Full Adder CMOS Layout on Glade

Alanoud Alsalem

Abstract

This report details the design and validation of a CMOS full-adder circuit using Glade software. CMOS technology's ideal properties, such as zero static power consumption, make it suitable for digital logic circuits. The full-adder, which adds three bits to produce sum and carry outputs, is implemented using XOR, AND, and OR gates. The design process involves creating pull-up and pull-down networks with PMOS and NMOS transistors, and ensuring accuracy through DRC, LPE, and LVS checks.

Table of Contents

1 INTRODUCTION

CMOS technology is used in digital logic circuits due to its close-to-ideal properties which include zero static power consumption, approximately infinite fan out, and almost ideal voltage transfer characteristic (VTC) curve. Hence, CMOS technology can be used to design digital logic circuits, with the focus in this report being on designing a full-adder circuit using the Glade software.

1.1 OBJECTIVES

The main objectives of this project are to design a full-adder circuit on the Glade software using CMOS technology. The resulting design will then be validated by testing input and output combinations using the Spice3 simulation software.

1.2 THEORY

A full adder is a circuit that adds three bits and produces the sum and carry. Hence, it is a circuit with three inputs and two outputs. Let A, B, and C_{in} be the three inputs and S and C_{out} be the two outputs representing the sum and carry, respectively. The full adder can be split into two functions, with the first generating the sum S, and the second generating the carry C_{out} [1].

The function for S is given by:

$$S = A \oplus B \oplus C_{in}$$

The function for C_{out} is given by:

$$C_{out} = AB + (A \oplus B)C_{in}$$

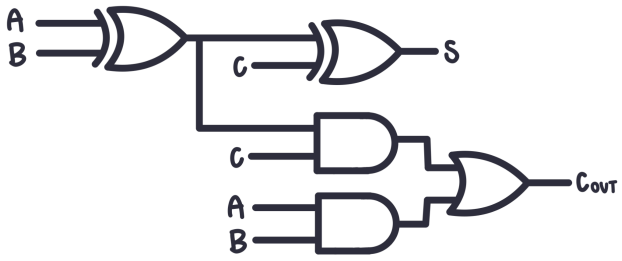


Figure 1: Full-Adder Circuit Schematic

The visual representation of the circuit schematic is shown in figure 1. The eight input and output combinations for the full-adder are tabulated below:

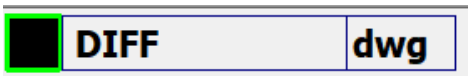
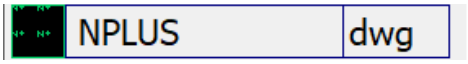
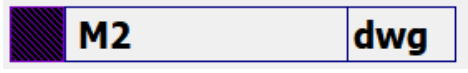
Table 1: Full-Adder Input and Output Combinations

A	B	C_{in}	S	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

In CMOS technology, a circuit is split into a pull-up network (PUN) composed of PMOS transistors, and a pull-down network composed of NMOS transistors. Generally, an AND function is represented by transistors in series while an OR function is represented by transistors in parallel. The overall function being designed is modeled twice in the circuit; once in the PUN and once in the PDN. In the PUN, the function's inputs are inverted, while in the PDN, the function itself is inverted. Therefore, these design principles must be implemented for both the sum and output carry functions.

Since the Glade software utilizes different silicone layers to design CMOS circuits, the functionality of each of the layers must first be defined. The different layers and their respective functions are organized in table 2 below:

Table 2: Silicone Layers and their Functionalities

Layer Name	Color	Function
Poly		Gate
Diffusion		Determines w and l of the transistor
Nwell		PMOS Substrate
Pwell	Embedded	NMOS Substrate
PPlus		PMOS source and drain
NPlus		NMOS source and drain
M1		Metal conductor insulated from M2
M2		Metal conductor insulated from M1
CO		Contacts for metals

Finally, the DRC, LPE, and LVS functions are used to verify the layout and schematic designs.

2 Procedure and Methods

Sum function

To simplify the design of the sum function, the three-input XOR function is implemented as two cascaded two-input XOR functions:

$$S = (A \oplus B) \oplus C = Y \oplus C, \qquad \text{where } Y = A \oplus B$$

The two XOR functions are expanded to allow for circuit visualization:

$$A \oplus B = A'B + B'A \qquad \text{and} \qquad Y \oplus C = Y'C + YC'$$

The CMOS circuit schematics for the two function are shown in figure 2 below:

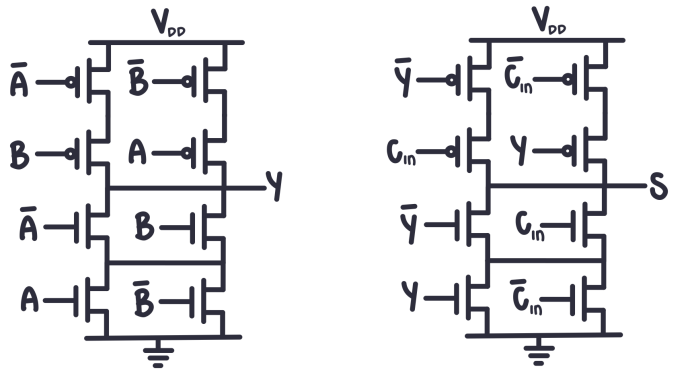


Figure 2: CMOS Sum Circuit Schematic

The stick diagrams shown in figure 3 are used as an initial step for layout design:

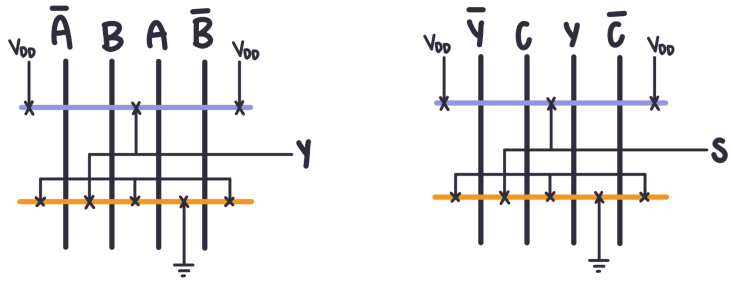
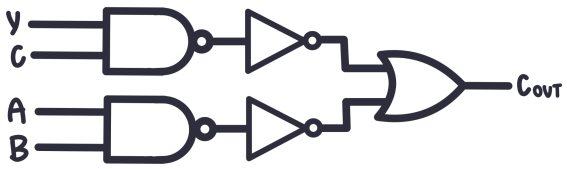


Figure 3: CMOS Sum Circuit Stick Diagrams

Output Carry Function

The output carry function is implemented using two AND gates and one OR gate. The AND gates will be implemented as NAND gates and an inverter will be added at the output. This design is visualized in figure 4.



