CMOS Design Project: 2-Bit Comparator

Indian Institute of Information Technology,

Nagpur

ECL 312: CMOS Design

A Project Report on: 2-bit comparator using CMOS

Submitted By:

Ayush Ambatkar (BT22ECI005)

Under the Guidance of:

Prof. Paritosh Peshwe

Department of Electronics and Communication

Project Overview

The 2-bit digital comparator project is a CMOS-based circuit designed to compare two 2-bit binary inputs, A and B, and output three signals indicating their relationship: whether A is less than, equal to, or greater than B. Using CMOS logic gates—AND, OR, NAND, NOR, XNOR, and inverters—the circuit combines these components to generate accurate comparison outputs. This design is essential in digital systems for data comparison tasks and logical decision-making in microprocessors and digital controllers. The netlist provided serves as the foundation for building the comparator in a simulation environment and guides the layout design for fabrication.

Objectives

1. Design a CMOS-based circuit to compare two 2-bit binary numbers (A and B).
2. Generate three output signals: A<B, A=B, and A>B, to indicate the relationship between A and B.
3. Utilize basic CMOS components, including AND, OR, NAND, NOR, XNOR gates, and inverters, for the comparison logic.
4. Implement the circuit in a simulation environment to verify functionality and performance.
5. Develop a layout suitable for fabrication using CMOS design rules and tools like Microwind.

Design Description

Circuit Operation

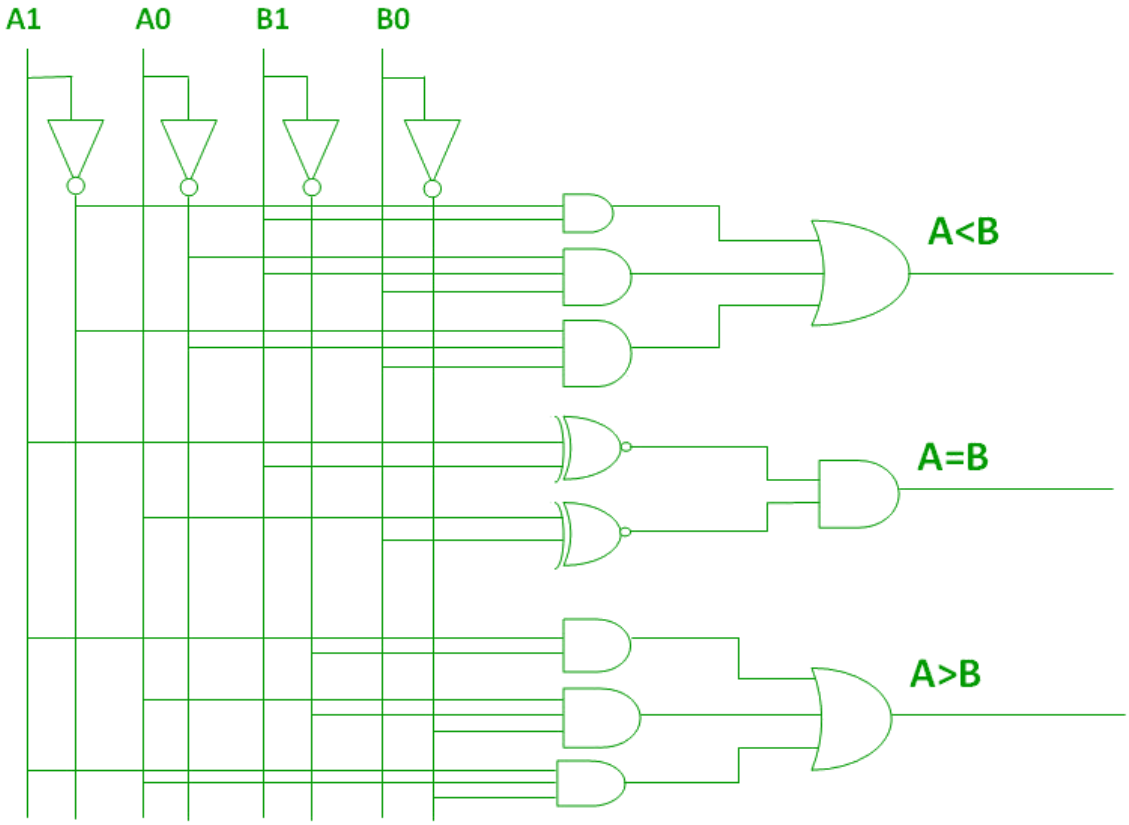
- The circuit takes two 2-bit binary inputs, A (A1A0) and B (B1B0), and compares them bit by bit.
- For the A=B condition, two XNOR gates check if corresponding bits are equal (A1=B1 and A0=B0), and an AND gate combines these results to confirm equality.
- For A<B, multiple AND gates check conditions where B has higher bits than A (e.g., A1<B1, or if A1=B1, then A0<B0).
- For A>B, similar AND gates verify conditions where A has higher bits than B.
- Finally, the three outputs—A<B, A=B, and A>B—indicate the relationship between A and B based on these logic combinations.

