CMOS VLSI Logic Designs using gpdk90

Presented by

Name: Ahasan Ullah Khalid

Institution: Chittagong University of Engineering and Technology (CUET)

Department: Electronics & Telecommunication Engineering (ETE)

GitHub: github.com/aukhalid

Website: aukhalid.vercel.app

Project Overview

This project contains the design and implementation of fundamental digital logic gates, a complete set of CMOS standard cells and digital system designs implemented in **Cadence Virtuoso** using the **gpdk90 open-source PDK**. Each cell has been designed at transistor-level, laid out following DRC/LVS rules, symbolized for hierarchy, and simulated in **ADE L** to verify correctness against theoretical truth tables.

The work demonstrates the full VLSI design flow for both fundamental building blocks and advanced digital circuits, including:

- Transistor-level schematic design
- DRC-clean layout implementation
- Symbol creation for hierarchical use
- Circuit simulation in ADE L
- Verification against theoretical truth tables

Tools and Technology

- Cadence Virtuoso (schematic, layout, simulation in ADE L)
- gpdk90 Open-Source PDK
- CMOS VLSI Design Principles

Project Structure

The Project is divided into two major sections:

Standard Cells

- Inverter (NOT Gate)
- 2-input NAND Gate (NAND2X1)
- 2-input AND Gate (AND2X1)
- 2-input NOR Gate (NOR2X1)
- 2-input OR Gate (OR2X1)
- M1_NWELL
- M1_PSUB

Digital Blocks and Systems

- Half Adder (HA)
- Full Adder (FA)
- 1 × 2 De-Multiplexer
- 1 × 8 De-Multiplexer

Section 1: Standard Cells

Each standard cell is presented with the following format:

- Objective
- Description
- Design Steps
- Schematic Diagram
- Layout
- Symbol
- Simulation & Verification

CMOS Inverter (NOT Gate)

Objective:

The objective of designing the inverter (NOT gate) is to create the simplest form of logic gate that outputs the complement of the input signal. This inverter is designed, laid out, and verified using the GPDK90 PDK in **Cadence Virtuoso**

Description:

The CMOS inverter consists of a PMOS transistor at the top and an NMOS transistor at the bottom. When the input is logic 1, the NMOS conducts, and the output is logic 0. Conversely, when the input is 0, the PMOS conducts, producing logic 1.

- **Input A** → single binary input
- **Output Y** → inverted value of input A

The truth table is:

A	**Y**
0	1
1	0

The inverter is realized using one PMOS and one NMOS transistor in CMOS configuration.

Design Steps:

Schematic Design

- A PMOS transistor was placed with its source connected to VDD (1.2 V) and its drain connected to the output node.
- An NMOS transistor was placed with its source connected to GND (0 V) and its drain connected to the same output node.
- Both gates were connected together and labeled as the input A.
- The output node was labeled as Y.

Layout Design

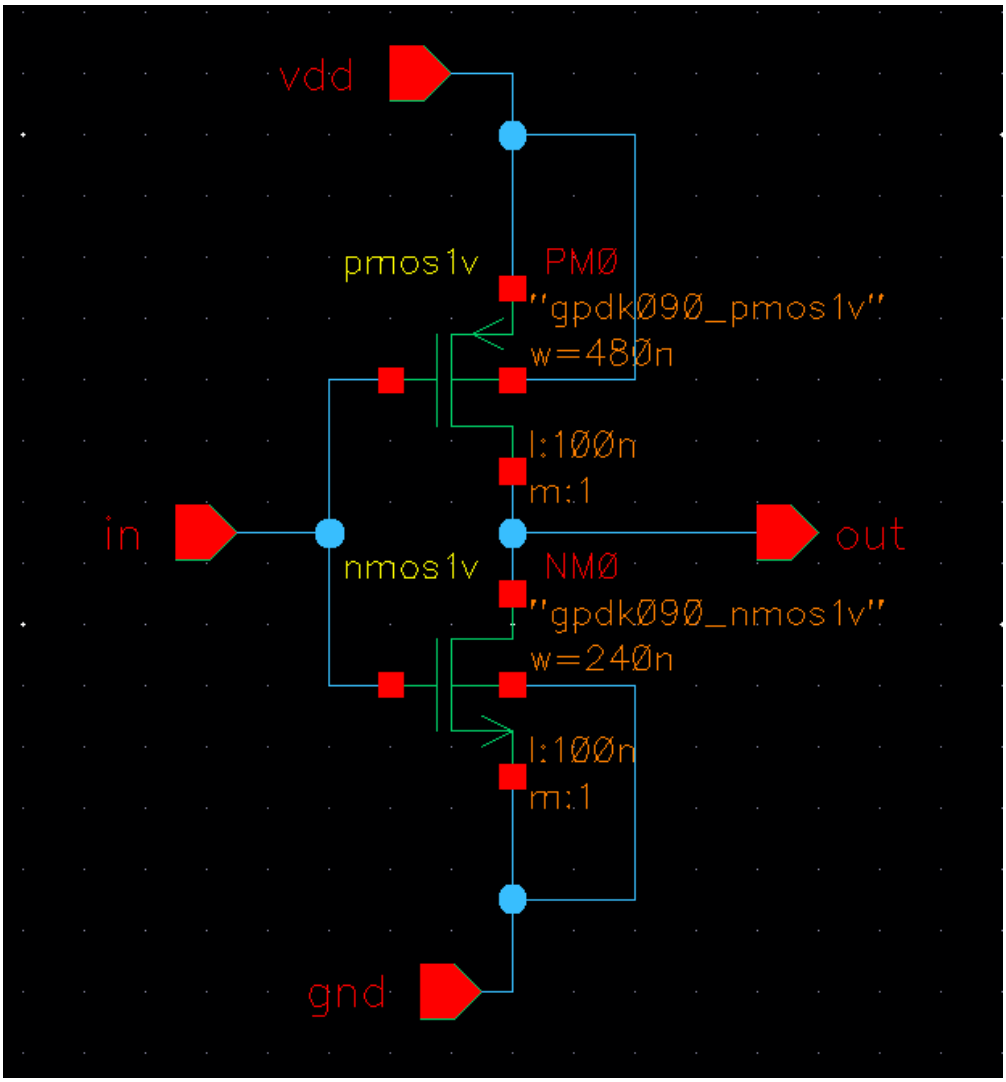
- Diffusion layers for the PMOS and NMOS transistors were drawn in their respective N-well and P-substrate regions.
- Metal layers were routed to connect:
- PMOS source to VDD