The CMOS circuit schematics for the NAND gates, inverter, and OR gate (X+Z) are shown in figure 5 below:

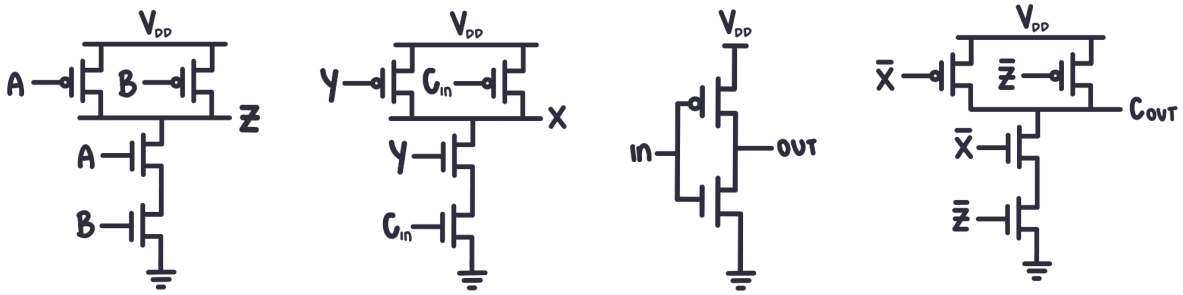


Figure 5: CMOS NAND, Inverter, and NOR Circuit Schematics

The stick diagrams shown in figure 6 are used as an initial step for layout design of the NAND and OR circuits.

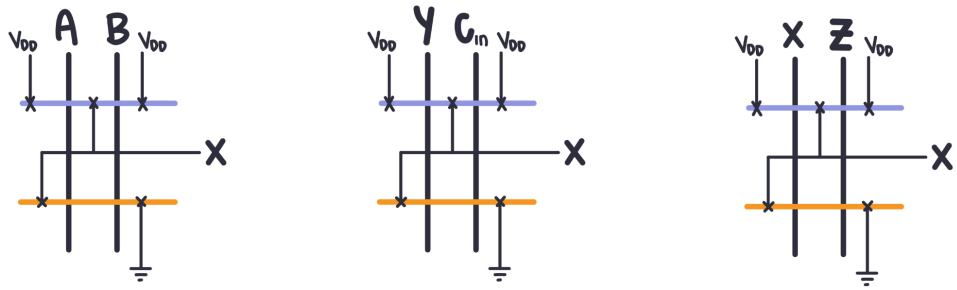


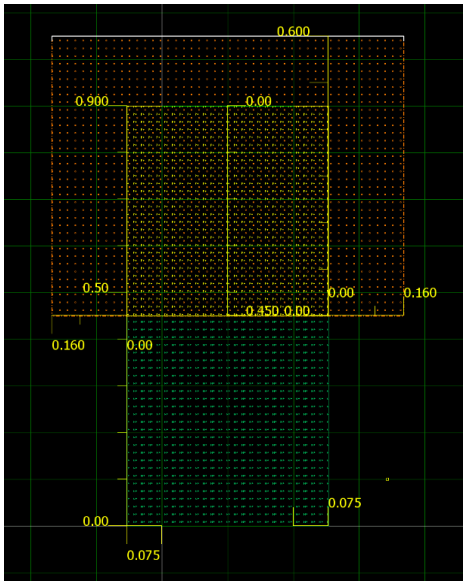
Figure 6: CMOS NAND and NOR Stick Diagrams

The inverter has a simple design and hence does not require a stick diagram.

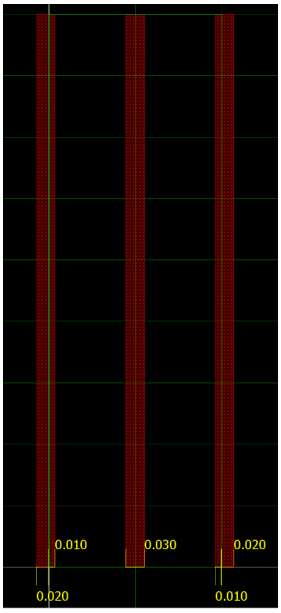
To design the circuits for each of the sum and output carry functions, the layouts of the gates required are created using the Glade software.

XOR Gate Layout and Schematic

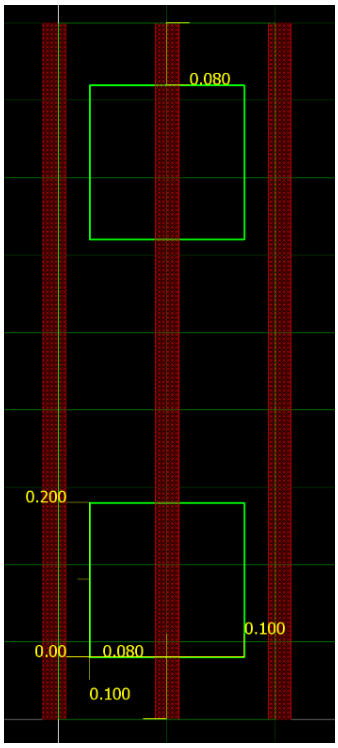
1. To design the XOR gate, the PPlus, NPlus, and Nwell layers for the first input with the appropriate dimensions were added:



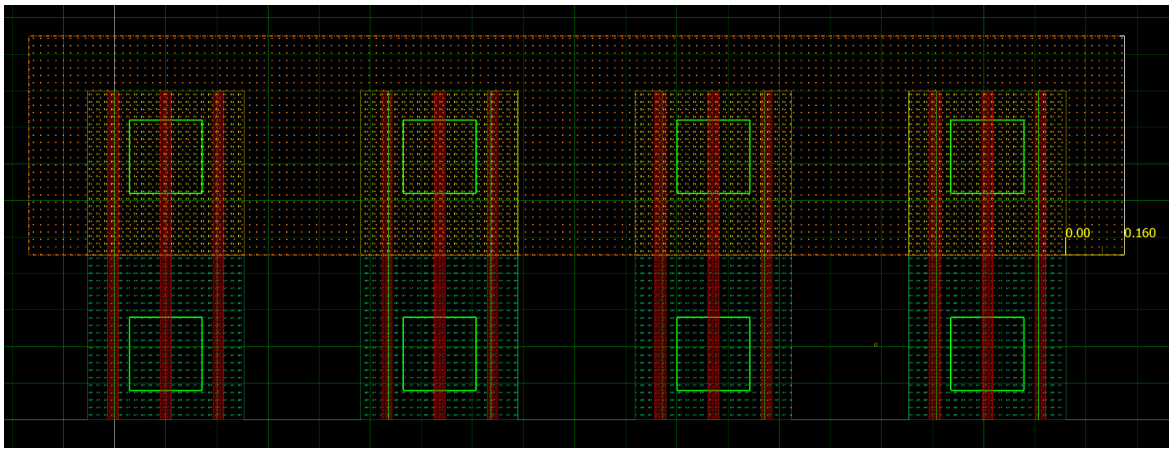
2. The poly and dummy polys were added:



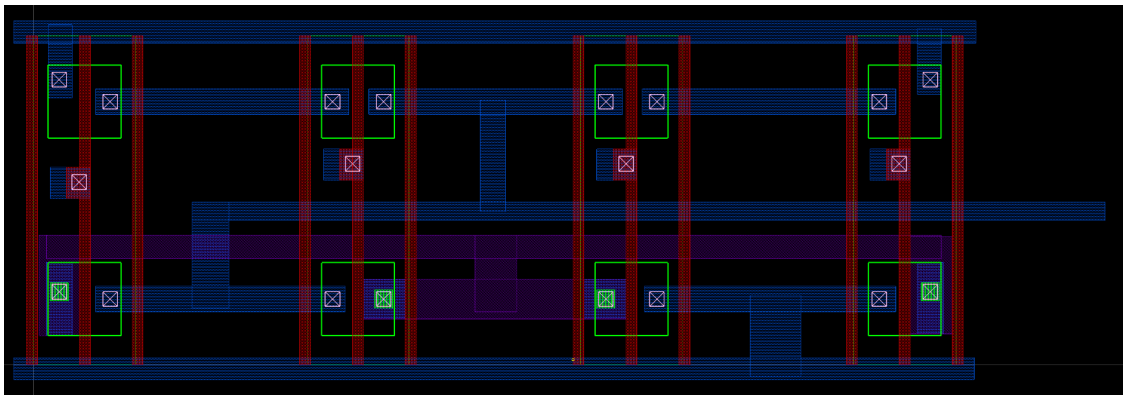
3. The diffusions were added. Since the last number of my student ID is 1, the lengths of the diffusions (w) are both equal to 0.2:



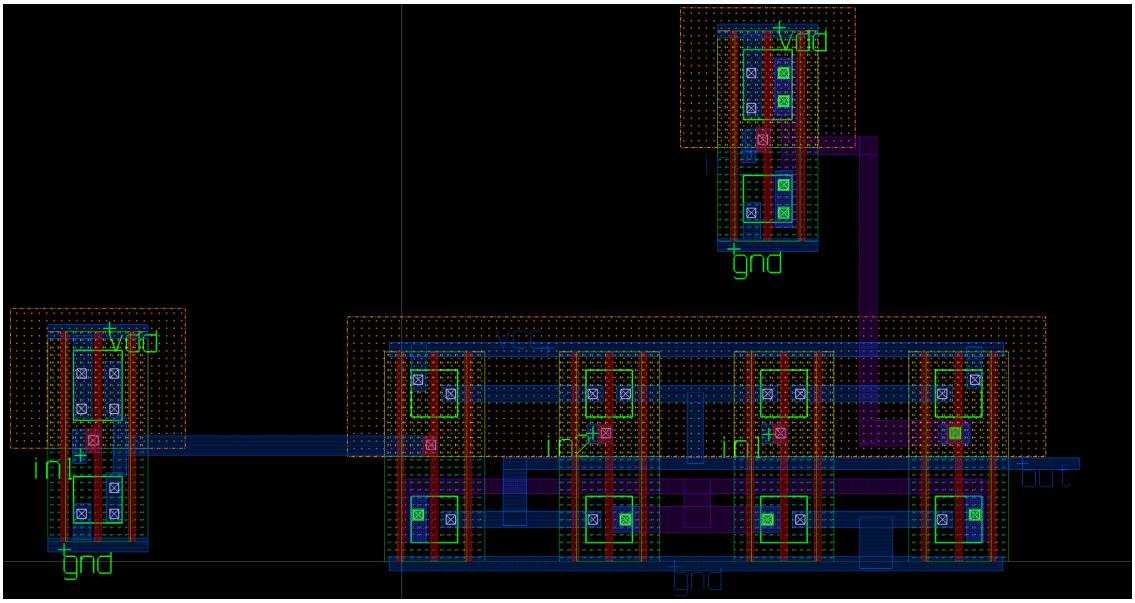
4. Everything except the NWell was duplicated for the remaining inputs and the NWell was extended so that it covers the remaining transistors:



5. The nodes of each transistor were connected according the stick diagrams initially created using contacts and the appropriate metals:



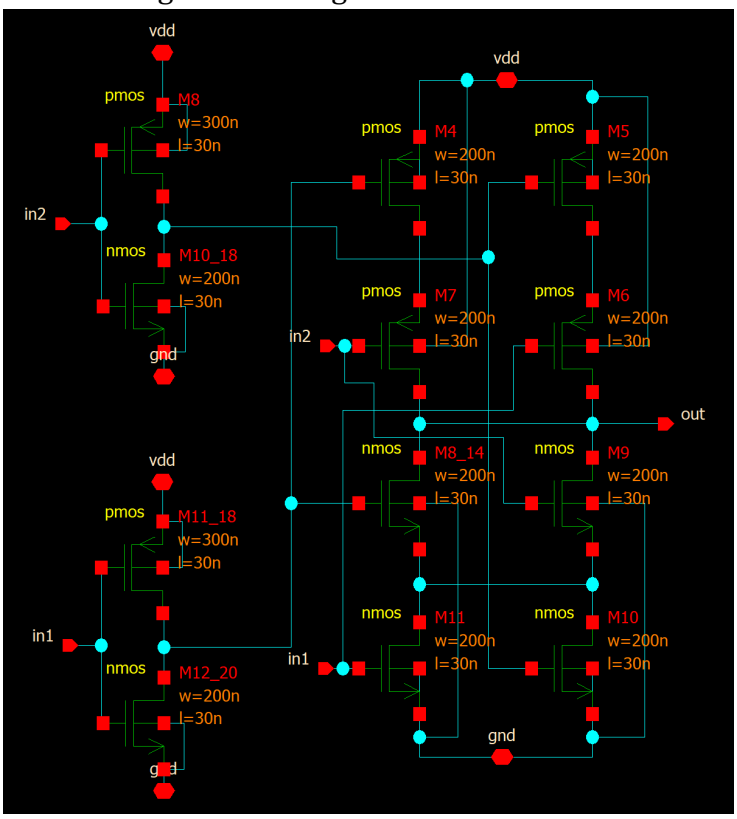
6. The inverter circuit design used for the previous assignment was imported and the inputs and outputs were labeled generically (in1, in2, and out) to allow for multi-purpose use:



7. Running the DRC to check for errors showed that no errors were found:

```
Checking VIA23 rules...
Checking M3 rules...
Checking overglass rules...
No DRC errors were found.
ui().winRedraw()
>>> >>> # INFO: DRC run completed.
>>>
```

8. The schematic for the generic XOR gate was created:

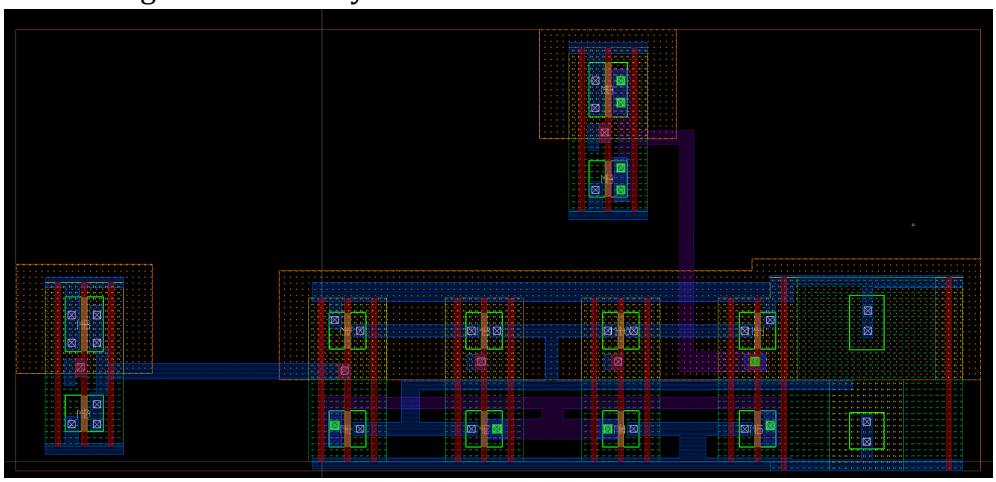


9. After checking the cell view for any errors:

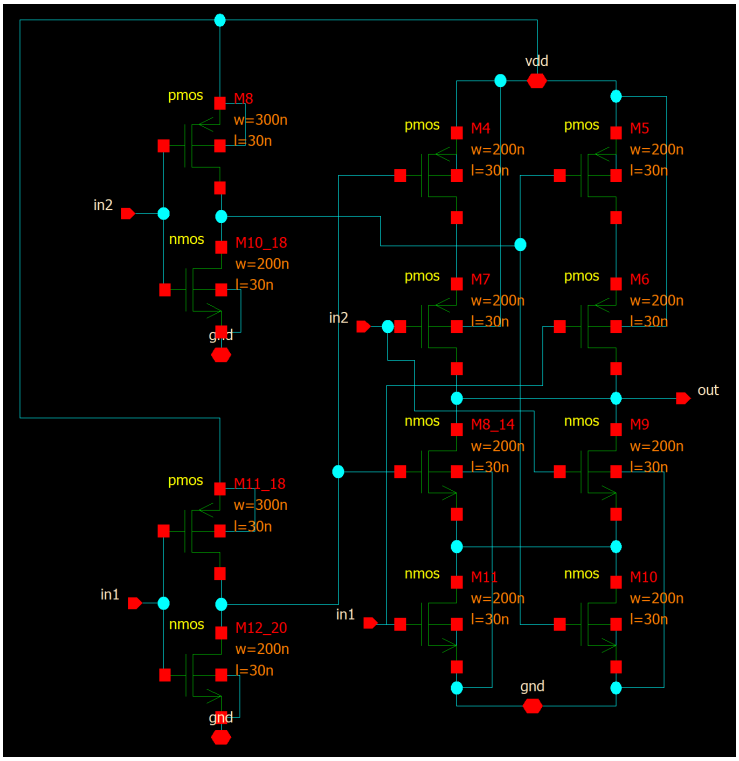
```
>>> # INFO: Adding wire point #2 at (2.55625, 0.65625)
>>> # INFO: Adding wire point #3 at (2.73125, 0.65625)
>>> # INFO: Created wire with 3 points
>>> ui.schCheck("Training","XOR","schematic", 0, 0.000000, 1)
# INFO: Cell XOR view schematic checked: 0 warnings, 0 errors
>>>
```