Logic Implementation

The circuit implements the following logic:

- Equality (A=B): The A=B condition is achieved by using XNOR gates to check if each pair of bits (A1 with B1 and A0 with B0) are equal. The outputs of these XNOR gates are combined using an AND gate to confirm equality.
- Less Than (A<B): The A<B condition is implemented with AND gates to evaluate specific cases where B is greater. This includes conditions like A1<B1, or if A1=B1, then A0<B0.
- Greater Than (A>B): Similarly, the A>B condition uses AND gates to determine cases where A is greater than B. These gates check scenarios like A1>B1, or if A1=B1, then A0>B0.
- Combining Results: The outputs from these gates (A=B, A<B, and A>B) represent the final comparison and are used as the three output signals of the circuit.
- CMOS Gates: Each logical operation (AND, OR, XNOR) is implemented with CMOS gate structures, ensuring that the circuit is optimized for power efficiency and compact layout in CMOS technology.

Logic Gate Circuit Diagram:



Simulation Files

Below are snippets of the circuit files used for simulation in **WinSpice**:

Winspice circuit file:

```
***2 Bit Comparator***
.model nmod nmos level=54 version=4.7
.model pmod pmos level=54 version=4.7

.ic v(13)=0 v(17)=0 v(20)=0

* Inverter Subcircuit
.subckt inverter in vdd out
M1 out in 0 0 nmod w=100u l=10u
M2 out in vdd vdd pmod w=200u l=10u
Cout out 0 1p
.ends

* AND Gate Subcircuit
.subckt and2 a b output vdd
M1 out a n1 n1 nmod w=100u l=10u
M2 n1 b 0 0 nmod w=100u l=10u
M3 out a vdd vdd pmod w=100u l=10u
M4 out b vdd vdd pmod w=100u l=10u
Xout out vdd output inverter
.ends

* 3 i/p AND Gate Subcircuit
.subckt and3 a b c output vdd
M1 out a n1 n1 nmod w=100u l=10u
```

```
M2 n1 b n2 n2 nmod w=100u l=10u
M3 n2 c 0 0 nmod w=100u l=10u
M4 out a vdd vdd pmod w=100u l=10u
M5 out b vdd vdd pmod w=100u l=10u
M6 out c vdd vdd pmod w=100u l=10u
Xout out vdd output inverter
.ends

* NAND Gate Subcircuit
.subckt nand2 a b out vdd
M1 out a n1 n1 nmod w=100u l=10u
M2 n1 b 0 0 nmod w=100u l=10u
M3 out a vdd vdd pmod w=200u l=10u
M4 out b vdd vdd pmod w=200u l=10u
.ends

* OR Gate Subcircuit
.subckt or2 a b out vdd
M1 n1 b 0 0 nmod w=100u l=10u
M2 n1 a 0 0 nmod w=100u l=10u
M3 n1 b n2 n2 pmod w=200u l=10u
M4 n2 a vdd vdd pmod w=200u l=10u
Xa_inv n1 vdd out inverter
.ends

* 3 i/p OR Gate Subcircuit
.subckt or3 a b c out vdd
```

```
M1 n1 b 0 0 nmod w=100u l=10u
M2 n1 a 0 0 nmod w=100u l=10u
M3 n1 c 0 0 nmod w=100u l=10u
M4 n1 c n2 n2 pmod w=200u l=10u
M5 n2 b n3 n3 pmod w=200u l=10u
M6 n3 a vdd vdd pmod w=200u l=10u
Xa_inv n1 vdd out inverter
.ends

* NOR Gate Subcircuit
.subckt nor2 a b out vdd
M1 out b 0 0 nmod w=100u l=10u
M2 out a 0 0 nmod w=100u l=10u
M3 out b n2 n2 pmod w=200u l=10u
M4 n2 a vdd vdd pmod w=200u l=10u
.ends

.subckt tg in out sel selb
M1 out sel in in nmod w=100u l=10u
M2 out selb in in pmod w=200u l=10u
.model nmod nmos level=54 version=4.7
.model pmod pmos level=54 version=4.7
.ends

.subckt xnor a b out ab bb vdd
M1 out ab n1 n1 nmod w=100u l=10u
M2 out a n2 n2 nmod w=100u l=10u
```

```
M3 n1 b 0 0 nmod w=100u l=10u
M4 n2 bb 0 0 nmod w=100u l=10u
M5 out ab n3 n3 pmod w=200u l=10u
M6 out b n3 n3 pmod w=200u l=10u
M7 n3 a vdd vdd pmod w=200u l=10u
M8 n3 bb vdd vdd pmod w=200u l=10u
.model nmod nmos level=54 version=4.7
.model pmod pmos level=54 version=4.7
.ends

vdd 9 0 dc 5v
Va0 1 0 dc 0v
Vb0 2 0 dc 0v
Va1 3 0 dc 5v
Vb1 4 0 dc 5v
Xa0i 1 9 5 inverter
Xb0i 2 9 6 inverter
Xa1i 3 9 7 inverter
Xb1i 4 9 8 inverter

*a<b at 13*
xalb 7 4 10 9 and2
xallb 5 2 4 11 9 and3
xalllb 2 5 7 12 9 and3
xalessb 10 11 12 13 9 or3

*a>b at 17*
```

