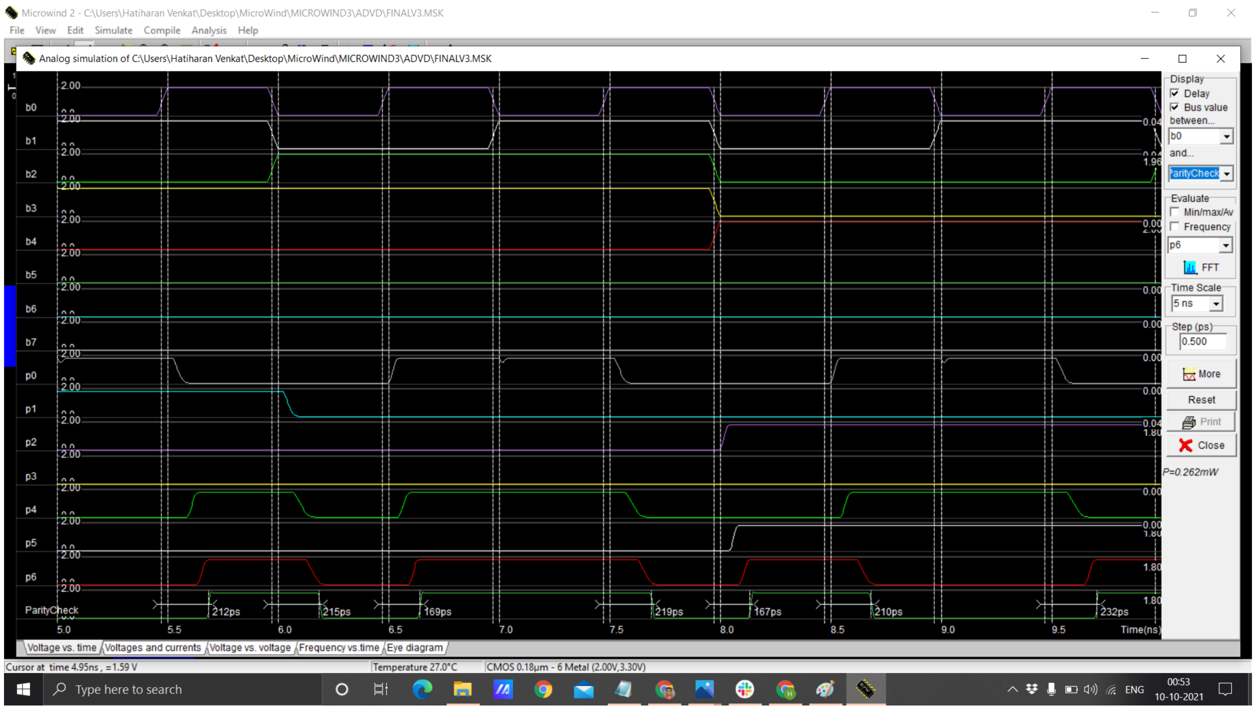
No design rule error. Congratulations (8.9% memory used)

No Error

CMOS 0.18µm - 6 Metal (2.00V,3.30V)

FInal layout of the 8 bit parity checker. (Design rule verification highlighted in red)

We then ran the simulations to observe the delays involved in the process. Critical to note here is the effect of the load capacitance. Our circuit was not able to drive the 1 pF load capacitance given in the question. We found that 0.12 pF was the maximum allowable load capacitance for valid output.


Microwind 2 - C:\Users\Hatiharan Venkat\Desktop\MicroWind\MICROWIND3\ADVD\FINALV3.MSK

Analog simulation of C:\Users\Hatiharan Venkat\Desktop\MicroWind\MICROWIND3\ADVD\FINALV3.MSK

