Princess Sumaya University for Technology
King Abdullah II Faculty of Engineering
Electrical Engineering Department

Digital Electronics (21332)
Full Adder Project

Author: Qais Jildeh (20210155)
Supervisor: Eng. Hazem Marar

August 6, 2024

Abstract

This project involves the design and implementation of a CMOS-based full adder circuit using GLADE software. The design utilizes 2 CMOS XOR gates, 2 CMOS AND gates, and 1 OR gate, the transistor sizing follows the ratio P=5N. This project includes running DRC (Design Rule Check), LVS (Layout Versus Schematic), and LPE (Layout Parameter Extraction) tests to ensure the design's correctness. The layout is carefully designed based on sketched stick diagrams and schematics. All design steps, including screenshots and diagrams are attached to this document.

TABLE OF CONTENTS

1 INTRODUCTION

The Introduction section of a laboratory report gives a brief description of the experiment. It should clearly identify the objectives of the experiment, the importance of the experiment, and the theoretical background for understanding the experiment.

1.1 OBJECTIVES

In digital electronics, designing reliable circuits is very important. This project focuses on creating a CMOS-based full adder circuit using GLADE software. The design uses 2 CMOS XOR gates, 2 CMOS AND gates, and 1 OR gate, with transistor sizes set in a P=5N ratio. The document will contain the steps on how to design each and every gate using many figures.

1.2 THEORY

1.2.1 Full Adder Logic

To build any digital logic circuit, we need to use logical gates to provide the logic required. We will explain how using AND, OR, and XOR gates aided in the construction of the full adder.

Starting with AND, an AND gate is a digital logic gate that implements logical conjunction from mathematical logic.

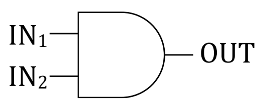


Figure 1. 2-Input, Single Output AND Gate

AND gate behaves according to Table 1:

IN_1	IN_2	OUT
0	0	0
0	1	0
1	0	0
1	1	1

Table 1. Truth Table for AND Gate

As shown in Table 1, an AND gate will output a logical 1 when both inputs are logical 1. Otherwise, it will output a logical 0. Thus, we can express this behavior with the following Boolean expression:

$$OUT = IN_1 IN_2$$

Secondly, an OR gate is a digital logic gate that implements logical disjunction.

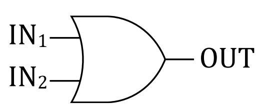


Figure 2. 2-Input, Single Output OR Gate

OR gate behaves according to Table 2:

IN_1	**IN_2**	**OUT**
0	0	0
0	1	1
1	0	1
1	1	1

Table 2. Truth Table for OR Gate

As shown in Table 2, an OR gate will output a logical 1 when either input is a logical 1. Otherwise, it will output a logical 0 when both inputs are logical 0. Thus, we can express this behavior with the following Boolean expression:

$$OUT = IN_1 + IN_2$$

Thirdly, an XOR gate is a digital logic gate that implements logical exclusive disjunction.

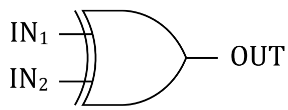


Figure 3. 2-Input, Single Output XOR Gate

XOR gate behaves according to Table 3:

IN_1	**IN_2**	**OUT**
0	0	0
0	1	1
1	0	1
1	1	0

Table 3. Truth Table for XOR Gate

As shown in Table 3, an XOR gate will output a logical 1 when one of the inputs is a logical 1 and the other is a logical 0. Otherwise, it will output a logical 0 when both inputs are logical 0 or when both inputs are logical 1. Thus, we can express this behavior with the following Boolean expression:

$$OUT = \overline{IN_1}\, IN_2 + IN_1\, \overline{IN_2}$$

Using the Boolean expression of XOR, we can find out that an XOR gate can be constructed using AND, OR gates, and inverter gates.

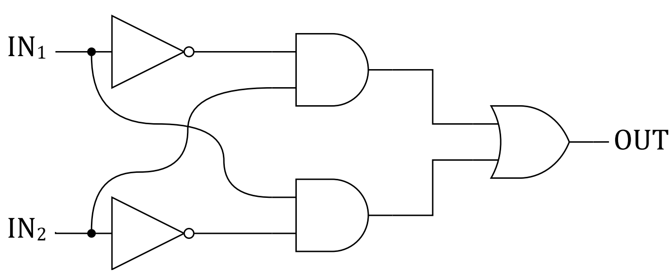


Figure 4. XOR Gate Using AND, OR, NOT Gates

Lastly, using the gates stated above we can construct a full adder. A full adder takes three inputs, adds them together and provides two outputs, sum and carry out.

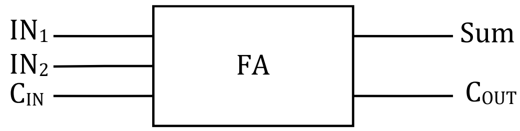


Figure 5. Full Adder with Inputs IN_1, IN_2, C_{IN} and Outputs Sum, C_{OUT}

Full Adder behaves according to Table 4:

C_{in}	IN_1	IN_2	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

Table 4. Truth Table for Full Adder

As shown in Table 4, a full adder will provide an output sum of logical 1 if all inputs are logical 1 or only a single input is equal to logical 1. Moreover, the output carry will be a logical 1 if all inputs are logical 1 or two of the inputs are logical 1.

Using the Table 4, we can find the Boolean expression to represent the sum and carry out of a full adder:

$$C_{OUT} = C_{IN}\overline{IN_1}IN_2 + C_{IN}IN_1\overline{IN_2} + \overline{C_{IN}}IN_1IN_2 + C_{IN}IN_1IN_2$$

$$C_{OUT} = C_{IN}(\overline{IN_1}IN_2 + IN_1\overline{IN_2}) + IN_1IN_2(\overline{C_{IN}} + C_{IN})$$

$$\rightarrow C_{OUT} = IN_1IN_2 + C_{IN}(IN_1 \oplus IN_2)$$

$$Sum = C_{IN}\overline{IN_1IN_2} + \overline{C_{IN}IN_1}IN_2 + \overline{C_{IN}}IN_1\overline{IN_2} + C_{IN}IN_1IN_2$$

$$\rightarrow Sum = C_{IN} \oplus IN_1 \oplus IN_2$$

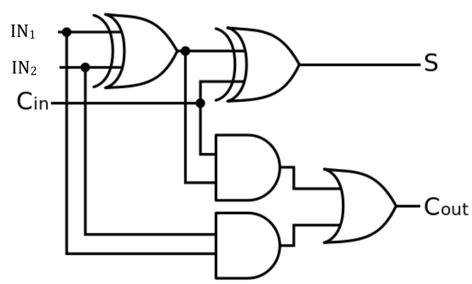


Figure 6. Full Adder using AND, OR, and XOR Gates

1.2.2 Glade Layout

In Glade, the layout is a detailed graphical representation of an IC that includes the geometric shapes and patterns that will be fabricated onto the silicon wafer. The layout defines the physical implementation of the circuit and consists of layers that represent different materials and processes used in IC fabrication. The following table shows the used layers in this project:

Layer	Description
Diffusion	Represents regions where the semiconductor material is doped to create p-type or n-type areas.
Nwell	A region of n-type semiconductor material created in a p-type substrate. This is typically used to house p-channel MOS transistors (PMOS).
Pplus	Highly doped p-type regions used to form p-channel transistors or make connections in the substrate. These regions have a higher concentration of p-type dopants.
Nplus	Highly doped n-type regions used to form n-channel transistors or make connections in the substrate. These regions have a higher concentration of n-type dopants.
Cont (Contact)	Represents the openings in the insulating layer that allow connections between different layers, such as between the metal layers and the semiconductor layers.
Metals (M1, M2, …)	Used to create interconnections between different components of the chip, allowing signals and power to be routed across the circuit.
VIA (VIA12, VIA23)	Connect different layers of metals to together which provides a vertical connection between these layers,
Boundary	Defines the physical boundary of the chip or a specific area within the chip design.

Table 1. Layers and their description

2 PROCEDURE AND METHODS

To start creating the full adder, the first step is to draw the CMOS circuit that represents our logical gates.

2.1 AND GATE

As stated in section 1.2.1, an AND gate can be represented with the following logical function $Y = AB$, this logical function is used to draw a schematic for the gate using CMOS technology. The circuit is divided into two networks, pull up network (PUN) and a pull down network (PDN). To draw the pull down network of the CMOS AND gate, the function will be inverted and use the same inputs. However, for a pull up circuit, invert the inputs only. The final result will look as follows:

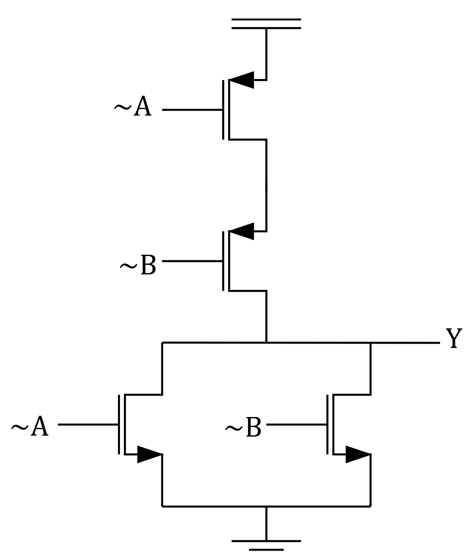


Figure 7. CMOS AND Gate

The following is the implementation of the CMOS AND gate circuit as a schematic in the GLADE software, which was done by referring to figure 7:

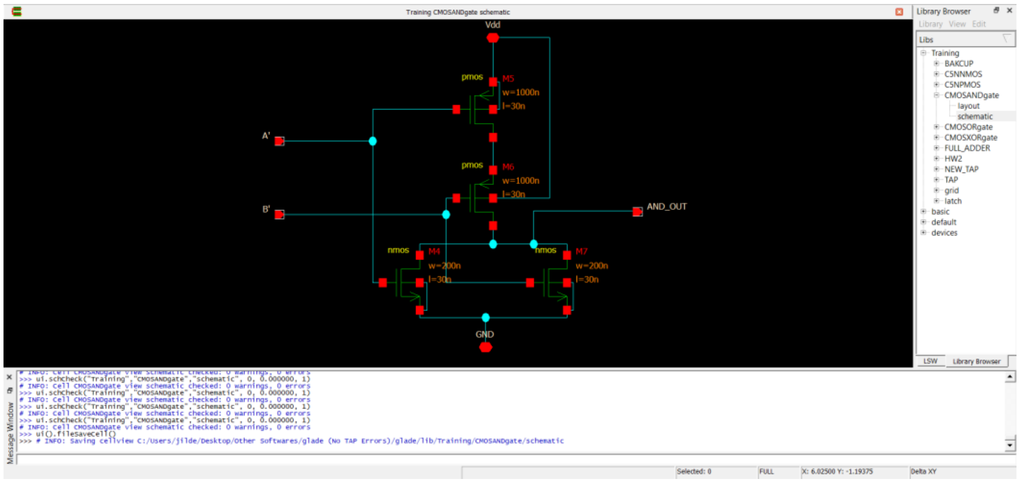


Figure 8. Schematic for CMOS AND Gate on GLADE

To build the layout of our gate, it is important to create a stick diagram which aids in giving a simplified view on how the layout would look like. It provides information about the inputs, outputs, wires, and contacts in each diffusion layer. For an AND gate, the stick diagram will be as follows:

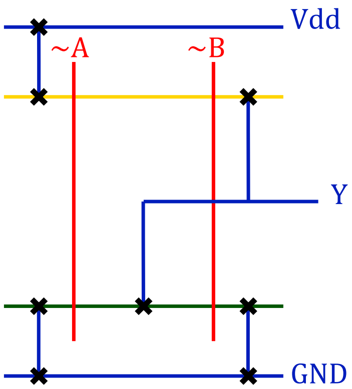


Figure 9. Stick Diagram for CMOS AND Gate

By using the stick diagram from figure 9, the layout was easily implemented on GLADE, and it looks as follows:

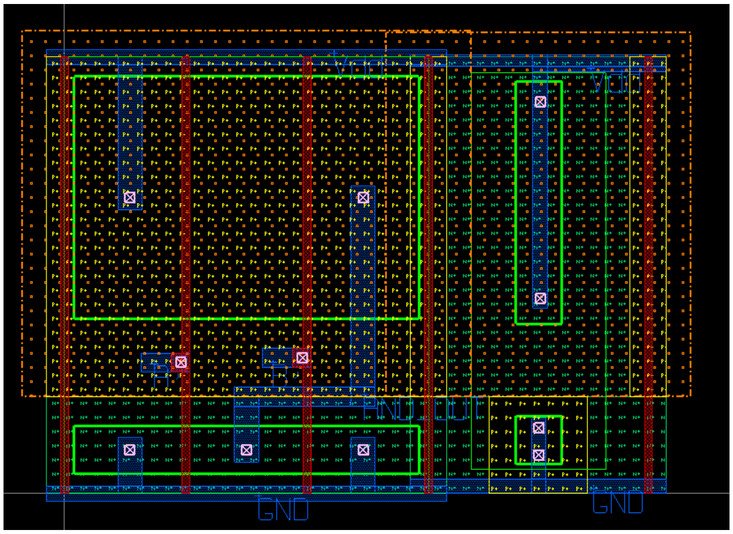
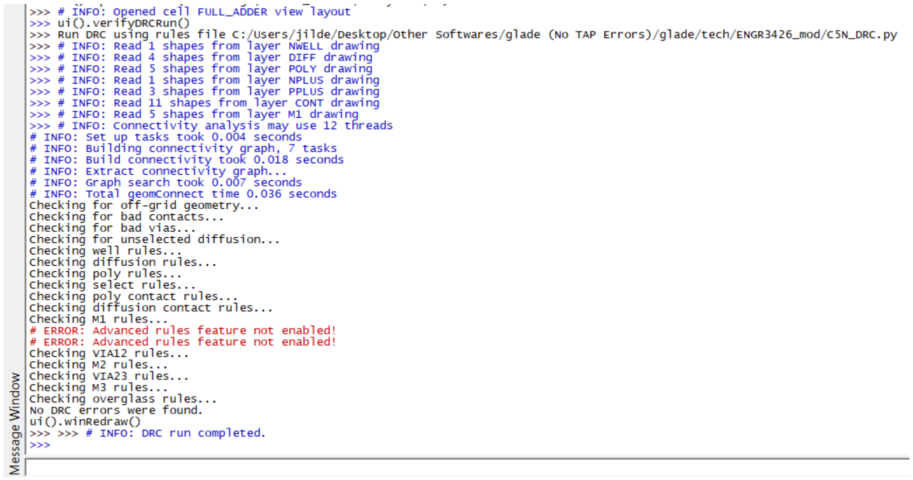


Figure 10. Layout for CMOS AND Gate

To ensure that the layout satisfies all design rules, the DRC test is used. When applying the layout created in figure 10, it shows that all rules are adhered to with no design flaws.



```
>>> # INFO: Opened cell FULL_ADDER view layout
>>> ui().verifyDRCRun()
>>> Run DRC using rules file C:/Users/jilde/Desktop/Other Softwares/glade (No TAP Errors)/glade/tech/ENGR3426_mod/C5N_DRC.py
>>> # INFO: Read 1 shapes from layer NWELL drawing
>>> # INFO: Read 4 shapes from layer DIFF drawing
>>> # INFO: Read 5 shapes from layer POLY drawing
>>> # INFO: Read 1 shapes from layer NPLUS drawing
>>> # INFO: Read 3 shapes from layer PPLUS drawing
>>> # INFO: Read 11 shapes from layer CONT drawing
>>> # INFO: Read 5 shapes from layer M1 drawing
>>> # INFO: Connectivity analysis may use 12 threads
# INFO: Set up tasks took 0.004 seconds
# INFO: Building connectivity graph, 7 tasks
# INFO: Build connectivity took 0.018 seconds
# INFO: Extract connectivity graph...
# INFO: Graph search took 0.007 seconds
# INFO: Total geomConnect time 0.036 seconds
Checking for off-grid geometry...
Checking for bad contacts...
Checking for bad vias...
Checking for unselected diffusion...
Checking well rules...
Checking diffusion rules...
Checking poly rules...
Checking select rules...
Checking poly contact rules...
Checking diffusion contact rules...
Checking M1 rules...
# ERROR: Advanced rules feature not enabled!
# ERROR: Advanced rules feature not enabled!
Checking VIA12 rules...
Checking M2 rules...
Checking VIA23 rules...
Checking M3 rules...
Checking overglass rules...
No DRC errors were found.
ui().winRedraw()
>>> >>> # INFO: DRC run completed.
>>>
```


Figure 11. DRC Test for AND Gate Layout

After making sure that no DRC errors exist, it is required to create an extract of our layout using LPE test that will be then used to test schematic against the layout. The following two figures show the results after running LPE test:

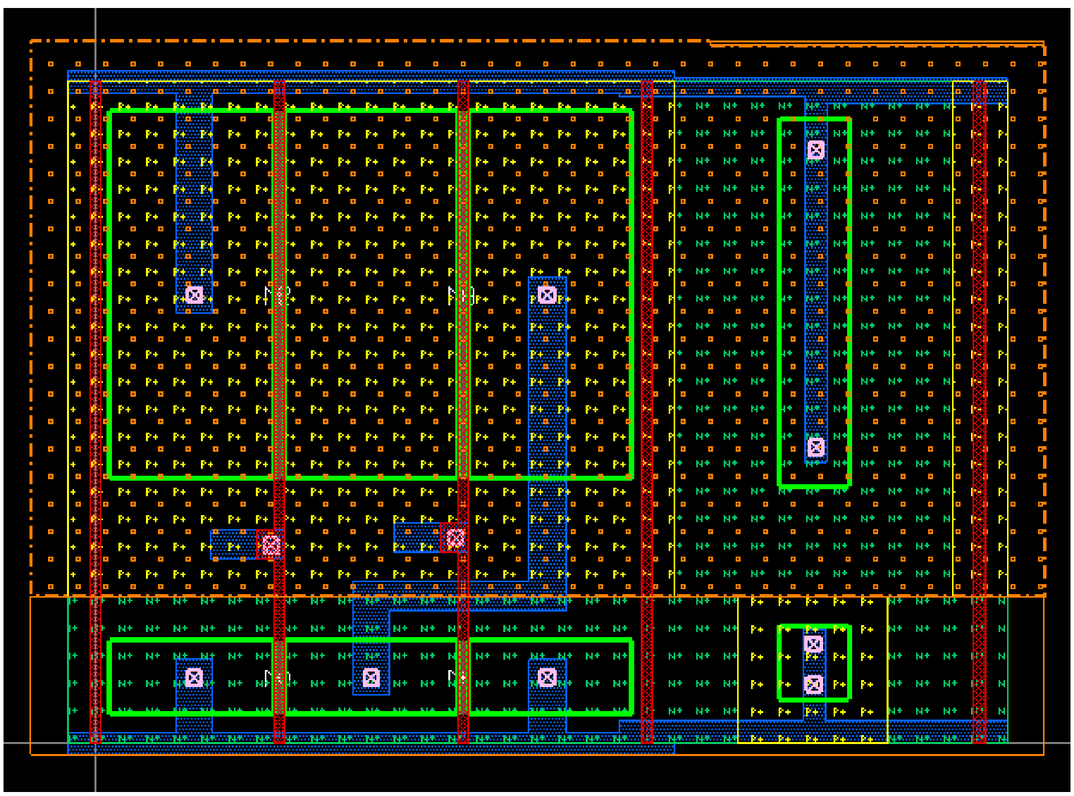
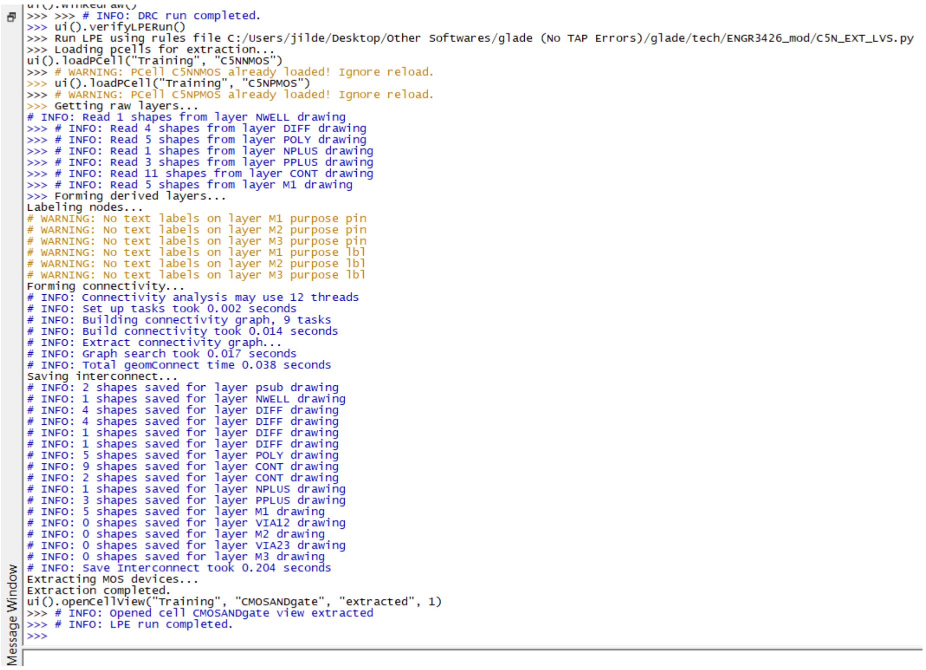