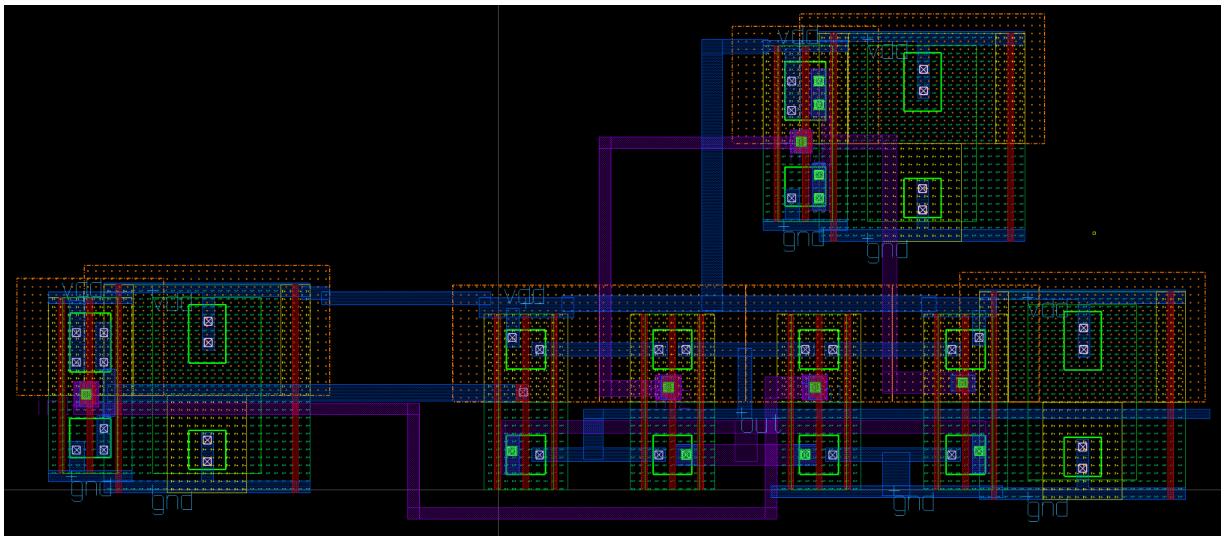
10. The schematic was extracted as a CDL file.

11. After adding the TAP the layout was extracted as an LPE.



12. After running the LVS, the errors that appeared were fixed and the final schematic and layout for the XOR gate are shown below:



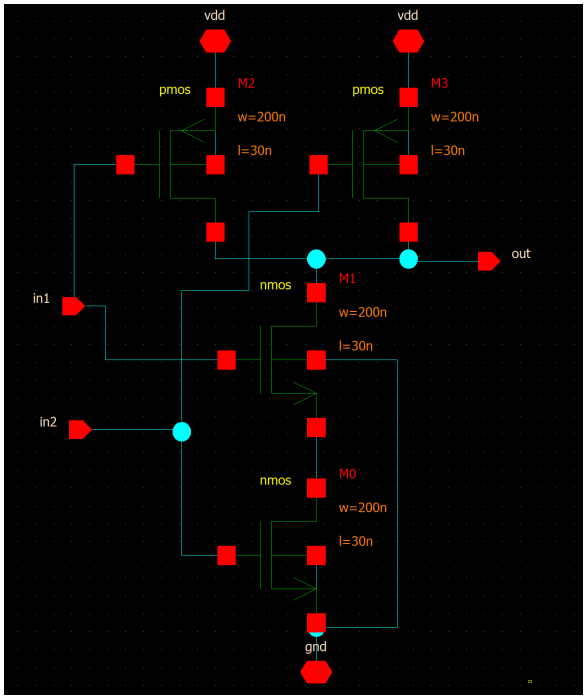


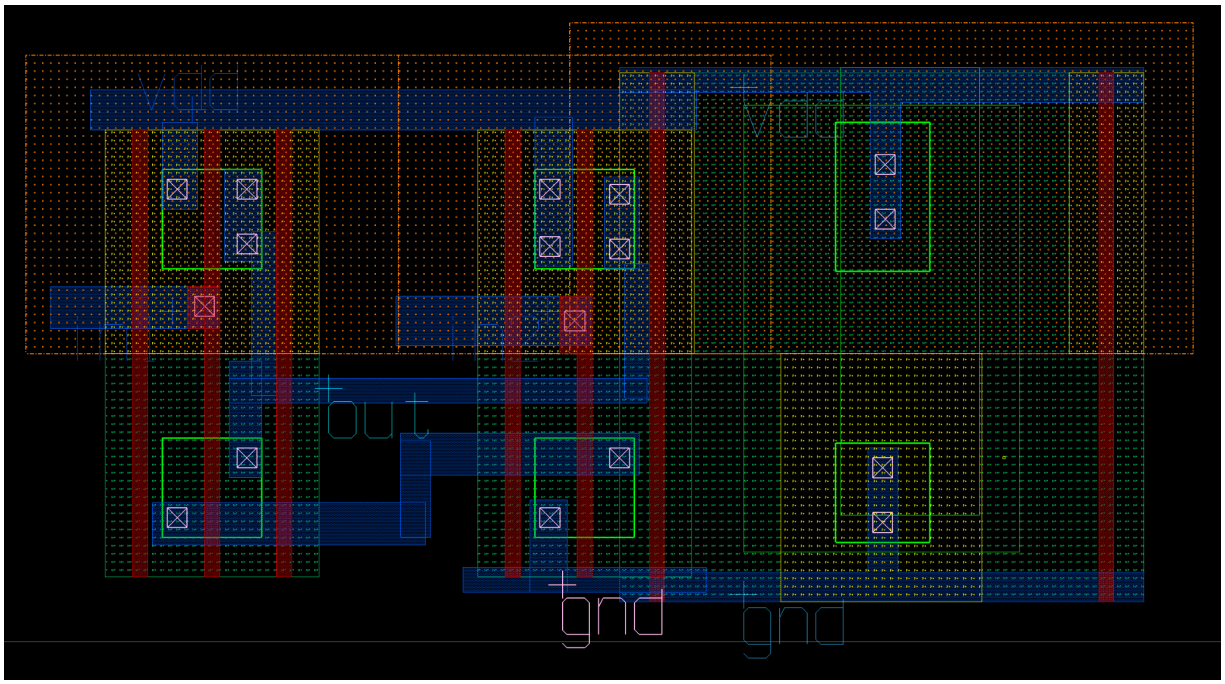
Running the LVS for the XOR shows no errors:

```
--------------------------------------------------------------------------------
      Netlist summary before reduction : XOR_extracted.cdl
--------------------------------------------------------------------------------
  Number of devices :       12
  Number of nets    :       10
  Number of ports   :       3

--------------------------------------------------------------------------------
      Netlist summary before reduction : XOR.cdl_flat
--------------------------------------------------------------------------------
  Number of devices :       12
  Number of nets    :       10
  Number of ports   :       3

--------------------------------------------------------------------------------
      Netlist summary after reduction :
--------------------------------------------------------------------------------
            XOR_extracted.cdl      XOR.cdl_flat
  Number of devices :       10             10
  Number of nets    :       8              8
  Number of ports   :       3              3

There were no device property errors.
10 (55%) matches were found by local matching.
All nodes were matched in 4 passes.

The netlists match.

0 devices and 0 nets written to C:\Users\Dodin\Downloads\XOR.err

Gemini completed at 17:52:24 on 06/08/2024

# INFO: LVS finished with exit code 0
# INFO: LVS completed.
>>>
```