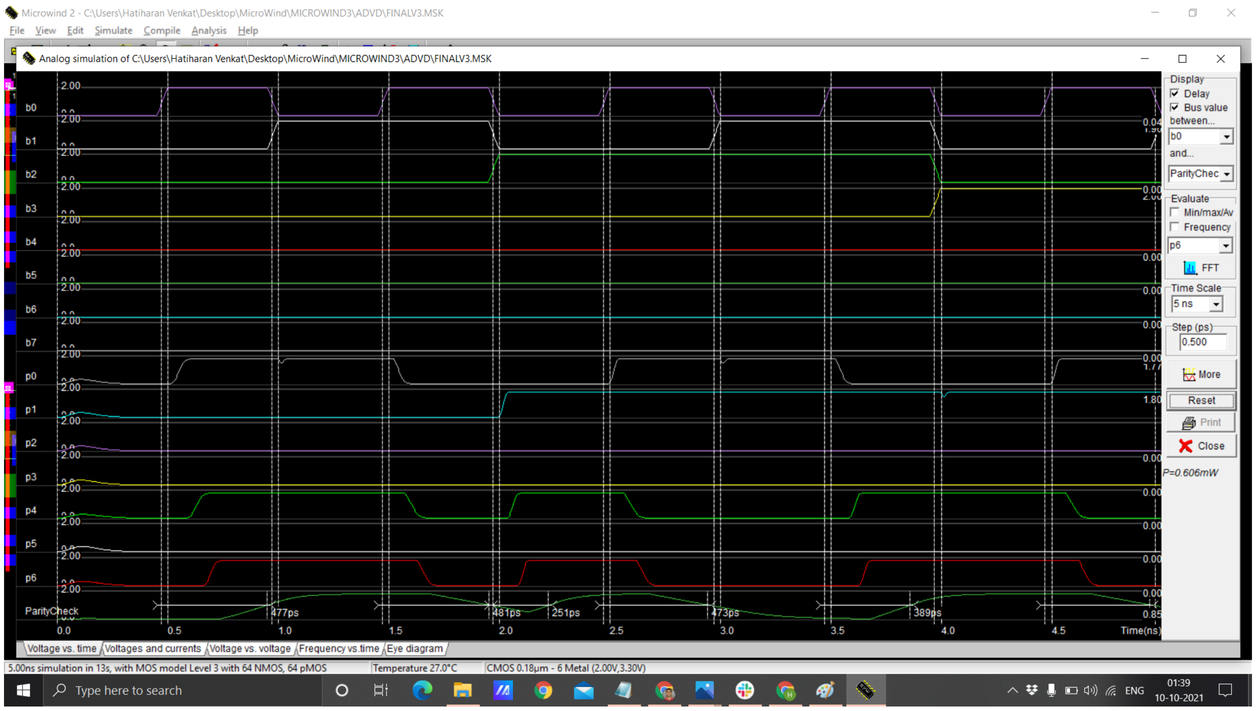
Cursor at time 4.95ns , =1.59 V

Temperature 27.0°C

CMOS 0.18µm - 6 Metal (2.00V,3.30V)

P=0.262mW

Output Voltage vs Time for the 8 bit parity checker (Load Capacitance = 0)


Microwind 2 - C:\Users\Hatiharan Venkat\Desktop\MicroWind\MICROWIND3\ADVD\FINALV3.MSK

Analog simulation of C:\Users\Hatiharan Venkat\Desktop\MicroWind\MICROWIND3\ADVD\FINALV3.MSK

5.00ns simulation in 13s, with MOS model Level 3 with 64 NMOS, 64 pMOS

Temperature 27.0°C

CMOS 0.18µm - 6 Metal (2.00V,3.30V)

P=0.606mW

Output Voltage vs TIme for the 8 bit parity checker (Load Capacitance = 0.12 pF)

Design Specifications

Given below are the specifications of the design submitted. The specifications of the inverter, the NAND gate, and the logical effort calculations for the failed design can be found in the appendix.