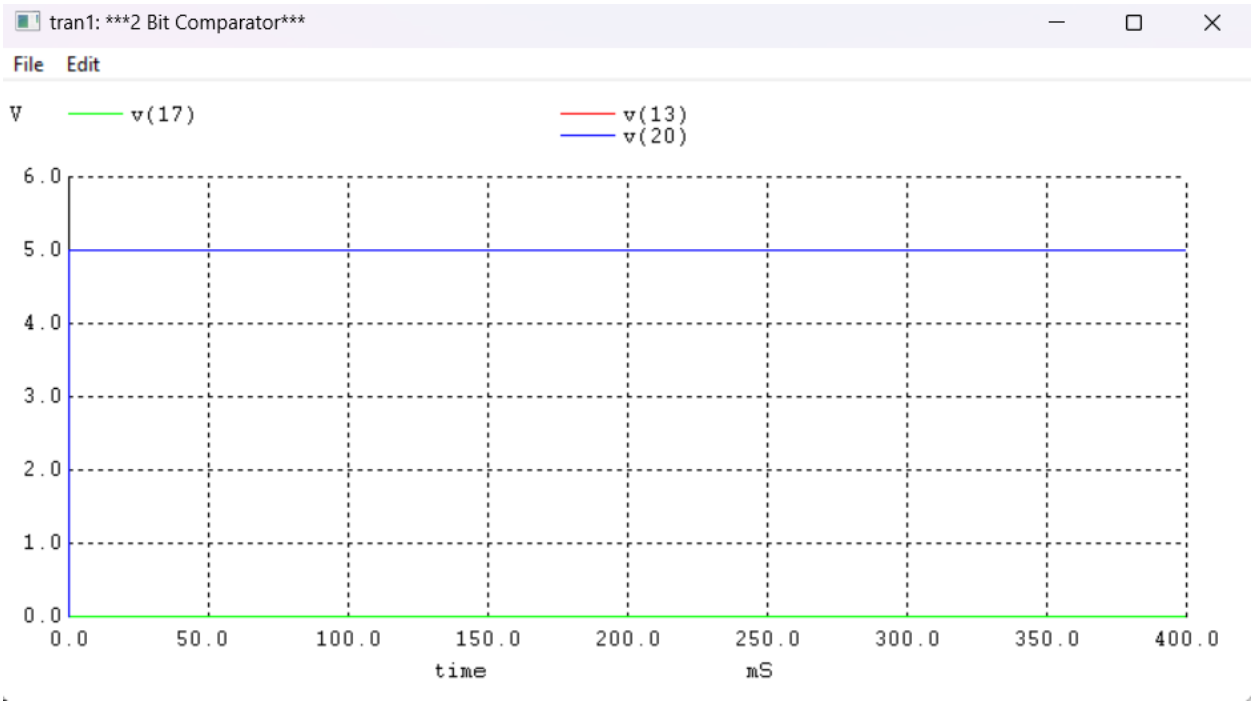
```
xagb 3 8 14 9 and2
xaggb 1 3 6 15 9 and3
xagggb 1 6 8 16 9 and3
xagreatb 14 15 16 17 9 or3

*a=b at 20*
xaeb 3 4 18 7 8 9 xnor
xaeeb 1 2 19 5 6 9 xnor
xaequalb 18 19 20 9 and2

.tran 0.1m 400m
.control
run
plot v(13) v(17) v(20)
.endc
.end
```