NAND Gate Layout and Schematic

Repeating the same procedure for the NAND gate resulted in the following schematic and layout:



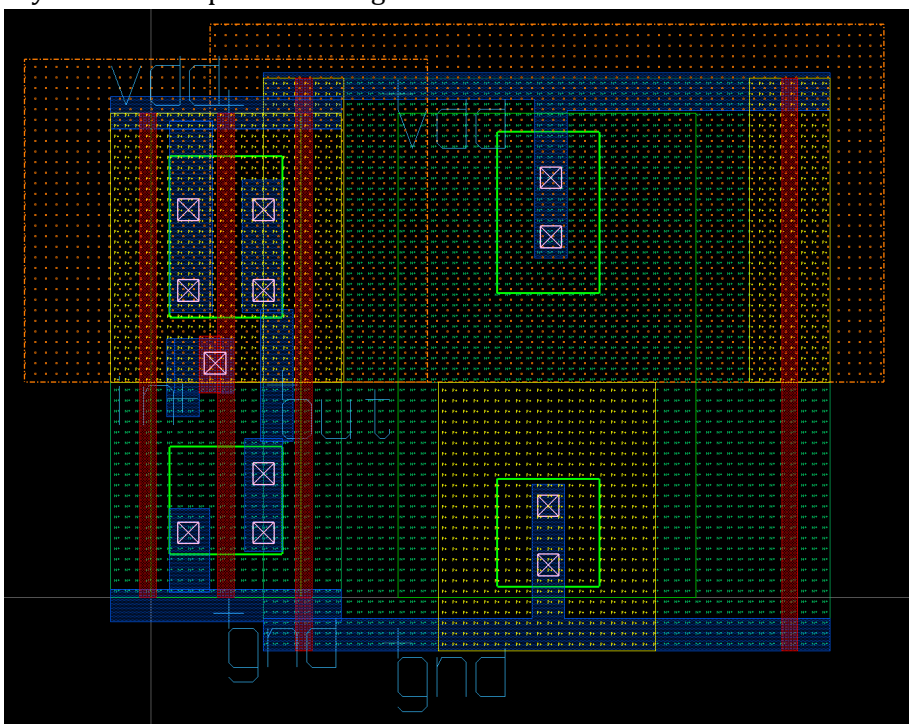


Running the LVS for the NAND shows no errors:

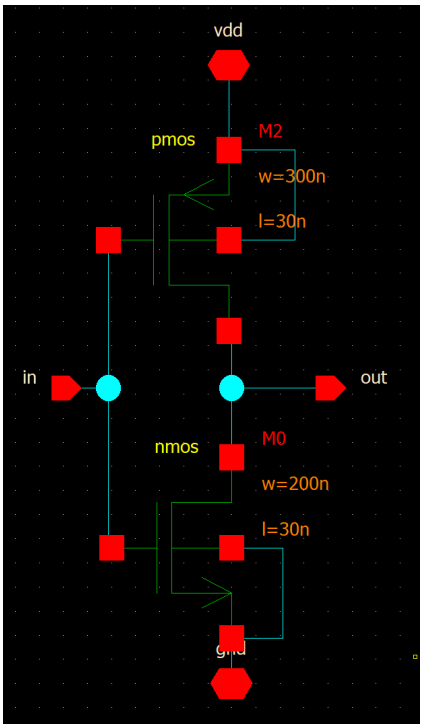
```
--------------------------------------------------------------------------------
      Netlist summary before reduction : NAND_extracted.cdl
--------------------------------------------------------------------------------
  Number of devices :       4
  Number of nets    :       6
  Number of ports   :       0

--------------------------------------------------------------------------------
      Netlist summary before reduction : NAND.cdl_flat
--------------------------------------------------------------------------------
  Number of devices :       4
  Number of nets    :       6
  Number of ports   :       3

--------------------------------------------------------------------------------
      Netlist summary after reduction :
--------------------------------------------------------------------------------
            NAND_extracted.cdl     NAND.cdl_flat
  Number of devices :       3              3
  Number of nets    :       5              5
  Number of ports   :       0              3


These circuits contain some symmetry (50% nodes not yet matched).
Gemini will attempt to find a valid match for symmetrical nodes.
##There were no device property errors.
3 (37%) matches were found by local matching.
All nodes were matched in 7 passes.

The netlists match.

0 devices and 0 nets written to C:\Users\Dodin\Downloads\glade (No TAP Errors)\NAND.err

Gemini completed at 17:55:47 on 06/08/2024

# INFO: LVS finished with exit code 0
# INFO: LVS completed.
>>>
```