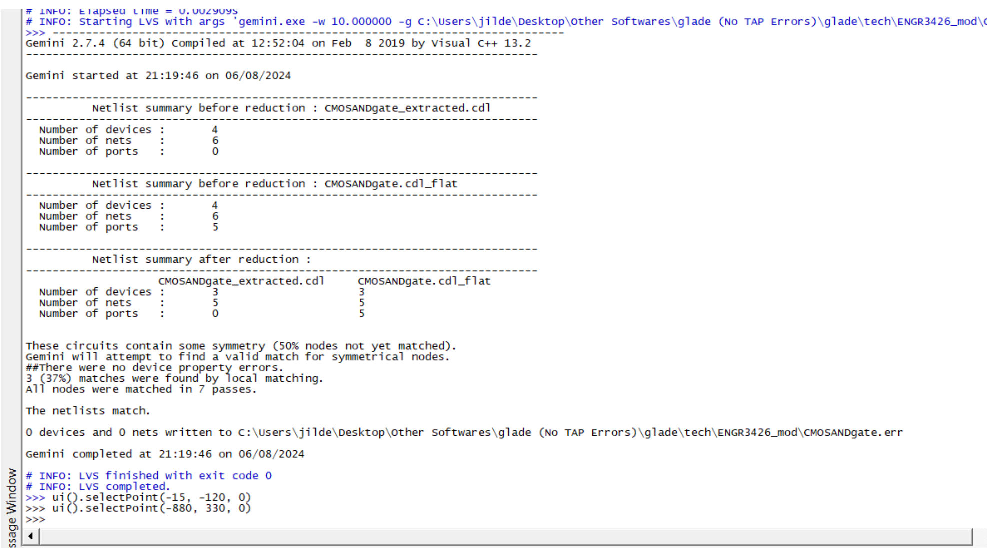
Figure 12. Extracted Layout for AND Gate



```
>>> >>> # INFO: DRC run completed.
>>> ui().verifyLPERun()
>>> Run LPE using rules file C:/Users/jilde/Desktop/Other Softwares/glade (No TAP Errors)/glade/tech/ENGR3426_mod/C5N_EXT_LVS.py
>>> Loading pcells for extraction...
ui().loadPCell("Training", "C5NNMOS")
>>> # WARNING: PCell C5NNMOS already loaded! Ignore reload.
>>> ui().loadPCell("Training", "C5NPMOS")
>>> # WARNING: PCell C5NPMOS already loaded! Ignore reload.
>>> Getting raw layers...
# INFO: Read 1 shapes from layer NWELL drawing
>>> # INFO: Read 4 shapes from layer DIFF drawing
>>> # INFO: Read 5 shapes from layer POLY drawing
>>> # INFO: Read 1 shapes from layer NPLUS drawing
>>> # INFO: Read 3 shapes from layer PPLUS drawing
>>> # INFO: Read 11 shapes from layer CONT drawing
>>> # INFO: Read 5 shapes from layer M1 drawing
>>> Forming derived layers...
Labeling nodes...
# WARNING: No text labels on layer M1 purpose pin
# WARNING: No text labels on layer M2 purpose pin
# WARNING: No text labels on layer M3 purpose pin
# WARNING: No text labels on layer M1 purpose lbl
# WARNING: No text labels on layer M2 purpose lbl
# WARNING: No text labels on layer M3 purpose lbl
Forming connectivity...
# INFO: Connectivity analysis may use 12 threads
# INFO: Set up tasks took 0.002 seconds
# INFO: Building connectivity graph, 9 tasks
# INFO: Build connectivity took 0.014 seconds
# INFO: Extract connectivity graph...
# INFO: Graph search took 0.017 seconds
# INFO: Total geomConnect time 0.038 seconds
Saving interconnect...
# INFO: 2 shapes saved for layer psub drawing
# INFO: 1 shapes saved for layer NWELL drawing
# INFO: 4 shapes saved for layer DIFF drawing
# INFO: 4 shapes saved for layer DIFF drawing
# INFO: 1 shapes saved for layer DIFF drawing
# INFO: 1 shapes saved for layer DIFF drawing
# INFO: 5 shapes saved for layer POLY drawing
# INFO: 9 shapes saved for layer CONT drawing
# INFO: 2 shapes saved for layer CONT drawing
# INFO: 1 shapes saved for layer NPLUS drawing
# INFO: 3 shapes saved for layer PPLUS drawing
# INFO: 5 shapes saved for layer M1 drawing
# INFO: 0 shapes saved for layer VIA12 drawing
# INFO: 0 shapes saved for layer M2 drawing
# INFO: 0 shapes saved for layer VIA23 drawing
# INFO: 0 shapes saved for layer M3 drawing
# INFO: Save Interconnect took 0.204 seconds
Extracting MOS devices...
Extraction completed.
ui().openCellView("Training", "CMOSANDgate", "extracted", 1)
>>> # INFO: Opened cell CMOSANDgate view extracted
>>> # INFO: LPE run completed.
>>>
```


Figure 13. LPE Test Result for AND Gate Layout

Then for the last test, layout versus schematic (LVS) test. This test is used to show the percentage of similarity between the layout and the schematic. In addition, it provides information about netlists, number of ports, etc. The result of running the LVS test is as follows:



```
# INFO: Starting LVS with args 'gemini.exe -w 10.000000 -g C:\Users\jilde\Desktop\Other Softwares\glade (No TAP Errors)\glade\tech\ENGR3426_mod\C
>>> ------------------------------------------------------------------------------
Gemini 2.7.4 (64 bit) Compiled at 12:52:04 on Feb  8 2019 by Visual C++ 13.2
----------------------------------------------------------------------------

Gemini started at 21:19:46 on 06/08/2024

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

----------------------------------------------------------------------------
          Netlist summary before reduction : CMOSANDgate.cdl_flat
----------------------------------------------------------------------------
  Number of devices :       4
  Number of nets    :       6
  Number of ports   :       5

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


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\jilde\Desktop\Other Softwares\glade (No TAP Errors)\glade\tech\ENGR3426_mod\CMOSANDgate.err

Gemini completed at 21:19:46 on 06/08/2024

# INFO: LVS finished with exit code 0
# INFO: LVS completed.
>>> ui().selectPoint(-15, -120, 0)
>>> ui().selectPoint(-880, 330, 0)
>>>
```


Figure 14. LVS Test Result for AND Gate

2.2 OR GATE

As stated in section 1.2.1, an OR gate can be represented with the following logical function $Y = A + B$, this logical function is used to draw a schematic for the gate using CMOS technology. The circuit is divided into two networks, pull up network (PUN) and a pull down network (PDN). To draw the pull down network of the CMOS OR gate, the function will be inverted and use the same inputs. However, for a pull up circuit, invert the inputs only. The final result will look as follows:

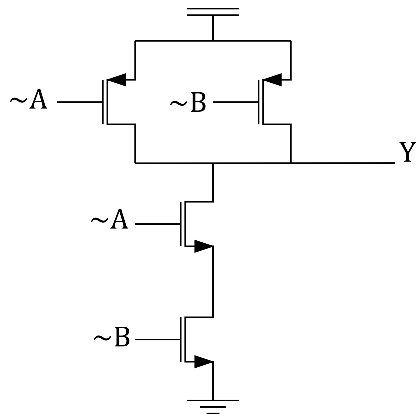


Figure 15. CMOS OR Gate

The following is the implementation of the CMOS OR gate circuit as a schematic in the GLADE software, which was done by referring to figure 15:

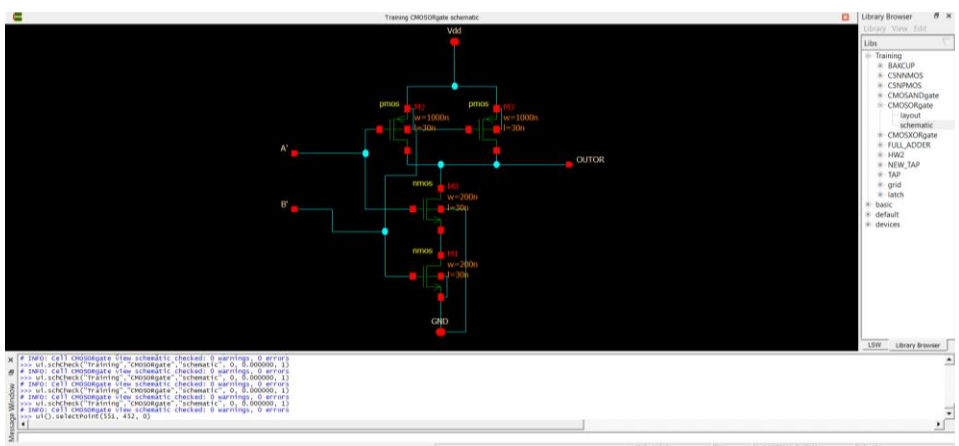


Figure 16. Schematic for CMOS OR Gate on GLADE

To build the layout of our gate, it is important to create a stick diagram which aids in giving a simplified view on how the layout would look like. It provides information about the inputs, outputs, wires, and contacts in each diffusion layer. For an OR gate, the stick diagram will be as follows:

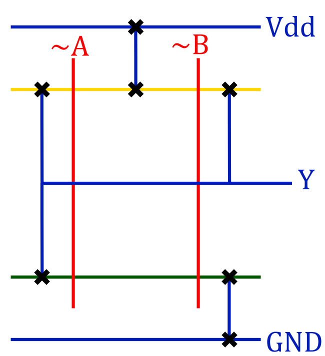


Figure 17. Stick Diagram for CMOS OR Gate

By using the stick diagram from figure 17, the layout was easily implemented on GLADE, and it looks as follows:

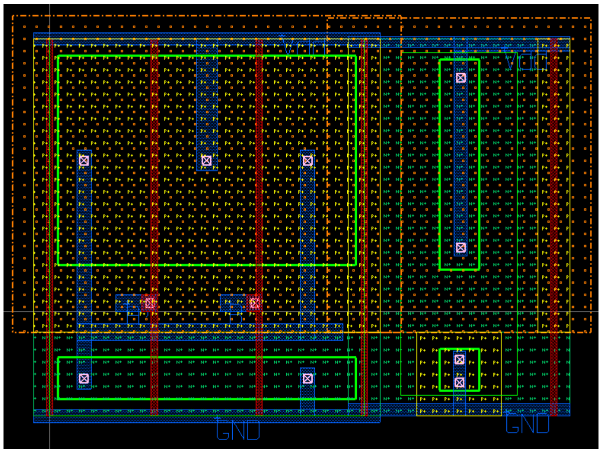
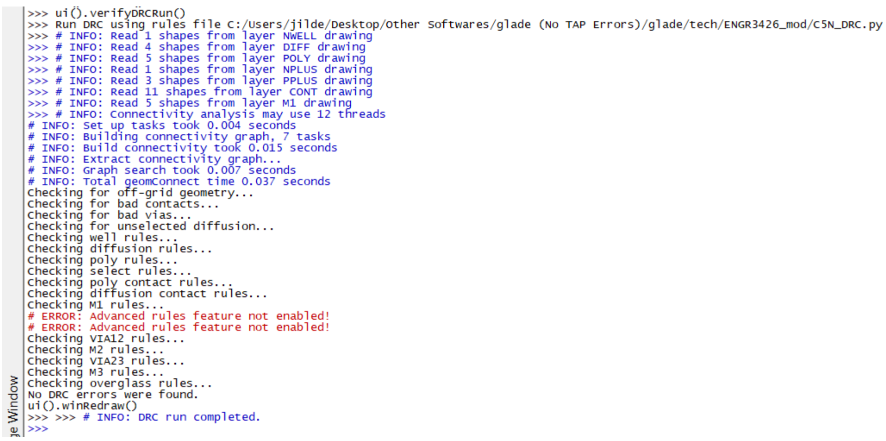


Figure 18. Layout for CMOS OR Gate

To ensure that the layout satisfies all design rules, the DRC test is used. When applying the layout created in figure 18, it shows that all rules are adhered to with no design flaws.



```
>>> ui().verifyDRCRun()
>>> Run DRC using rules file C:/Users/jilde/Desktop/Other Softwares/glade (No TAP Errors)/glade/tech/ENGR3426_mod/C5N_DRC.py
>>> # INFO: Read 1 shapes from layer NWELL drawing
>>> # INFO: Read 4 shapes from layer DIFF drawing
>>> # INFO: Read 5 shapes from layer POLY drawing
>>> # INFO: Read 1 shapes from layer NPLUS drawing
>>> # INFO: Read 3 shapes from layer PPLUS drawing
>>> # INFO: Read 11 shapes from layer CONT drawing
>>> # INFO: Read 5 shapes from layer M1 drawing
>>> # INFO: Connectivity analysis may use 12 threads
# INFO: Set up tasks took 0.004 seconds
# INFO: Building connectivity graph, 7 tasks
# INFO: Build connectivity took 0.015 seconds
# INFO: Extract connectivity graph...
# INFO: Graph search took 0.007 seconds
# INFO: Total geomConnect time 0.037 seconds
Checking for off-grid geometry...
Checking for bad contacts...
Checking for bad vias...
Checking for unselected diffusion...
Checking well rules...
Checking diffusion rules...
Checking poly rules...
Checking select rules...
Checking poly contact rules...
Checking diffusion contact rules...
Checking M1 rules...
# ERROR: Advanced rules feature not enabled!
# ERROR: Advanced rules feature not enabled!
Checking VIA12 rules...
Checking M2 rules...
Checking VIA23 rules...
Checking M3 rules...
Checking overglass rules...
No DRC errors were found.
ui().winRedraw()
>>> >>> # INFO: DRC run completed.
>>>
```

Figure 19. DRC Test for OR Gate Layout

After making sure that no DRC errors exist, it is required to create an extract of our layout using LPE test that will be then used to test schematic against the layout. The following two figures show the results after running LPE test:

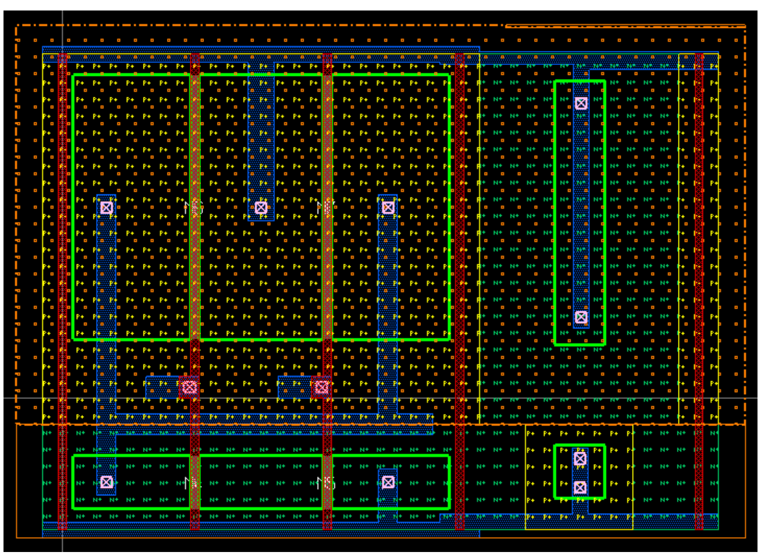


Figure 20. Extracted Layout for OR Gate

```
ui().verifyLPERun()
>>> Run LPE using rules file C:/Users/jilde/Desktop/Other Softwares/glade (No TAP Errors)/glade/tech/ENGR3426_mod/C5N_EXT_LVS.py
>>> Loading pcells for extraction...
ui().loadPCell("Training", "C5NNMOS")
>>> # WARNING: PCell C5NNMOS already loaded! Ignore reload.
>>> ui().loadPCell("Training", "C5NPMOS")
>>> # WARNING: PCell C5NPMOS already loaded! Ignore reload.
>>> Getting raw layers...
# INFO: Read 1 shapes from layer NWELL drawing
>>> # INFO: Read 4 shapes from layer DIFF drawing
>>> # INFO: Read 5 shapes from layer POLY drawing
>>> # INFO: Read 1 shapes from layer NPLUS drawing
>>> # INFO: Read 3 shapes from layer PPLUS drawing
>>> # INFO: Read 11 shapes from layer CONT drawing
>>> # INFO: Read 5 shapes from layer M1 drawing
>>> Forming derived layers...
Labeling nodes...
# WARNING: No text labels on layer M1 purpose pin
# WARNING: No text labels on layer M2 purpose pin
# WARNING: No text labels on layer M3 purpose pin
# WARNING: No text labels on layer M1 purpose lbl
# WARNING: No text labels on layer M2 purpose lbl
# WARNING: No text labels on layer M3 purpose lbl
Forming connectivity...
# INFO: Connectivity analysis may use 12 threads
# INFO: Set up tasks took 0.001 seconds
# INFO: Building connectivity graph, 9 tasks
# INFO: Build connectivity took 0.016 seconds
# INFO: Extract connectivity graph...
# INFO: Graph search took 0.021 seconds
# INFO: Total geomConnect time 0.044 seconds
Saving interconnect...
# INFO: 2 shapes saved for layer psub drawing
# INFO: 1 shapes saved for layer NWELL drawing
# INFO: 4 shapes saved for layer DIFF drawing
# INFO: 4 shapes saved for layer DIFF drawing
# INFO: 1 shapes saved for layer DIFF drawing
# INFO: 1 shapes saved for layer DIFF drawing
# INFO: 5 shapes saved for layer POLY drawing
# INFO: 9 shapes saved for layer CONT drawing
# INFO: 2 shapes saved for layer CONT drawing
# INFO: 1 shapes saved for layer NPLUS drawing
# INFO: 3 shapes saved for layer PPLUS drawing
# INFO: 5 shapes saved for layer M1 drawing
# INFO: 0 shapes saved for layer VIA12 drawing
# INFO: 0 shapes saved for layer M2 drawing
# INFO: 0 shapes saved for layer VIA23 drawing
# INFO: 0 shapes saved for layer M3 drawing
# INFO: Save Interconnect took 0.048 seconds
Extracting MOS devices...
Extraction completed.
ui().openCellView("Training", "CMOSORgate", "extracted", 1)
>>> # INFO: Opened cell CMOSORgate view extracted
>>> # INFO: LPE run completed.
>>>
```

Figure 21. LPE Test Result for OR Gate Layout

Then for the last test, layout versus schematic (LVS) test. This test is used to show the percentage of similarity between the layout and the schematic. In addition, it provides information about netlists, number of ports, etc. The result of running the LVS test is as follows:

```
>>> --------------------------------------------------------------------------
Gemini 2.7.4 (64 bit) Compiled at 12:52:04 on Feb  8 2019 by Visual C++ 13.2
--------------------------------------------------------------------------

Gemini started at 21:40:29 on 06/08/2024

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

--------------------------------------------------------------------------
         Netlist summary before reduction : CMOSORgate.cdl_flat
--------------------------------------------------------------------------
  Number of devices :       4
  Number of nets    :       6
  Number of ports   :       5

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

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\jilde\Desktop\Other Softwares\glade (No TAP Errors)\glade\tech\ENGR3426_mod\CMOSORgate.err

Gemini completed at 21:40:29 on 06/08/2024
# INFO: LVS finished with exit code 0
# INFO: LVS completed.
>>>
```