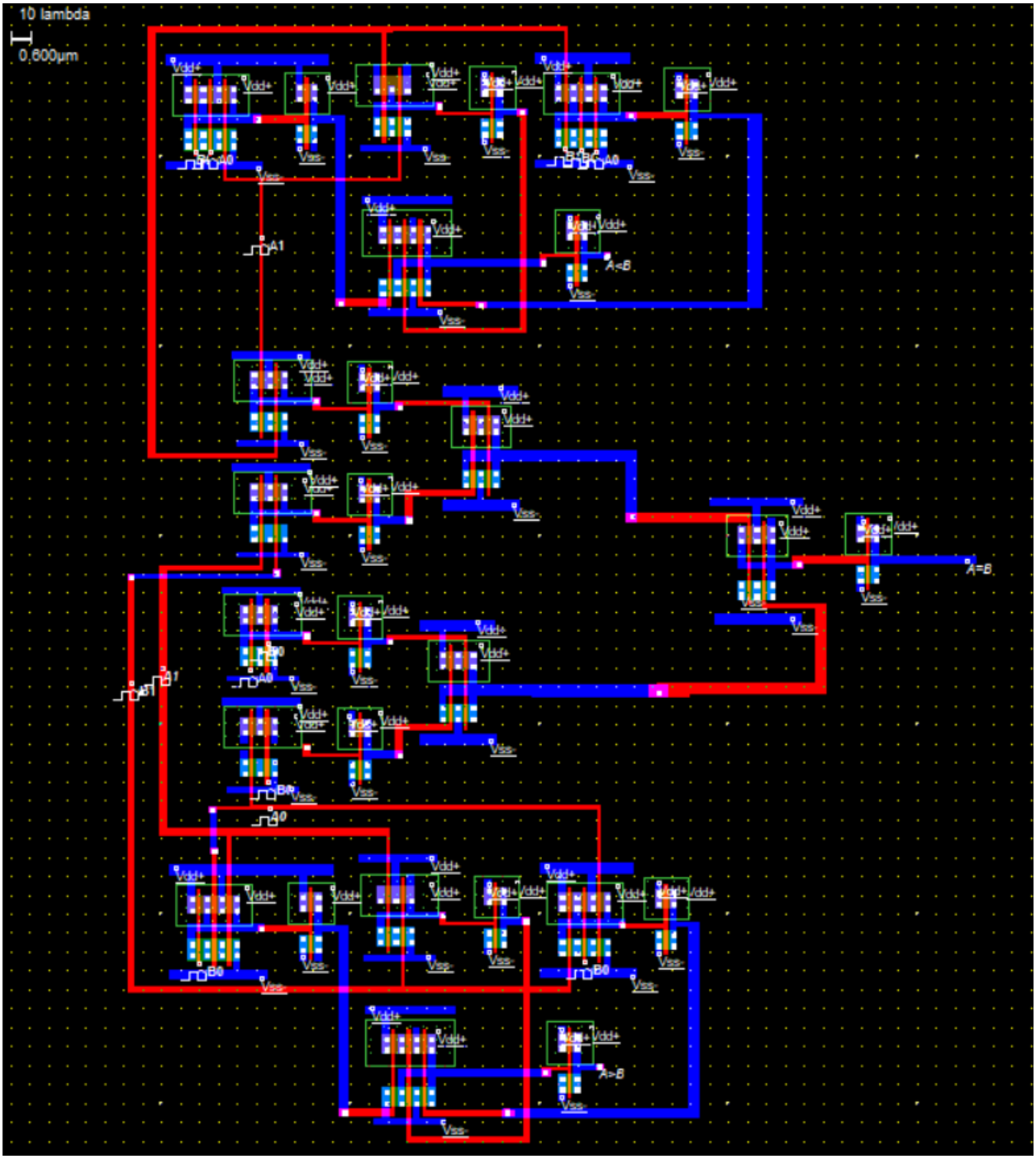
Winspice Output:

For DC input:

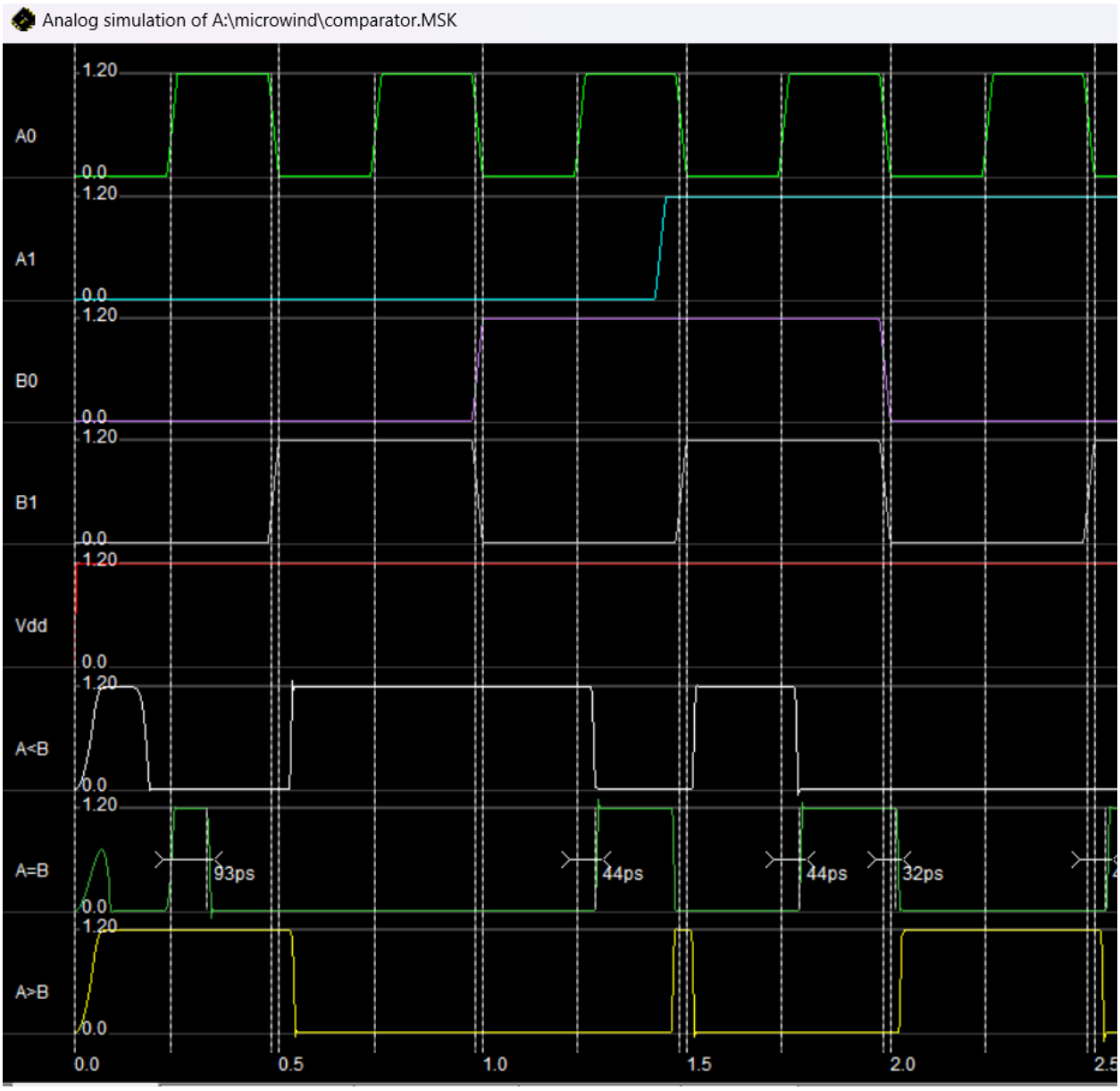


Layout and Waveforms

Circuit Waveform:


10 lambda
0,600µm
Vdd+
Vss-
A1
A0
B1
B0
A<B
A=B
A>B

Output Waveforms:



Use Cases and Applications?

- Threshold Detection: In sensor systems, compares data (e.g., temperature) against set thresholds to trigger actions (e.g., activate a fan if temperature is high).
- Control Systems: Used in microcontrollers for decision-making, such as enabling or disabling processes based on conditions.

- Sorting Operations: Helps in sorting or selecting smaller binary numbers in simple digital circuits.
- Binary Counting: Aids in counting circuits by determining if a counter has reached a certain binary value.
- Alarm Systems: Can trigger alerts when a value falls below or exceeds a predefined limit.

Implementation Challenges and Solutions:

1. **Power Consumption**

Challenge: Comparators require multiple logic gates, which can lead to high power consumption, especially when many gates switch simultaneously.

Solution:

- Use CMOS technology, known for low static power consumption.
- Optimize transistor sizing to balance power and speed, ensuring only necessary power is used for each operation.

2. **Area Optimization**

Challenge: The layout area can become large when multiple gates are used, especially in designs where compactness is essential.

Solution:

- Share common logic gates or inverters where possible to reduce the overall gate count.
- Arrange the layout efficiently by minimizing spacing between gates while adhering to design rules, reducing the circuit footprint.

3. **Signal Integrity and Noise**

Challenge: Close placement of components can lead to coupling and noise, potentially affecting signal integrity and causing inaccurate outputs.

Solution:

- Implement proper isolation between polysilicon, metal layers, and other layers.