Challenges faced

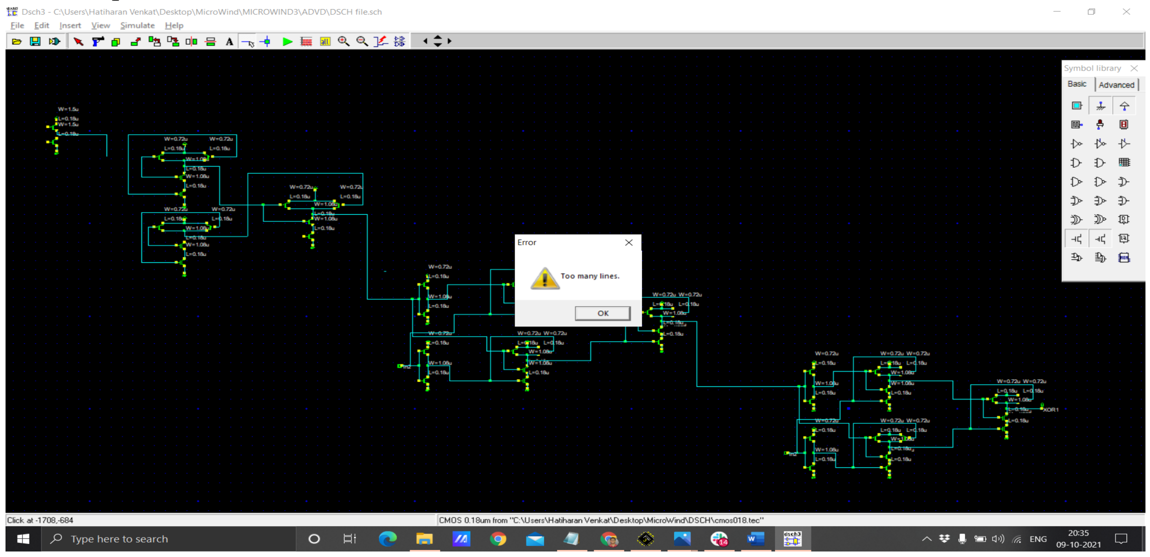
In this section, we elucidate the various design challenges encountered when completing this assignment. While these hurdles seemed too hard to overcome at the start, it allowed us to find better and more efficient solutions.

1. **Extraction of the capacitances of the unit inverter circuit designed on Microwind:** On using the electrical node analyzer feature that microwind provided us with, a number in low fermiFarads. For this reason, to drive a 1pF load, the logical method approach required us to use gates that are sized 243 times larger than a unit inverter. The width of the transistor required approached 0.5mm, which is an extraordinarily high value for a single 180 nm node transistor. Since we were unable to ascertain the source of this error, and since we did not understand exactly which value microwind provided us with, we had to alter the capacitive load, and implement a circuit with all the XOR gates used being identical.

2. **Driving the 1pF load:** Due to the extreme results we obtained from our logical effort calculations, we were unable to realize the optimal version of our circuit. We realized too late that had we sized our unit inverter larger, our logical effort calculations would have gotten us acceptable transistor sizes.

3. **Fitting the 6 transistor XOR into the logical effort model:** When attempting to implement the aforementioned PTL-based XOR circuit, we were unable to describe the same properly using the logical effort scheme. Since a transmission gate was used, the inputs were also fed into the source/drain of the transistors, and we were unfamiliar with how to model the same using the logical effort scheme.

4. **Simulating large circuits on DSCH:** When we tried to simulate our final 8 bit parity checker on DSCH, the software was unable to handle the complexity of the circuit. Initially, we tried using the CMOS implementation, where 1 XOR gate used 16 transistors. Using the 8 XORs to achieve the design we planned involved 128 transistors. When creating the circuit on DSCH, an error "Too many lines" was thrown and hampered us from proceeding further. Even when we tried to switch to the PTL-based XOR implementation which consisted of 6 transistors per XOR, the same problem was encountered.


Dsch3 - C:\Users\Hatiharan Venkat\Desktop\MicroWind\MICROWIND3\ADVD\DSCH file.sch

Error

Too many lines.

OK

Click at -1708,-684

CMOS 0.18um from "C:\Users\Hatiharan Venkat\Desktop\MicroWind\DSCH\cmos018.tec"

A screenshot showing the error message displayed by the DSCH software

Despite our best attempts at self-studying, an overwhelming number of slip-ups and unknowns survived till the last minute, and could not be addressed in our designs. However, we have listed the same throughout this report. The biggest constraint faced was hence the design time limitation.

Innovations:

1. **Layout:**
    a. Multi-fingered transistors were used in place of a series connection of 2 transistors.
    b. The 4 stages were packaged into a single rectangle by rearranging the stages efficiently.
    c. The routing was done such that only 2 metal layers were required.