Figure 22. LVS Test Result for OR Gate

2.3 XOR GATE

As stated in section 1.2.1, an XOR gate can be represented with the following logical function $Y = A\bar{B} + \bar{A}B$, this logical function is used to draw a schematic for the gate using CMOS technology. The circuit is divided into two networks, pull up network (PUN) and a pull down network (PDN). To draw the pull down network of the CMOS XOR gate, the function will be inverted and use the same inputs. However, for a pull up circuit, invert the inputs only. The final result will look as follows:

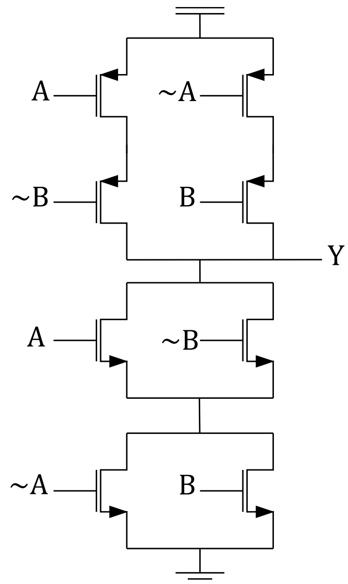


Figure 23. CMOS XOR Gate

The following is the implementation of the CMOS XOR gate circuit as a schematic in the GLADE software, which was done by referring to figure 23:

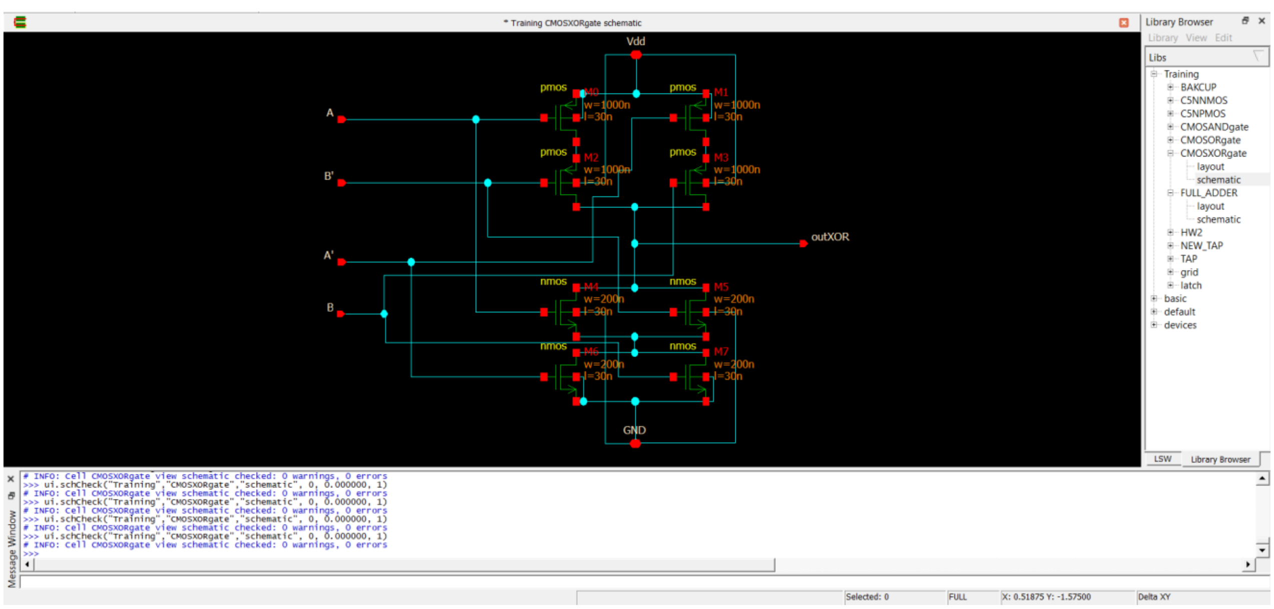


Figure 24. Schematic for CMOS XOR Gate on GLADE

To build the layout of our gate, it is important to create a stick diagram which aids in giving a simplified view on how the layout would look like. It provides information about the inputs, outputs, wires, and contacts in each diffusion layer. For an XOR gate, the stick diagram will be as follows:

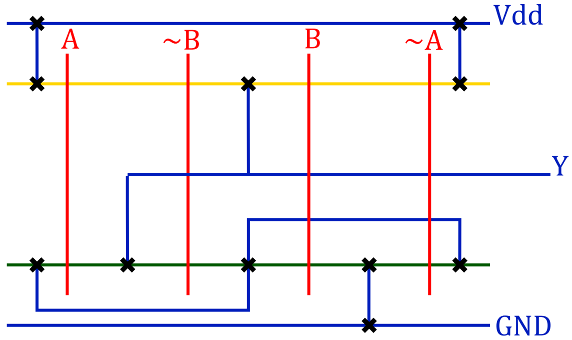


Figure 25. Stick Diagram for CMOS XOR Gate

By using the stick diagram from figure 25, the layout was easily implemented on GLADE, and it looks as follows:

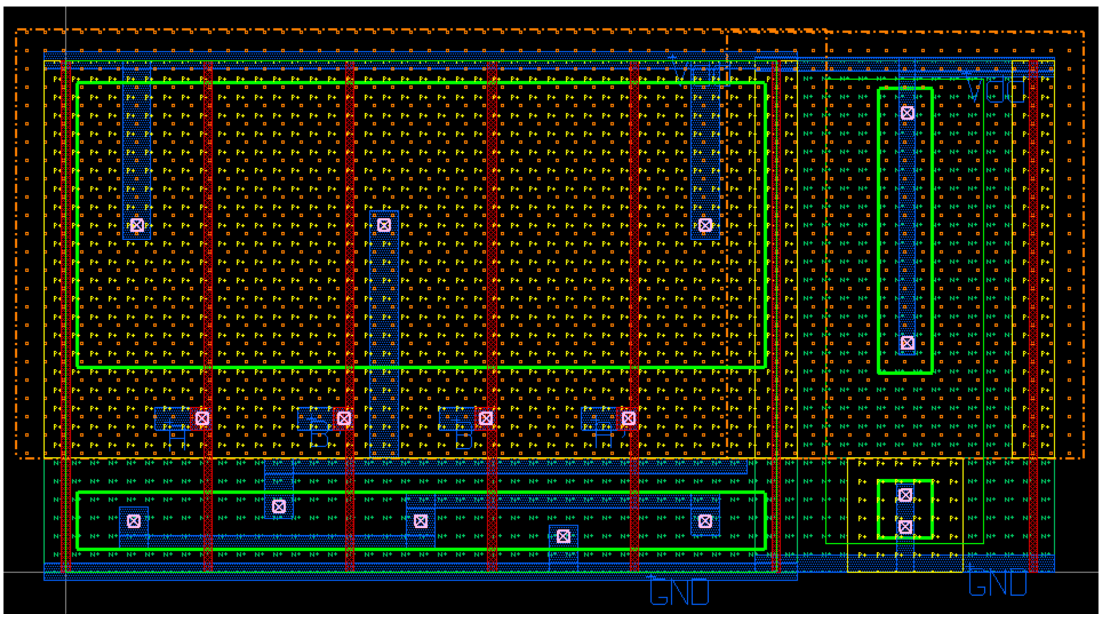
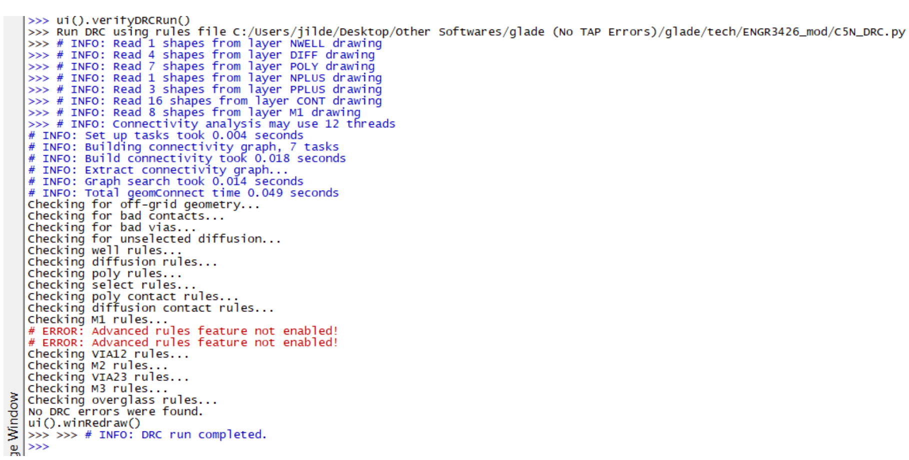


Figure 26. Layout for CMOS XOR Gate

To ensure that the layout satisfies all design rules, the DRC test is used. When applying the layout created in figure 26, it shows that all rules are adhered to with no design flaws.



```
>>> ui().verifyDRCRun()
>>> Run DRC using rules file C:/Users/jilde/Desktop/Other Softwares/glade (No TAP Errors)/glade/tech/ENGR3426_mod/C5N_DRC.py
>>> # INFO: Read 1 shapes from layer NWELL drawing
>>> # INFO: Read 4 shapes from layer DIFF drawing
>>> # INFO: Read 7 shapes from layer POLY drawing
>>> # INFO: Read 1 shapes from layer NPLUS drawing
>>> # INFO: Read 3 shapes from layer PPLUS drawing
>>> # INFO: Read 16 shapes from layer CONT drawing
>>> # INFO: Read 8 shapes from layer M1 drawing
>>> # INFO: Connectivity analysis may use 12 threads
# INFO: Set up tasks took 0.004 seconds
# INFO: Building connectivity graph, 7 tasks
# INFO: Build connectivity took 0.018 seconds
# INFO: Extract connectivity graph...
# INFO: Graph search took 0.014 seconds
# INFO: Total geomConnect time 0.049 seconds
Checking for off-grid geometry...
Checking for bad contacts...
Checking for bad vias...
Checking for unselected diffusion...
Checking well rules...
Checking diffusion rules...
Checking poly rules...
Checking select rules...
Checking poly contact rules...
Checking diffusion contact rules...
Checking M1 rules...
# ERROR: Advanced rules feature not enabled!
# ERROR: Advanced rules feature not enabled!
Checking VIA12 rules...
Checking M2 rules...
Checking VIA23 rules...
Checking M3 rules...
Checking overglass rules...
No DRC errors were found.
ui().winRedraw()
>>> >>> # INFO: DRC run completed.
>>>
```