- NMOS source to GND
- Drains of PMOS and NMOS to the output node Y
- Both gates tied to input A
- DRC and LVS checks were followed to ensure layout correctness.

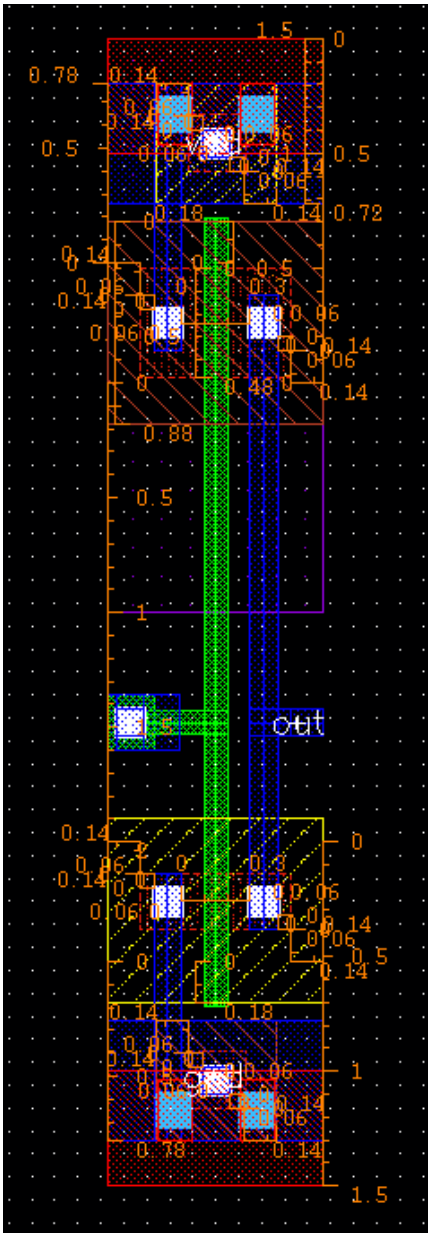
Symbol Creation

- A symbol view of the inverter was generated with one input pin (A), one output pin (Y), and the power pins (VDD, GND).

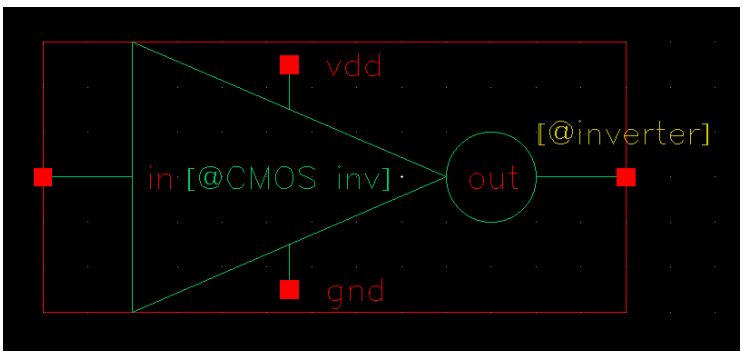
Schematic Diagram:



Layout:



Symbol:



Simulation and Verification:

- ADE L in Cadence Virtuoso was opened.
- The following testbench conditions were applied:
  - VDD = 1.2 V (DC source was connected to the power pin).
  - GND = 0 V (reference ground was connected).
  - Input A = pulse source ranging from 0 V to 1.2 V, with rise/fall times set for clear transitions.

Transient Analysis was run for a sufficient time (e.g., 20 ns) to observe multiple switching cycles.

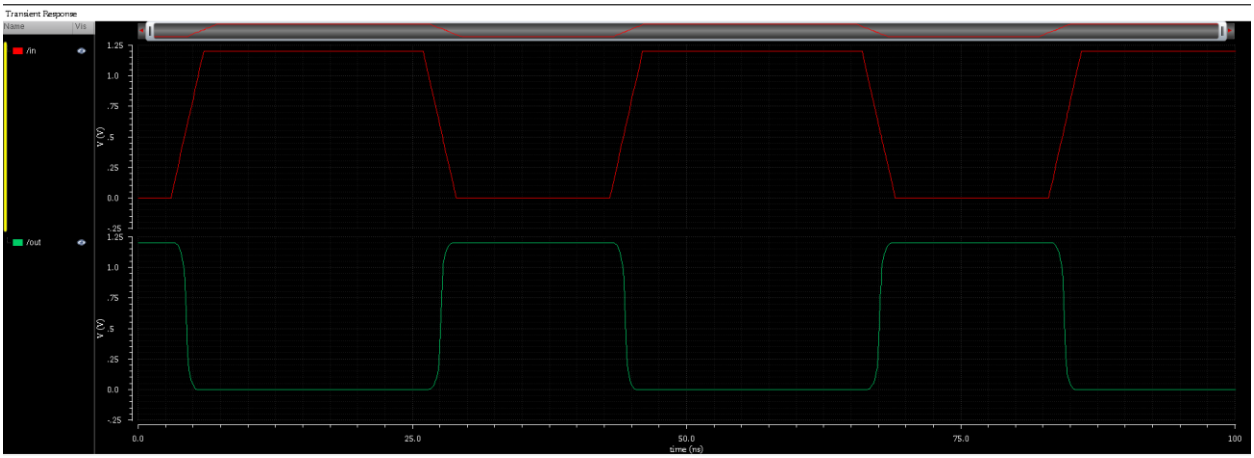
It was verified that:

- When A = 0 V, output Y = 1.2 V (HIGH).
- When A = 1.2 V, output Y = 0 V (LOW).

Simulation waveforms were compared with the truth table to confirm correct inverter operation.

Truth Table:

A	Y
0	1
1	0



NAND2X1 (2-Input NAND Gate)

Objective:

Design, lay out, and verify a 2-input CMOS NAND gate (NAND2X1) using gpdk90. Confirm functionality via ADE L simulations against the theoretical truth table.

Description:

A 2-input CMOS NAND gate outputs logic '1' for all input combinations **except** when both inputs are '1'.

- **Pull-Up Network (PUN):** 2 PMOS in **parallel** (each to VDD).
- **Pull-Down Network (PDN):** 2 NMOS in **series** (to GND).
- When A = B = 1 → PDN conducts (low), PUN off → **Y = 0**.
- Otherwise at least one PMOS is on → **Y = 1**.

Pins:

- Inputs: A, B
- Output: Y
- Supplies: VDD, GND

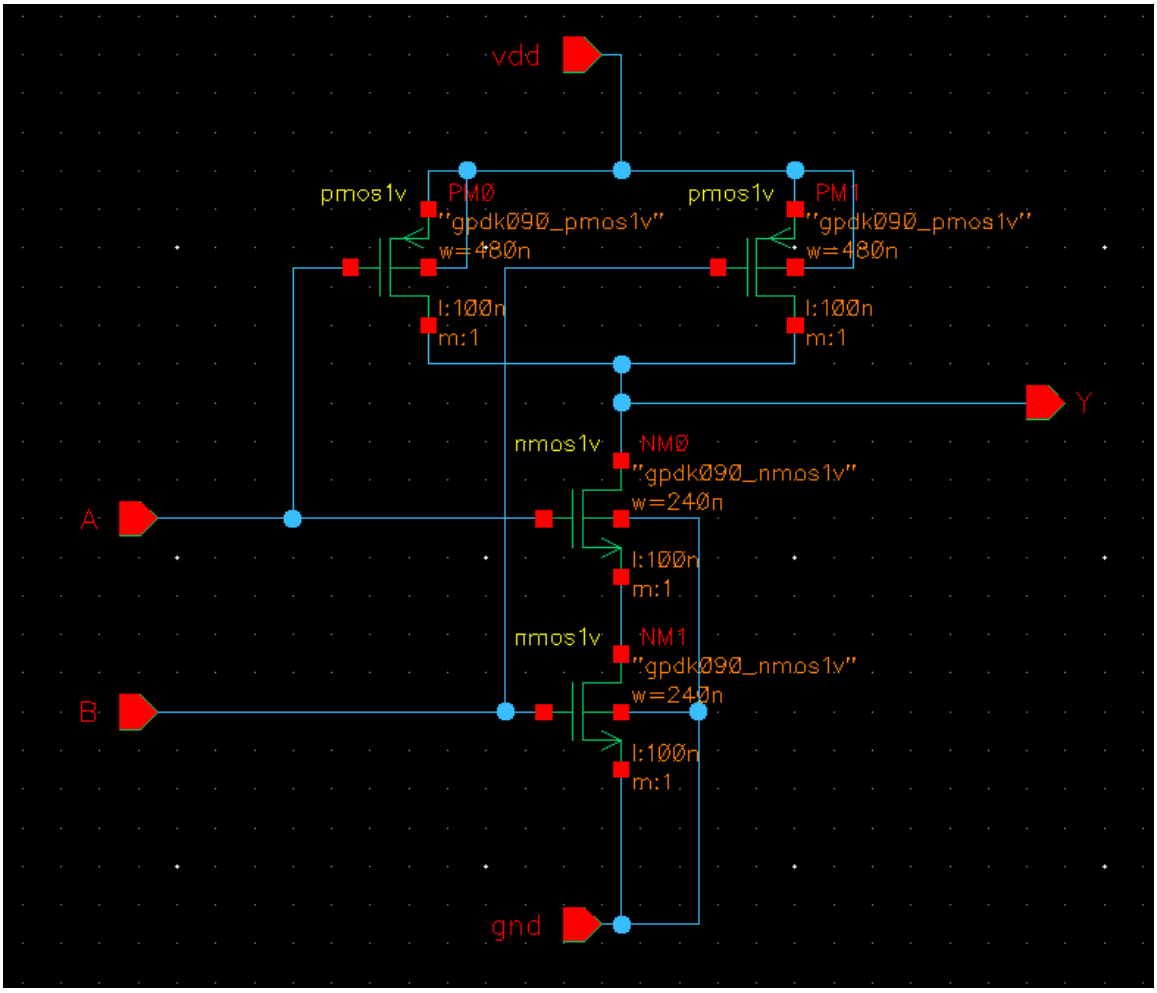
Design Steps:

1. **Schematic (Transistor-Level)**
    - Instantiate 2 PMOS (sources to VDD, drains tied together at node Y) in **parallel**.
    - Instantiate 2 NMOS (drain of top NMOS at Y, source of bottom NMOS to VSS) in **series**.
    - Gate connections: one PMOS + one NMOS get A; the other pair get B.
    - Add pins: A, B, Y, VDD, GND.
2. **Pre-Layout Simulation (ADE L)**
    - Create a testbench: drive A, B with piecewise digital pulses; load Y with a capacitive load (e.g., 5–10 fF) to mimic fan-out.
    - Run DC and transient to confirm basic function and estimate rise/fall.
3. **Layout**
    - Place **common-centroid friendly** arrangement if you care about mismatch (optional).
    - Abut the **series NMOS** to share diffusion; **parallel PMOS** share the drain diffusion at Y.

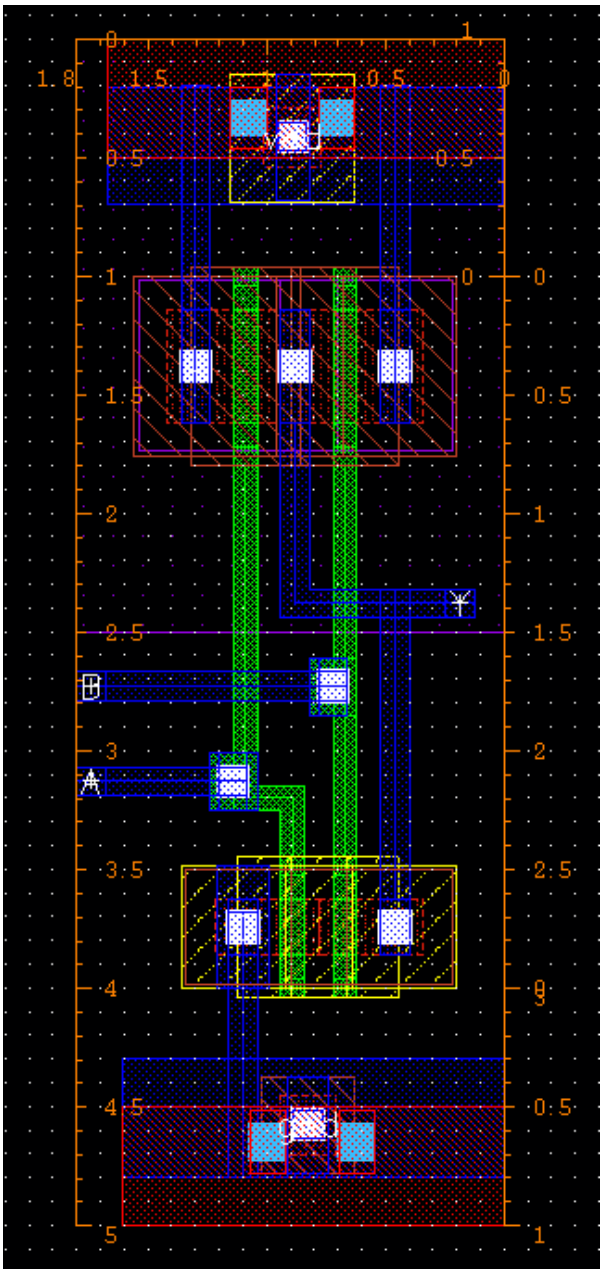
- Add **n-well** for PMOS, p-substrate for NMOS; insert **well/substrate taps** (p+ to VSS, n+ to VDD) at recommended density.
- Route A and B polysilicon to both NMOS/PMOS gates; route Y on M1/M2 (as per your stack).
- Respect **DRC** rules (min spacing, width, enclosure, contact density).

4. **LVS & Extraction**
   - Run **LVS** to match schematic ↔ layout.
   - Run **PEX** (RC extraction) for post-layout timing simulations.
5. **Symbol**
   - Create a clean symbol: pins left (A, B), pin top (VDD), bottom (VSS), right (Y).
   - Add property fields (cell name, view).

Schematic Diagram:



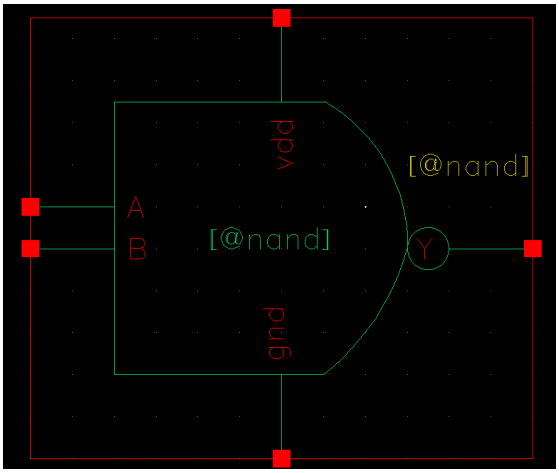
Layout:



Checklist:

- DRC: clean
- Well guard taps density
- Y node diffusion sharing
- A/B poly alignment
- Pin text on correct layers

Symbol:

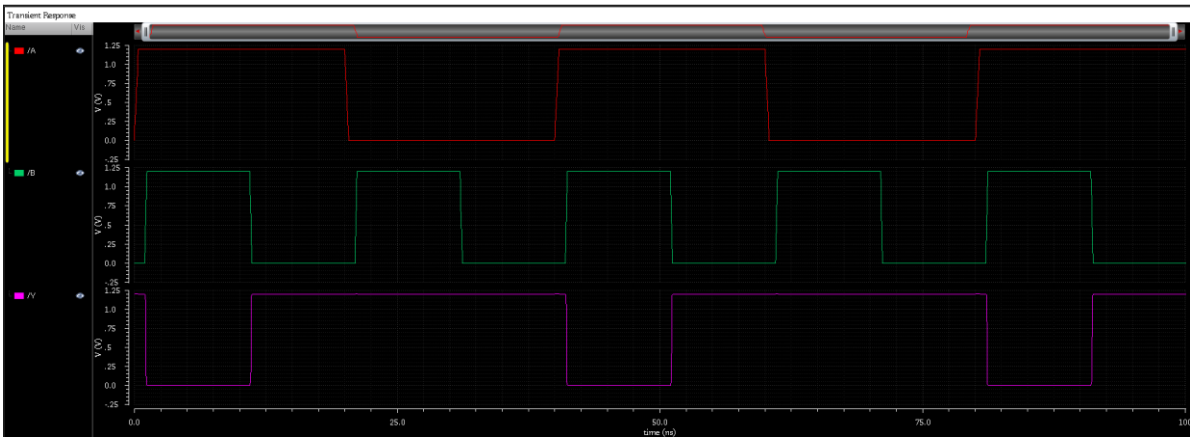


Simulation & Verification:

Transient analysis was performed in Cadence Virtuoso ADE L. A pulse input varying from 0 V to 1.2 V was applied at the input. The output waveform was observed to switch inversely with respect to the input signal. The simulated behavior matched the truth table of the nand gate,

Truth Table:

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0



AND2X1 (2-Input AND Gate)

Objective:

The objective is to design, implement, and verify a 2-input AND gate using the standard cells (NAND and Inverter) previously created.

Description:

An AND gate is a fundamental digital logic gate that produces a high output (**logic 1**) only when all of its inputs are high. If any input is low (**logic 0**), the output will be low. The Boolean expression for a 2-input AND gate is $Y=A \cdot B$. A common way to implement an AND gate in CMOS is by inverting the output of a NAND gate. This is because a NAND gate's output is $A \cdot B$, so an inverter at its output, which performs the operation $A \cdot B$, yields the desired result of $A \cdot B$.

Pins:

- Inputs: A, B
- Output: Y
- Supplies: VDD, GND

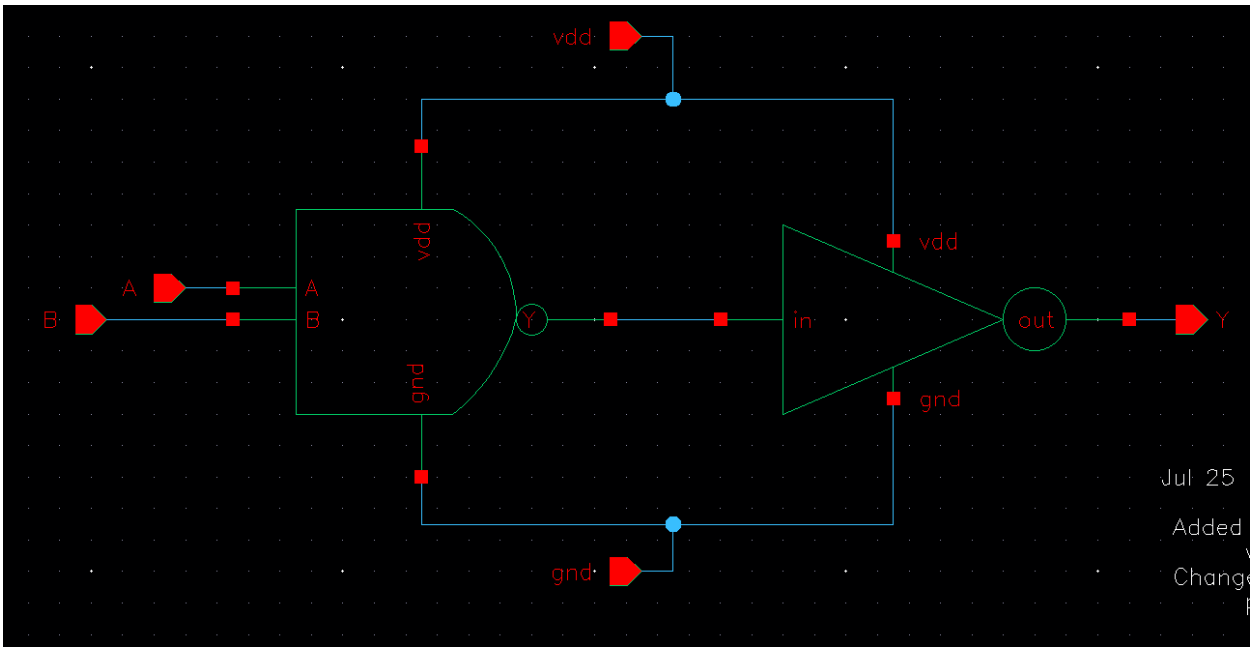
Design Steps:

1. **Hierarchical Schematic Design**: We use a hierarchical design approach. Instead of building the AND gate from scratch with transistors, we instantiate the previously created NAND2X1 and Inverter standard cells.
2. **Schematic Capture**: Connect the two inputs, A and B, to the inputs of the NAND2X1 cell. Then, connect the output of the NAND2X1 cell to the input of the Inverter cell. The output of the Inverter becomes the final output, Y, of the AND gate.
3. **Layout**: Create the layout for the AND gate by placing the layouts of the NAND2X1 and Inverter cells side-by-side. Carefully route the connections between them to connect the output of the NAND gate to the input of the inverter and to connect the power and ground rails.
4. **DRC/LVS**: Perform Design Rule Check (DRC) to ensure the layout adheres to the manufacturing rules. Execute Layout Versus Schematic (LVS) to verify that the physical layout accurately matches the logical schematic.