Inverter Layout and Schematic

The inverter layout from the previous assignment was used:



The inverter schematic:



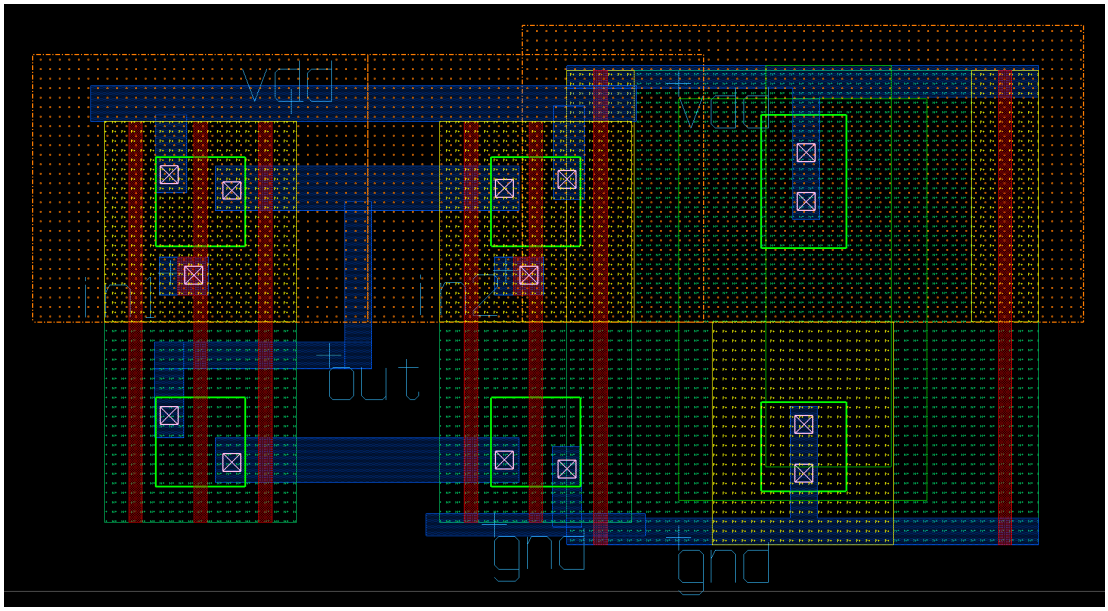
Running the LVS for the inverter shows no errors:

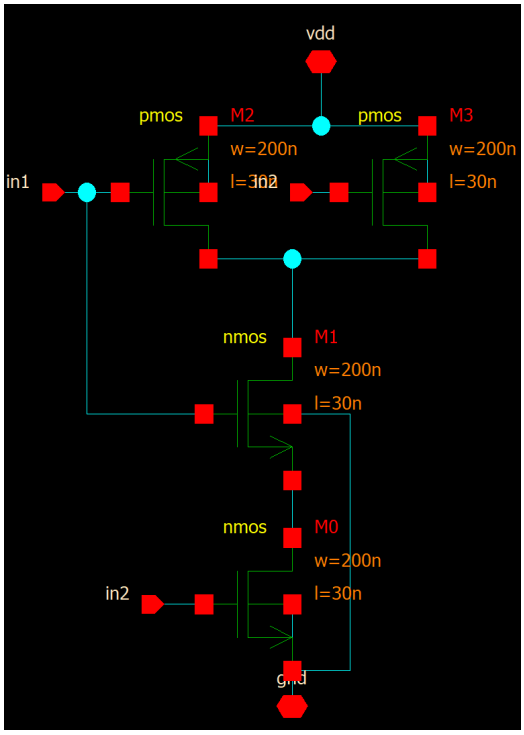
```
--------------------------------------------------------------------------------
      Netlist summary before reduction : InverterFinal_extracted.cdl
--------------------------------------------------------------------------------
  Number of devices :       2
  Number of nets    :       4
  Number of ports   :       2

--------------------------------------------------------------------------------
      Netlist summary before reduction : InverterFinal.cdl_flat
--------------------------------------------------------------------------------
  Number of devices :       2
  Number of nets    :       4
  Number of ports   :       0

--------------------------------------------------------------------------------
      Netlist summary after reduction :
--------------------------------------------------------------------------------
            InverterFinal_extracted.cdl    InverterFinal.cdl_flat
  Number of devices :       2              2
  Number of nets    :       4              4
  Number of ports   :       2              0

There were no device property errors.
2 (33%) matches were found by local matching.
All nodes were matched in 2 passes.

The netlists match.

0 devices and 0 nets written to C:\Users\Dodin\Downloads\glade (No TAP Errors)\InverterFinal.err

Gemini completed at 18:12:16 on 06/08/2024

# INFO: LVS finished with exit code 0
# INFO: LVS completed.
```

OR Gate Layout and Schematic

The inputs in the stick diagram for the OR gate are inverted. This will be implemented using inverters for the inputs before they enter the gate.

Using the same procedure The final schematic and layout are shown below:



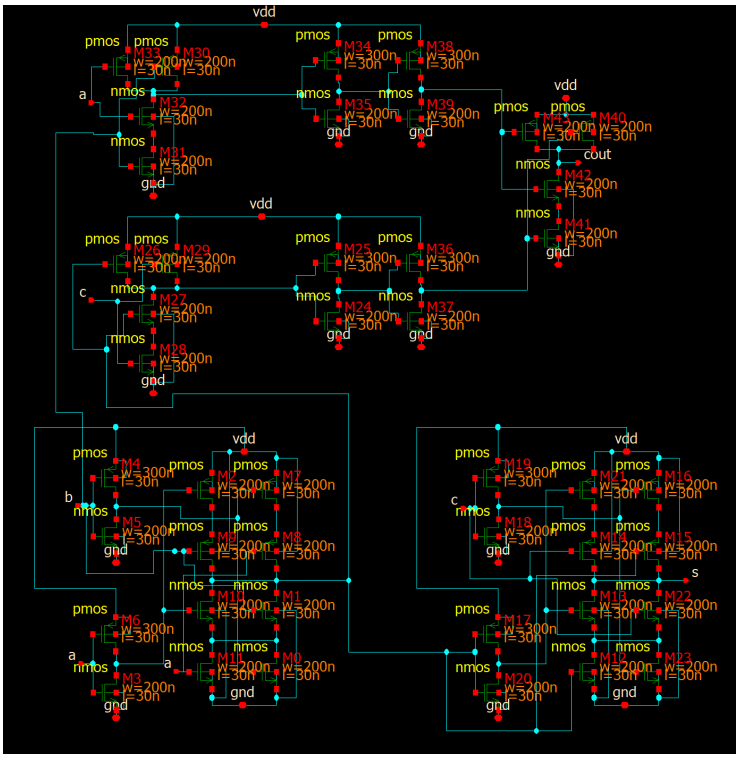


Running the LVS for the OR shows no errors:

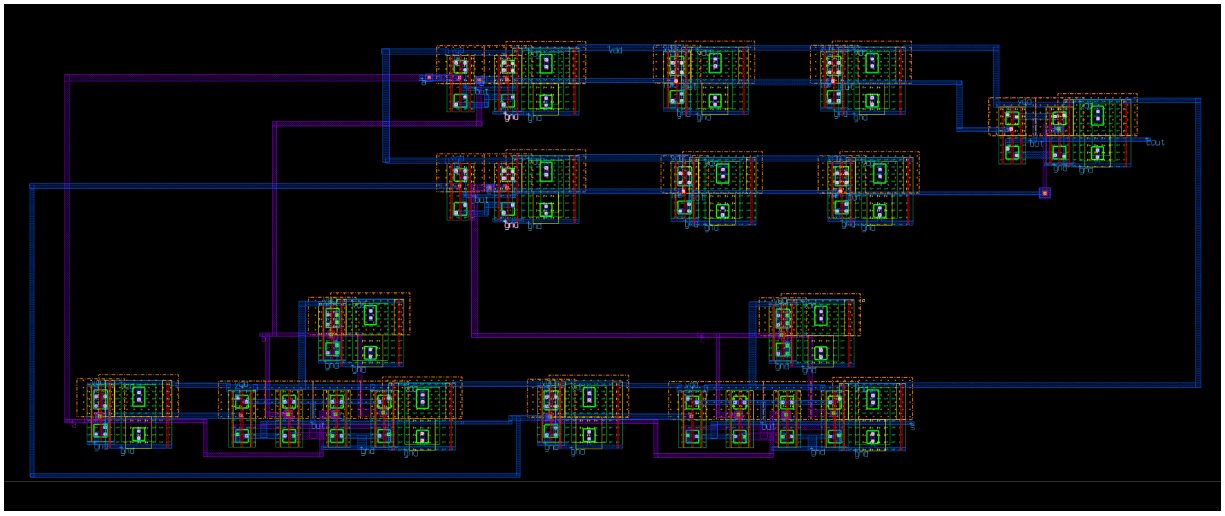
```
--------------------------------------------------------------------------------
      Netlist summary before reduction : OR_extracted.cdl
--------------------------------------------------------------------------------
  Number of devices :       4
  Number of nets    :       6
  Number of ports   :       3

--------------------------------------------------------------------------------
      Netlist summary before reduction : OR.cdl
--------------------------------------------------------------------------------
  Number of devices :       4
  Number of nets    :       6
  Number of ports   :       2

--------------------------------------------------------------------------------
      Netlist summary after reduction :
--------------------------------------------------------------------------------
             OR_extracted.cdl            OR.cdl
  Number of devices :       3             3
  Number of nets    :       5             5
  Number of ports   :       3             2


These circuits contain some symmetry (50% nodes not yet matched).
Gemini will attempt to find a valid match for symmetrical nodes.
M##There were no device property errors.
3 (37%) matches were found by local matching.
All nodes were matched in 7 passes.

The netlists match.

0 devices and 0 nets written to C:\Users\Dodin\Downloads\glade (No TAP Errors)\glade\tech\ENGR3426_mod\OR.err

Gemini completed at 18:23:15 on 06/08/2024

# INFO: LVS finished with exit code 0
# INFO: LVS completed.
```

Full Adder Design

I then used the layouts of each of the designed gates as instances in the full adder layout. I also used their schematics to create the overall schematic of the full adder. The schematic of the full adder is shown below:



The layout of the full adder is shown below:



The instances on the bottom are the cascaded XOR gates while the instances on the top are the NAND gates followed by a set of inverters (AND gates), then two sets of inverters with the final instance acting as the OR gate.

After fixing all errors, extracting the LPE, and running the LVS, the output was the following:

```
Message Window
--------------------------------------------------------------------------------
      Netlist summary before reduction : FULLADDER_extracted.cdl
--------------------------------------------------------------------------------
  Number of devices :       44
  Number of nets    :       27
  Number of ports   :       5

--------------------------------------------------------------------------------
      Netlist summary before reduction : FULLADDER.cdl_flat
--------------------------------------------------------------------------------
  Number of devices :       44
  Number of nets    :       27
  Number of ports   :       5

--------------------------------------------------------------------------------
      Netlist summary after reduction :
--------------------------------------------------------------------------------
            FULLADDER_extracted.cdl  FULLADDER.cdl_flat
  Number of devices :       37              37
  Number of nets    :       20              20
  Number of ports   :       5               5

There were no device property errors.
44 (77%) matches were found by local matching.
All nodes were matched in 10 passes.

The netlists match.

0 devices and 0 nets written to C:\Users\Dodin\Downloads\FULLADDER.err

Gemini completed at 16:42:44 on 06/08/2024

# INFO: LVS finished with exit code 0
# INFO: LVS completed.
>>>
```

3 CONCLUSIONS

In conclusion, a full-adder circuit was successfully designed using the appropriate layout and schematic. The process showed the importance of abstraction and how designing gates then using them to construct a larger function allows for a streamlined process of digital logic circuit design. The final design was verified by extracting the LPE and running the LVS.

4 References

[1] GeeksforGeeks, “Full adder in digital logic,” GeeksforGeeks, https://www.geeksforgeeks.org/full-adder-in-digital-logic/.