Figure 27. DRC Test for XOR Gate Layout

After making sure that no DRC errors exist, it is required to create an extract of our layout using LPE test that will be then used to test schematic against the layout. The following two figures show the results after running LPE test:

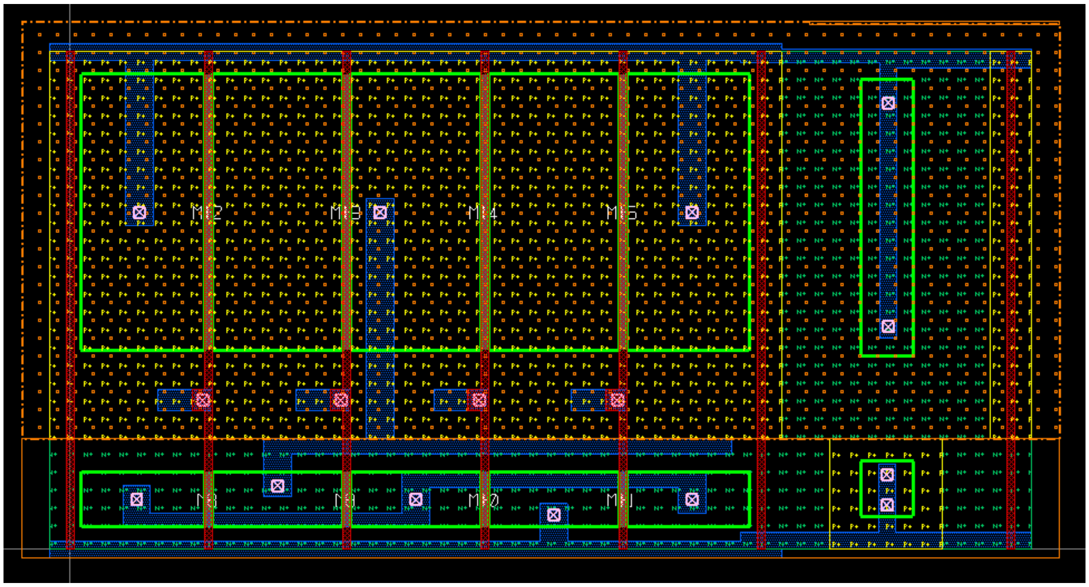
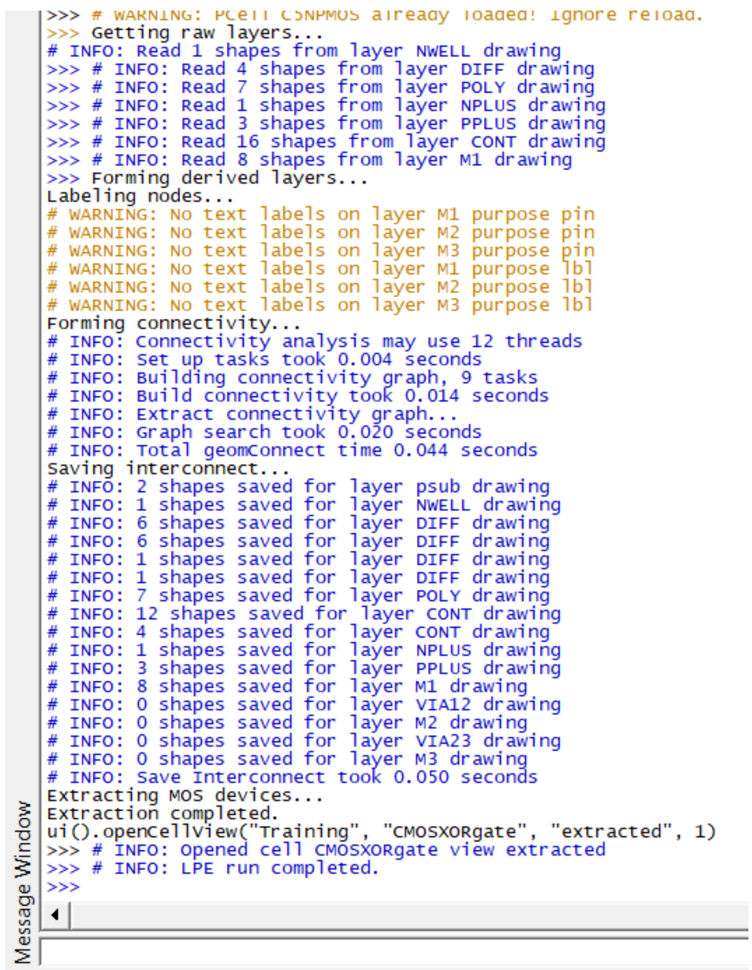


Figure 28. Extracted Layout for XOR Gate



```
>>> # WARNING: PCell C5NPMOS already loaded! Ignore reload.
>>> Getting raw layers...
# INFO: Read 1 shapes from layer NWELL drawing
>>> # INFO: Read 4 shapes from layer DIFF drawing
>>> # INFO: Read 7 shapes from layer POLY drawing
>>> # INFO: Read 1 shapes from layer NPLUS drawing
>>> # INFO: Read 3 shapes from layer PPLUS drawing
>>> # INFO: Read 16 shapes from layer CONT drawing
>>> # INFO: Read 8 shapes from layer M1 drawing
>>> Forming derived layers...
Labeling nodes...
# WARNING: No text labels on layer M1 purpose pin
# WARNING: No text labels on layer M2 purpose pin
# WARNING: No text labels on layer M3 purpose pin
# WARNING: No text labels on layer M1 purpose lbl
# WARNING: No text labels on layer M2 purpose lbl
# WARNING: No text labels on layer M3 purpose lbl
Forming connectivity...
# INFO: Connectivity analysis may use 12 threads
# INFO: Set up tasks took 0.004 seconds
# INFO: Building connectivity graph, 9 tasks
# INFO: Build connectivity took 0.014 seconds
# INFO: Extract connectivity graph...
# INFO: Graph search took 0.020 seconds
# INFO: Total geomConnect time 0.044 seconds
Saving interconnect...
# INFO: 2 shapes saved for layer psub drawing
# INFO: 1 shapes saved for layer NWELL drawing
# INFO: 6 shapes saved for layer DIFF drawing
# INFO: 6 shapes saved for layer DIFF drawing
# INFO: 1 shapes saved for layer DIFF drawing
# INFO: 1 shapes saved for layer DIFF drawing
# INFO: 7 shapes saved for layer POLY drawing
# INFO: 12 shapes saved for layer CONT drawing
# INFO: 4 shapes saved for layer CONT drawing
# INFO: 1 shapes saved for layer NPLUS drawing
# INFO: 3 shapes saved for layer PPLUS drawing
# INFO: 8 shapes saved for layer M1 drawing
# INFO: 0 shapes saved for layer VIA12 drawing
# INFO: 0 shapes saved for layer M2 drawing
# INFO: 0 shapes saved for layer VIA23 drawing
# INFO: 0 shapes saved for layer M3 drawing
# INFO: Save Interconnect took 0.050 seconds
Extracting MOS devices...
Extraction completed.
ui().openCellView("Training", "CMOSXORgate", "extracted", 1)
>>> # INFO: Opened cell CMOSXORgate view extracted
>>> # INFO: LPE run completed.
>>>
```


Figure 29. LPE Test Result for XOR Gate Layout

Then for the last test, layout versus schematic (LVS) test. This test is used to show the percentage of similarity between the layout and the schematic. In addition, it provides information about netlists, number of ports, etc. The result of running the LVS test is as follows:

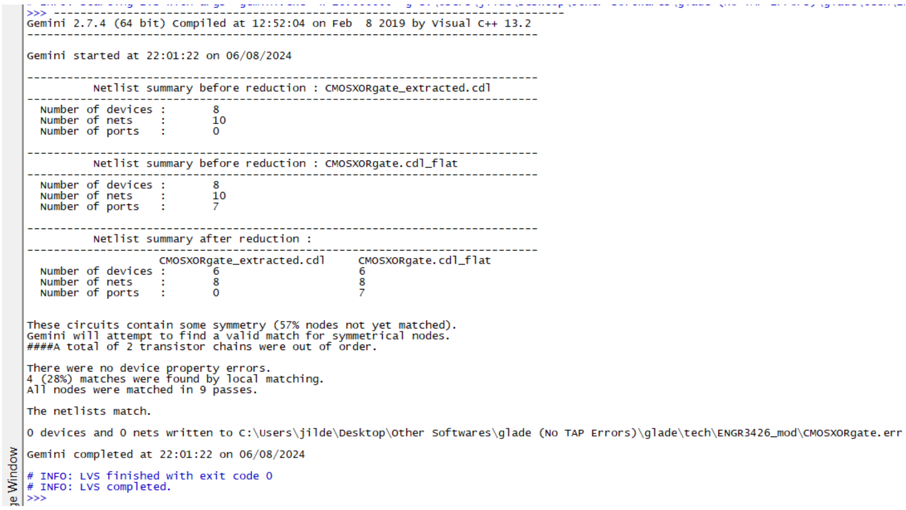


Figure 30. LVS Test Result for XOR Gate

2.4 Full Adder

After designing and testing each logic gate separately, it is now possible to construct the full adder using the designs used in sections 2.1, 2.2, 2.3. The schematic fn the full adder is as follows:

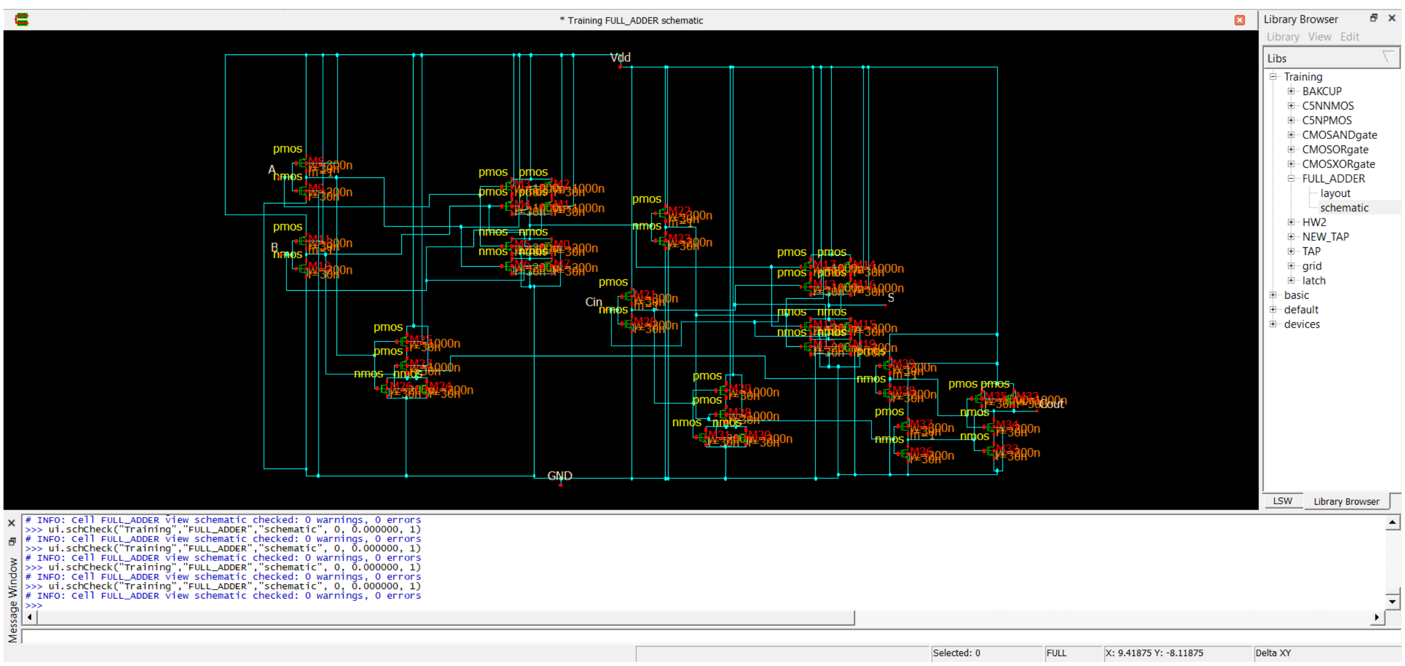


Figure 31. Schematic for Full Adder

In addition, the layout for the full adder is as follows:

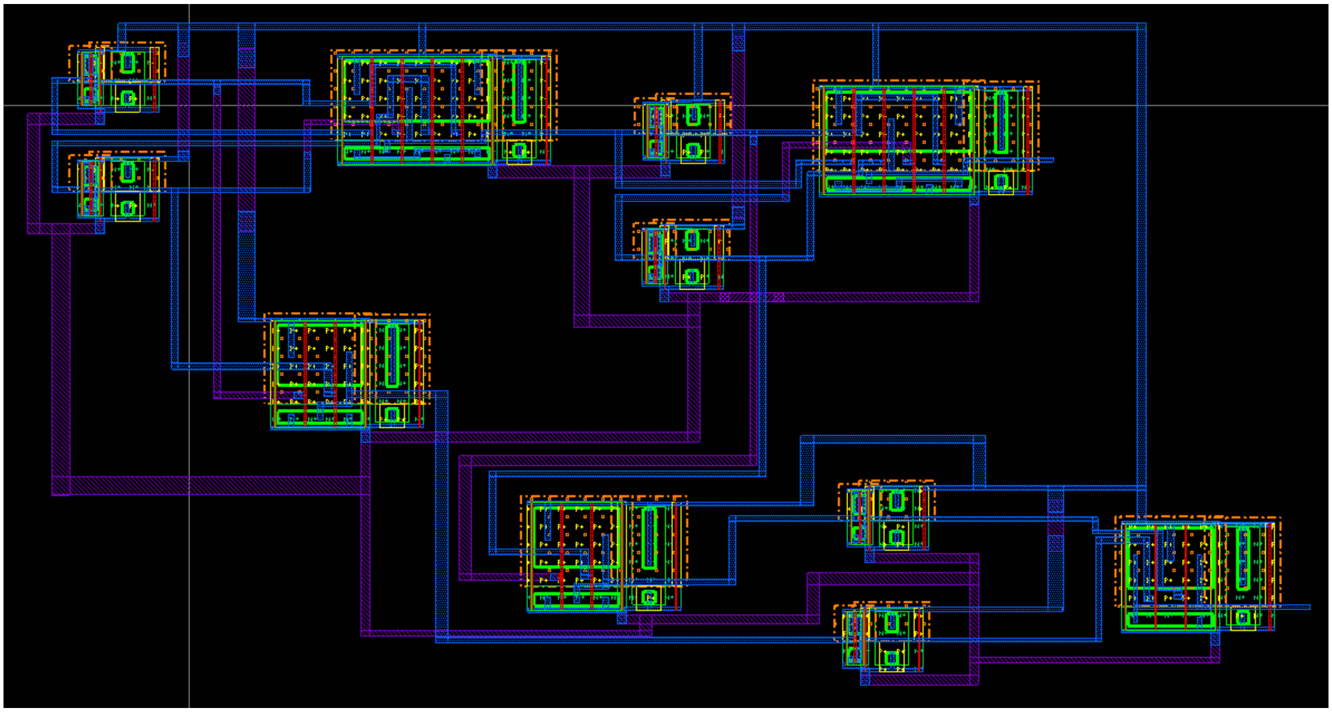
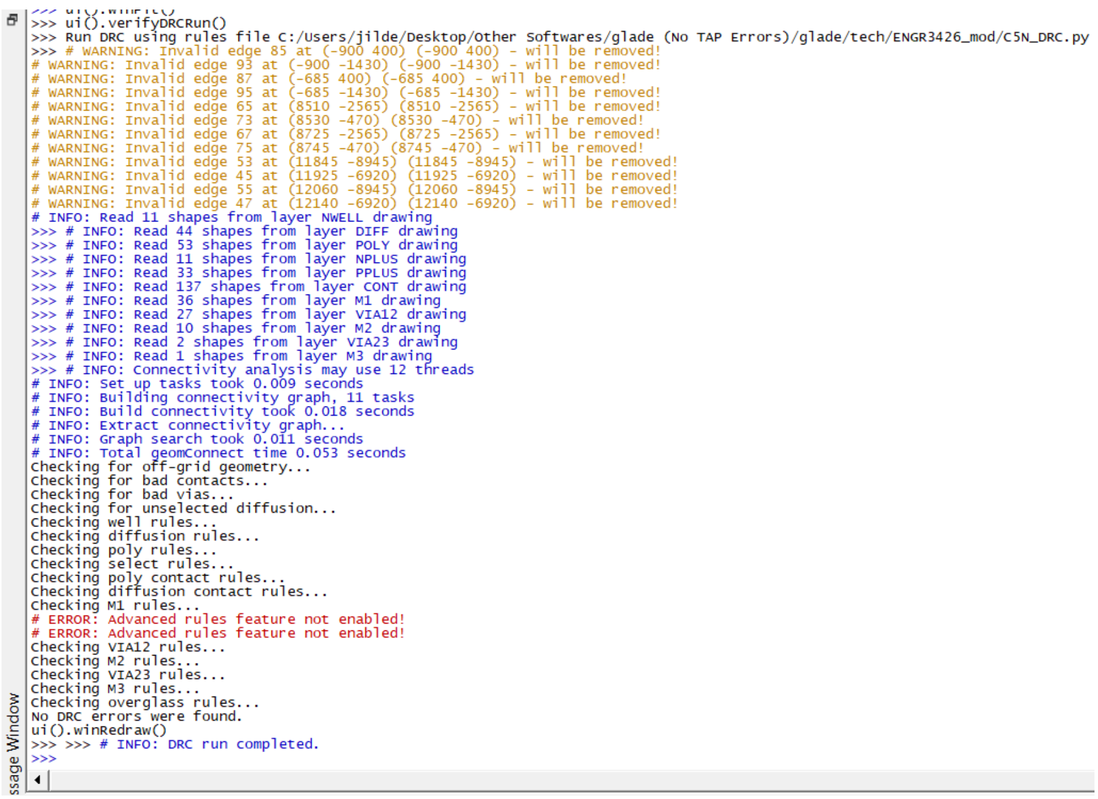


Figure 32. Layout for Full Adder

3 RESULTS AND DISCUSSIONS

The following shows the test results of applying the DRC test on the full adder layout:



```
>>> ui().verifyDRCRun()
>>> Run DRC using rules file C:/Users/jilde/Desktop/Other Softwares/glade (No TAP Errors)/glade/tech/ENGR3426_mod/C5N_DRC.py
>>> # WARNING: Invalid edge 85 at (-900 400) (-900 400) - will be removed!
# WARNING: Invalid edge 93 at (-900 -1430) (-900 -1430) - will be removed!
# WARNING: Invalid edge 87 at (-685 400) (-685 400) - will be removed!
# WARNING: Invalid edge 95 at (-685 -1430) (-685 -1430) - will be removed!
# WARNING: Invalid edge 65 at (8510 -2565) (8510 -2565) - will be removed!
# WARNING: Invalid edge 73 at (8530 -470) (8530 -470) - will be removed!
# WARNING: Invalid edge 67 at (8725 -2565) (8725 -2565) - will be removed!
# WARNING: Invalid edge 75 at (8745 -470) (8745 -470) - will be removed!
# WARNING: Invalid edge 53 at (11845 -8945) (11845 -8945) - will be removed!
# WARNING: Invalid edge 45 at (11925 -6920) (11925 -6920) - will be removed!
# WARNING: Invalid edge 55 at (12060 -8945) (12060 -8945) - will be removed!
# WARNING: Invalid edge 47 at (12140 -6920) (12140 -6920) - will be removed!
# INFO: Read 11 shapes from layer NWELL drawing
>>> # INFO: Read 44 shapes from layer DIFF drawing
>>> # INFO: Read 53 shapes from layer POLY drawing
>>> # INFO: Read 11 shapes from layer NPLUS drawing
>>> # INFO: Read 33 shapes from layer PPLUS drawing
>>> # INFO: Read 137 shapes from layer CONT drawing
>>> # INFO: Read 36 shapes from layer M1 drawing
>>> # INFO: Read 27 shapes from layer VIA12 drawing
>>> # INFO: Read 10 shapes from layer M2 drawing
>>> # INFO: Read 2 shapes from layer VIA23 drawing
>>> # INFO: Read 1 shapes from layer M3 drawing
>>> # INFO: Connectivity analysis may use 12 threads
# INFO: Set up tasks took 0.009 seconds
# INFO: Building connectivity graph, 11 tasks
# INFO: Build connectivity took 0.018 seconds
# INFO: Extract connectivity graph...
# INFO: Graph search took 0.011 seconds
# INFO: Total geomConnect time 0.053 seconds
Checking for off-grid geometry...
Checking for bad contacts...
Checking for bad vias...
Checking for unselected diffusion...
Checking well rules...
Checking diffusion rules...
Checking poly rules...
Checking select rules...
Checking poly contact rules...
Checking diffusion contact rules...
Checking M1 rules...
# ERROR: Advanced rules feature not enabled!
# ERROR: Advanced rules feature not enabled!
Checking VIA12 rules...
Checking M2 rules...
Checking VIA23 rules...
Checking M3 rules...
Checking overglass rules...
No DRC errors were found.
ui().winRedraw()
>>> >>> # INFO: DRC run completed.
>>>
```