Problem 2:

The verilog code for the module designed:

```
module flipregister ( pos, en, clk, op );
        input   [0:2] pos;
        input   en;
        input   clk;

        reg     [7:0] flip;

        output  [0:7] op;
        reg             [0:7] op;

        always @(pos) begin
                case(pos)
                        3'b000: begin flip = 8'b00000001; end
                        3'b001: begin flip = 8'b00000010; end
                        3'b010: begin flip = 8'b00000100; end
                        3'b011: begin flip = 8'b00001000; end
                        3'b100: begin flip = 8'b00010000; end
                        3'b101: begin flip = 8'b00100000; end
                        3'b110: begin flip = 8'b01000000; end
                        3'b111: begin flip = 8'b10000000; end
                endcase
        end

        always @(posedge clk)
                if (en)
                        op = op ^ flip;
                else
                        op = op;
endmodule
```

References

1. “Parity Generator and Parity Check” - Electronics Hub . https://www.electronicshub.org/parity-generator-and-parity-check/
2. I.E. Sutherland & Bob. F. Sproull., “Logical effort: Designing fast CMOS circuits, Advanced research in VLSI”, Morgan Kaufmanns Publishers, 1999.
3. I.E. Sutherland and R.F. Sproull, “Logical Effort: Designing for Speed on the Back of an Envelope”, in C.H. Sequin, Ed., Advanced Research in VLSI. Cambridge, MA: MIT Press, 1991.
4. New Efficient Design for XOR Function on the Transistor Level , AIP Conference Proceedings 1324, 346 (2010), Tripti Sharma, K. G. Sharma, B. P. Singh, and Neha Arora - https://doi.org/10.1063/1.3526229
5. Hardware Modelling using Verilog , NPTEL - Hardware Modeling using Verilog - YouTube
6. Logical effort : ElectronTube - YouTube

Appendix

Inverter Specifications:

λ = 90 nm		
Wp = 8λ		tplh = 9 ps
Wn = 6λ		tphl = 10 ps
Unit inverter delay: 9.5 ps		
Power	8.058 μW	
max Cgsn	0.674	fF
max Cgdn	0.674	fF
max Cgsp	0.68	fF
max Cgdp	0.68	fF

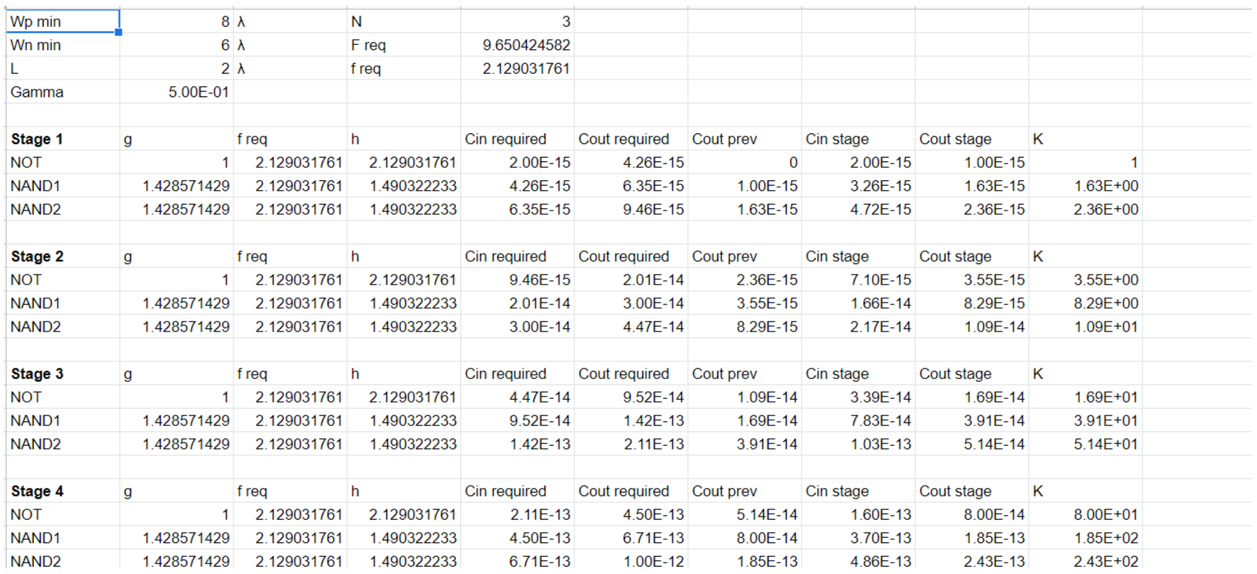
NAND Specifications:

Power	8.452	μW
tphl	12	ps
tplh	9	ps

Logical Effort Calculation (Chain):

t	9 ps		N	4					
g (XOR)	2.040816327		H	5.00E+02					
p (XOR)	5		B	1					
Cin (Inv)	2.00E-15		G	17.34665256					
Cout (Inv)	1.00E-15		F	8.67E+03		f	9.650424582		
Cload	1.00E-12					h	4.728708045		
Gamma	5.00E-01								
	Stage	g	p	h	Cin required	Cout required	Cout prev	Cin stage req	Cout stage
	1	2.040816327	5	4.728708045	2.00E-15	9.46E-15	0	2.00E-15	1.00E-1
	2	2.040816327	5	4.728708045	9.46E-15	4.47E-14	1.00E-15	8.46E-15	4.23E-1
	3	2.040816327	5	4.728708045	4.47E-14	2.11E-13	4.23E-15	4.05E-14	2.02E-1
	4	2.040816327	5	4.728708045	2.11E-13	1.00E-12	2.02E-14	1.91E-13	9.56E-1

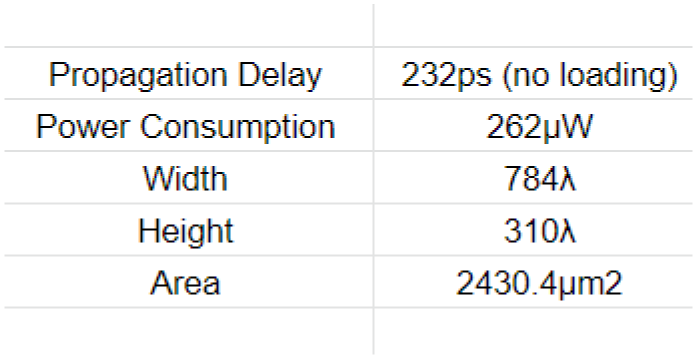
K value calculations:



Wp min	8	λ	N	3					
Wn min	6	λ	F req	9.650424582					
L	2	λ	f req	2.129031761					
Gamma	5.00E-01								
Stage 1	g	f req	h	Cin required	Cout required	Cout prev	Cin stage	Cout stage	K
NOT	1	2.129031761	2.129031761	2.00E-15	4.26E-15	0	2.00E-15	1.00E-15	1
NAND1	1.428571429	2.129031761	1.490322233	4.26E-15	6.35E-15	1.00E-15	3.26E-15	1.63E-15	1.63E+00
NAND2	1.428571429	2.129031761	1.490322233	6.35E-15	9.46E-15	1.63E-15	4.72E-15	2.36E-15	2.36E+00
Stage 2	g	f req	h	Cin required	Cout required	Cout prev	Cin stage	Cout stage	K
NOT	1	2.129031761	2.129031761	9.46E-15	2.01E-14	2.36E-15	7.10E-15	3.55E-15	3.55E+00
NAND1	1.428571429	2.129031761	1.490322233	2.01E-14	3.00E-14	3.55E-15	1.66E-14	8.29E-15	8.29E+00
NAND2	1.428571429	2.129031761	1.490322233	3.00E-14	4.47E-14	8.29E-15	2.17E-14	1.09E-14	1.09E+01
Stage 3	g	f req	h	Cin required	Cout required	Cout prev	Cin stage	Cout stage	K
NOT	1	2.129031761	2.129031761	4.47E-14	9.52E-14	1.09E-14	3.39E-14	1.69E-14	1.69E+01
NAND1	1.428571429	2.129031761	1.490322233	9.52E-14	1.42E-13	1.69E-14	7.83E-14	3.91E-14	3.91E+01
NAND2	1.428571429	2.129031761	1.490322233	1.42E-13	2.11E-13	3.91E-14	1.03E-13	5.14E-14	5.14E+01
Stage 4	g	f req	h	Cin required	Cout required	Cout prev	Cin stage	Cout stage	K
NOT	1	2.129031761	2.129031761	2.11E-13	4.50E-13	5.14E-14	1.60E-13	8.00E-14	8.00E+01
NAND1	1.428571429	2.129031761	1.490322233	4.50E-13	6.71E-13	8.00E-14	3.70E-13	1.85E-13	1.85E+02
NAND2	1.428571429	2.129031761	1.490322233	6.71E-13	1.00E-12	1.85E-13	4.86E-13	2.43E-13	2.43E+02

K = how many unit inverters the circuit corresponds to sizing wise.

Overall Circuit Specifications:



Propagation Delay	232ps (no loading)
Power Consumption	262µW
Width	784λ
Height	310λ
Area	2430.4µm2