- Use grounding and spacing between layers or critical components to reduce interference.

- **Speed and Propagation Delay**

Challenge: Comparators must perform fast, especially when part of a larger system. Delay through multiple gates can affect timing.

Solution:

- Optimize transistor width-to-length ratios to increase speed in critical paths, like in the XNOR gates used for equality checking.
- Balance speed and power by selectively increasing transistor sizes where faster switching is needed.

- **Design Rule Compliance**

Challenge: Violating design rules during layout can cause manufacturing errors or circuit malfunctions.

Solution:

- Carefully follow CMOS design rules, especially for minimum spacing and layer alignment.
- Run design rule checks (DRCs) frequently during layout to catch issues early, ensuring a compliant and functional layout.

Future Work:

Potential improvements and extensions could include:

1. Extending the design to handle 8-bit inputs
2. Optimizing the circuit for lower power consumption
3. Implementing error correction capabilities
4. Optimizing the circuit for high frequency inputs

Conclusion:

The 2-bit digital comparator project successfully demonstrates the use of CMOS technology for comparing binary numbers. Key takeaways include:

- Efficient Functionality: The circuit accurately compares two 2-bit binary inputs, generating outputs for A<B, A=B, and A>B.
- Low Power and Compact Design: CMOS logic gates like AND, OR, and XNOR ensure low power consumption and minimal area usage.
- Challenges Addressed: Power optimization, area constraints, and signal integrity were effectively managed through careful design and transistor sizing.
- Foundation for Complex Systems: This simple comparator serves as a building block for more advanced digital circuits in embedded and digital electronics.