Figure 33. DRC Test for Full Adder Layout

The following shows the test results of running the LPE test:

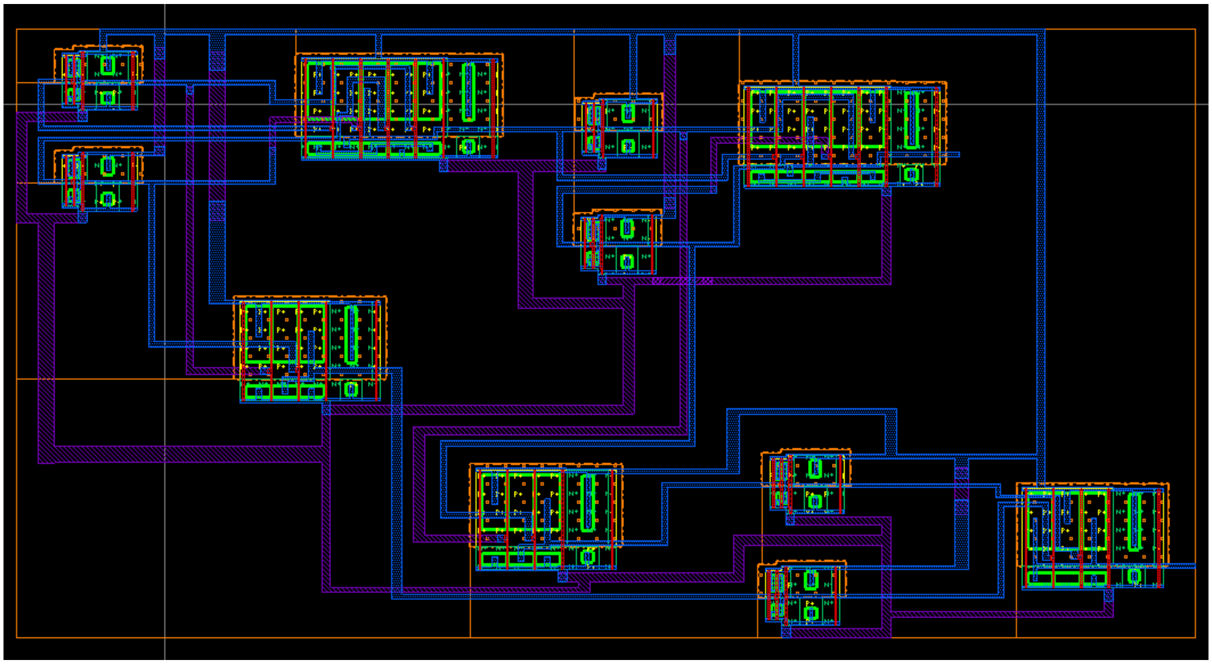
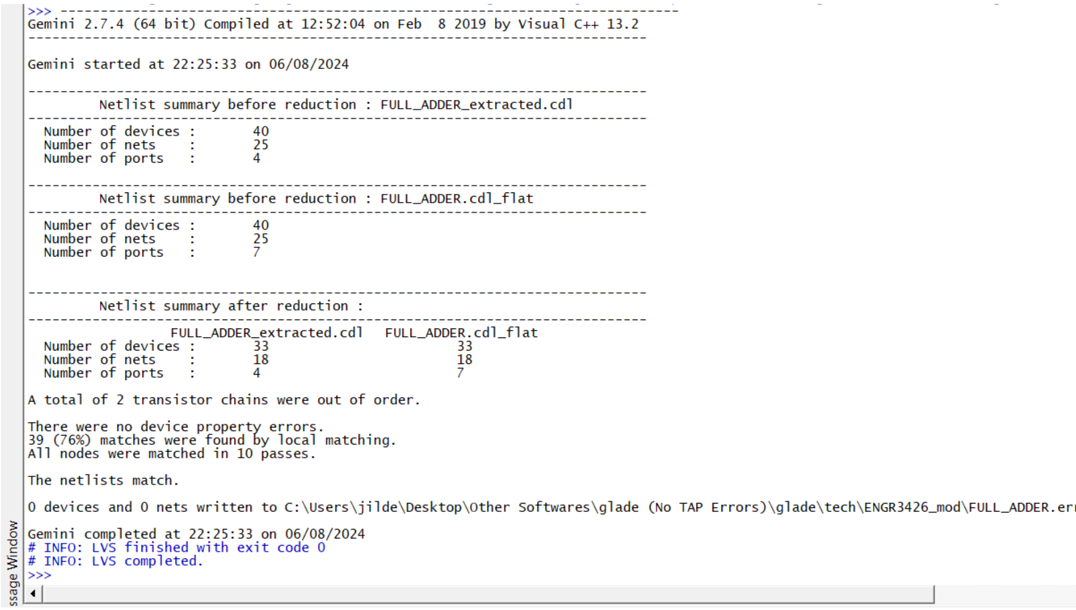


Figure 34. Extracted Layout for Full Adder

```
# INFO: Read 11 shapes from layer NWELL drawing
>>> # INFO: Read 44 shapes from layer DIFF drawing
>>> # INFO: Read 53 shapes from layer POLY drawing
>>> # INFO: Read 11 shapes from layer NPLUS drawing
>>> # INFO: Read 33 shapes from layer PPLUS drawing
>>> # INFO: Read 137 shapes from layer CONT drawing
>>> # INFO: Read 36 shapes from layer M1 drawing
>>> # INFO: Read 27 shapes from layer VIA12 drawing
>>> # INFO: Read 10 shapes from layer M2 drawing
>>> # INFO: Read 2 shapes from layer VIA23 drawing
>>> # INFO: Read 1 shapes from layer M3 drawing
>>> Forming derived layers...
Labeling nodes...
# WARNING: No text labels on layer M3 purpose pin
# WARNING: No text labels on layer M1 purpose lbl
# WARNING: No text labels on layer M2 purpose lbl
# WARNING: No text labels on layer M3 purpose lbl
Forming connectivity...
# INFO: Connectivity analysis may use 12 threads
# INFO: Set up tasks took 0.004 seconds
# INFO: Building connectivity graph, 13 tasks
# INFO: Build connectivity took 0.023 seconds
# INFO: Extract connectivity graph...
# INFO: Graph search took 0.022 seconds
# INFO: Total geomConnect time 0.056 seconds
Saving interconnect...
# INFO: 12 shapes saved for layer psub drawing
# INFO: 11 shapes saved for layer NWELL drawing
# INFO: 42 shapes saved for layer DIFF drawing
# INFO: 42 shapes saved for layer DIFF drawing
# INFO: 11 shapes saved for layer DIFF drawing
# INFO: 11 shapes saved for layer DIFF drawing
# INFO: 53 shapes saved for layer POLY drawing
# INFO: 117 shapes saved for layer CONT drawing
# INFO: 20 shapes saved for layer CONT drawing
# INFO: 11 shapes saved for layer NPLUS drawing
# INFO: 33 shapes saved for layer PPLUS drawing
# INFO: 36 shapes saved for layer M1 drawing
# INFO: 27 shapes saved for layer VIA12 drawing
# INFO: 10 shapes saved for layer M2 drawing
# INFO: 2 shapes saved for layer VIA23 drawing
# INFO: 1 shapes saved for layer M3 drawing
# INFO: Save Interconnect took 0.085 seconds
Extracting MOS devices...
Extraction completed.
ui().openCellView("Training", "FULL_ADDER", "extracted", 1)
>>> # INFO: Opened cell FULL_ADDER view extracted
>>> # INFO: LPE run completed.
>>>
```

Figure 35. LPE Test Result for Full Adder Layout

The following is the result of running the LVS test:



```
>>> ------------------------------------------------------------------
Gemini 2.7.4 (64 bit) Compiled at 12:52:04 on Feb  8 2019 by Visual C++ 13.2
--------------------------------------------------------------------------

Gemini started at 22:25:33 on 06/08/2024

--------------------------------------------------------------------------
         Netlist summary before reduction : FULL_ADDER_extracted.cdl
--------------------------------------------------------------------------
  Number of devices :        40
  Number of nets    :        25
  Number of ports   :        4

--------------------------------------------------------------------------
         Netlist summary before reduction : FULL_ADDER.cdl_flat
--------------------------------------------------------------------------
  Number of devices :        40
  Number of nets    :        25
  Number of ports   :        7


--------------------------------------------------------------------------
         Netlist summary after reduction :
--------------------------------------------------------------------------
                  FULL_ADDER_extracted.cdl   FULL_ADDER.cdl_flat
  Number of devices :        33                         33
  Number of nets    :        18                         18
  Number of ports   :        4                          7

A total of 2 transistor chains were out of order.

There were no device property errors.
39 (76%) 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\jilde\Desktop\Other Softwares\glade (No TAP Errors)\glade\tech\ENGR3426_mod\FULL_ADDER.err

Gemini completed at 22:25:33 on 06/08/2024
# INFO: LVS finished with exit code 0
# INFO: LVS completed.
>>>
```

Figure 36. LVS Test Result for Full Adder

The result given by the LVS test shows that the full adder has been designed properly with no design flaws.

4 CONCLUSIONS

This project demonstrates the effective use of Glade for IC design and reinforces the importance of thorough verification steps like DRC and LVS to ensure the design is reliable and manufacturable. The full adder design meets the functional requirements and is ready for fabrication.

5 REFERENCES

[1] https://peardrop.co.uk/

[2] https://www.youtube.com/watch?v=fKJpa9LJ-cQ

[3] https://www.youtube.com/watch?v=vGsNH8Ocz8s