5. **Simulation Setup**: Configure the ADE L environment for a transient analysis. Apply pulsed voltage sources to inputs A and B to test all four possible input combinations (00, 01, 10, 11).
6. **Simulation & Verification**: Run the simulation and observe the output waveform. Compare the results against the AND gate's truth table to confirm correct functionality.

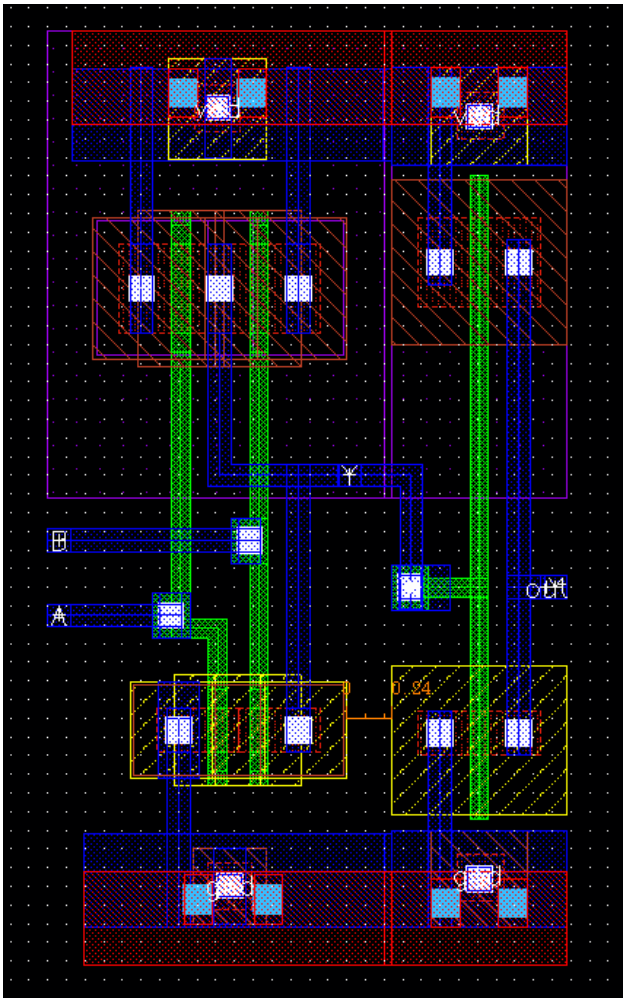
Schematic Diagram:

The schematic shows the two cells, NAND2X1 and Inverter, cascaded to create the AND gate functionality.

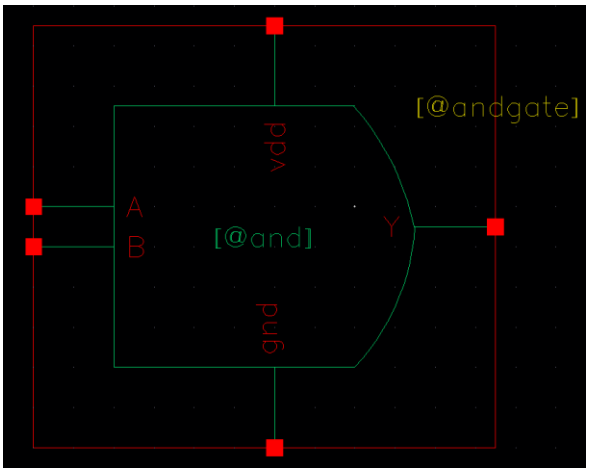


Layout:

The layout displays the physical placement of the two cell layouts with the metal interconnects routed to create the required connections.



Symbol:

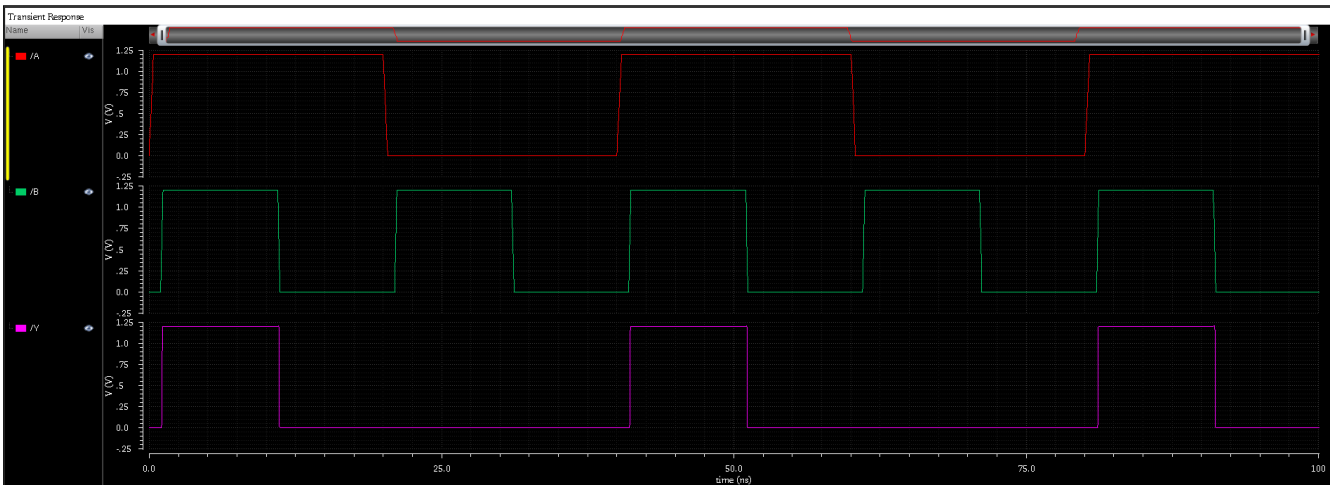


Simulation & Verification:

The simulation plot below shows the input waveforms for A and B and the resulting output waveform for Y. The output is only high when both inputs A and B are simultaneously high, which is consistent with the truth table.

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

The simulation successfully validates the AND gate's operation, confirming that the output matches the truth table for all possible input combinations.



NOR2X1 (2-Input NOR Gate)

Objective:

The objective is to design, implement, and verify a 2-input NOR gate from a CMOS transistor-level perspective.

Description:

A NOR gate is a digital logic gate that produces a low output (**logic 0**) if any of its inputs are high (**logic 1**). The output is high only when all inputs are low. The Boolean expression for a 2-input NOR gate is Y=A+B. In CMOS logic, a NOR gate is implemented with PMOS transistors in series and NMOS transistors in parallel.

Pins:

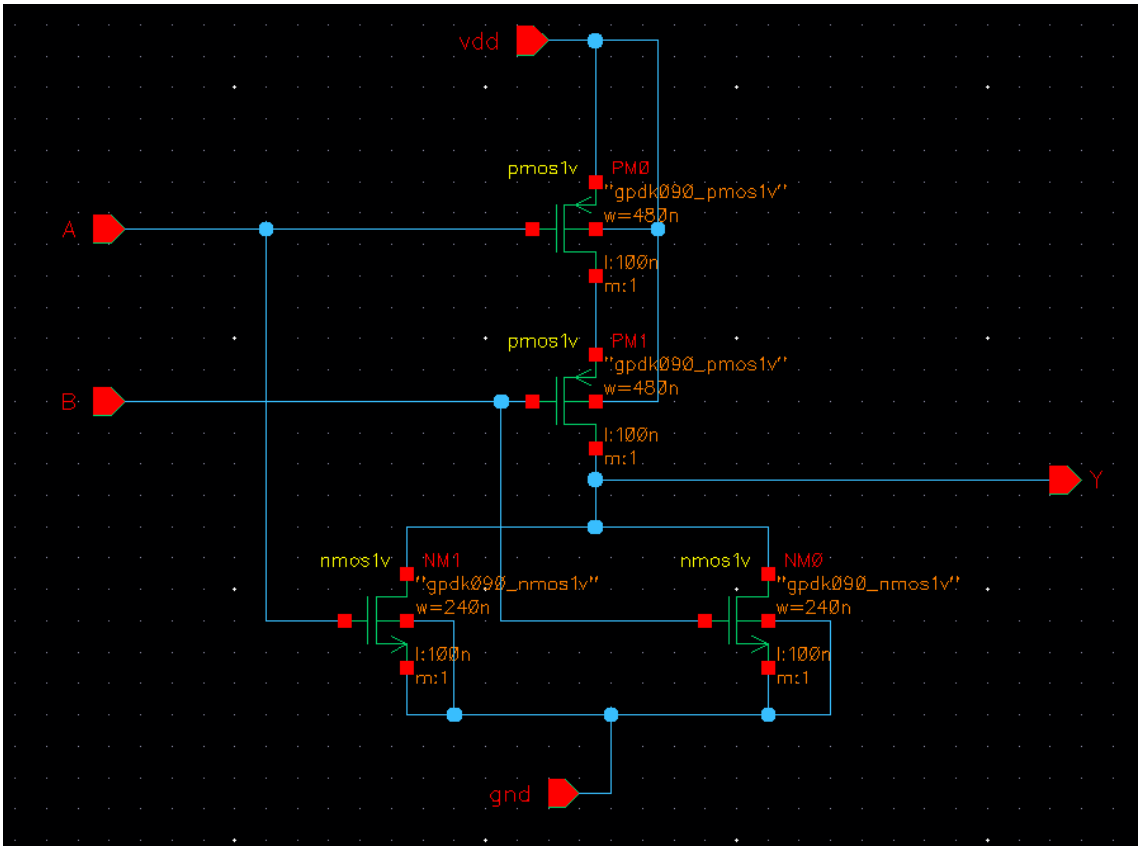
- Inputs: A, B
- Output: Y
- Supplies: VDD, GND

Design Steps:

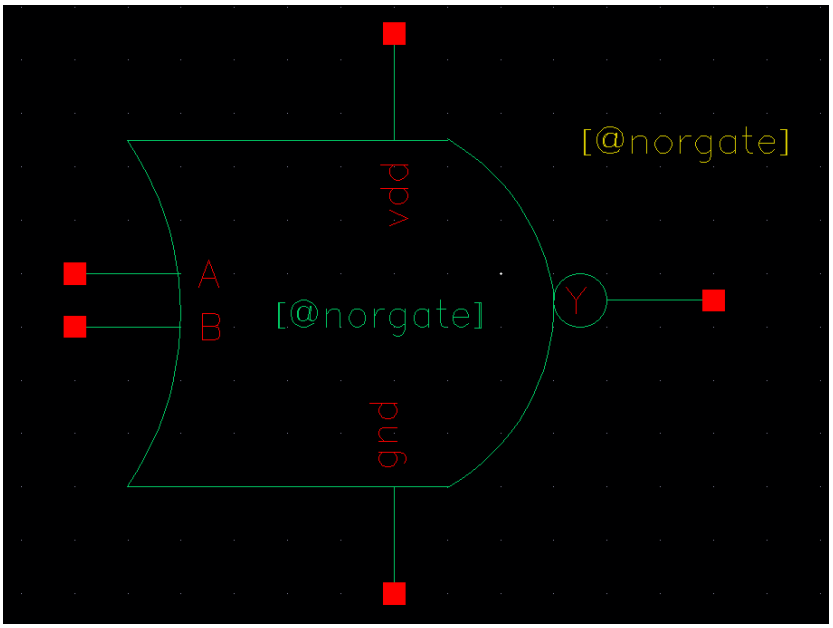
1. **Schematic Design**: Use two PMOS transistors connected in series between VDD and the output node (Y). Then, use two NMOS transistors connected in parallel between the output node (Y) and GND. Connect the inputs A and B to the gates of one PMOS and one NMOS transistor each.
2. **Layout Generation**: Create the physical layout for the gate. Place the transistors and use metal layers to connect the series PMOS transistors, the parallel NMOS transistors, and the input/output nodes.
3. **DRC/LVS**: Run Design Rule Check (DRC) to confirm that the layout adheres to the manufacturing specifications. Perform Layout Versus Schematic (LVS) to verify that the physical connections in the layout match the logical connections in the schematic.
4. **Simulation Setup**: Set up a transient analysis in ADE L. Apply pulsed inputs to A and B to cover all four possible input combinations (00, 01, 10, 11).
5. **Simulation & Verification**: Run the simulation and plot the input and output waveforms. Compare the simulated output to the NOR gate's truth table to confirm correct functionality.

Schematic Diagram:

The schematic shows the series PMOS pull-up network and the parallel NMOS pull-down network, which form the core of the CMOS NOR gate.

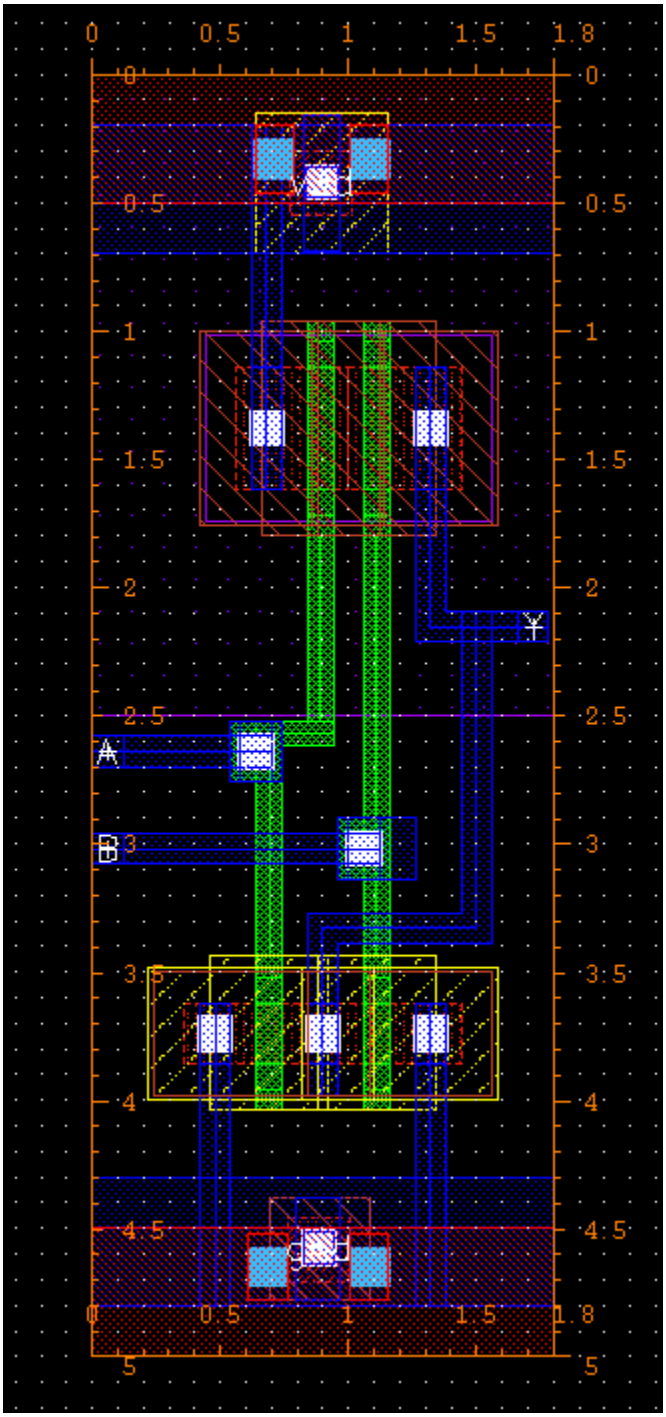


Symbol:



Layout:

The layout shows the physical arrangement of the transistors and the routed connections, including the VDD, GND, input, and output pins.

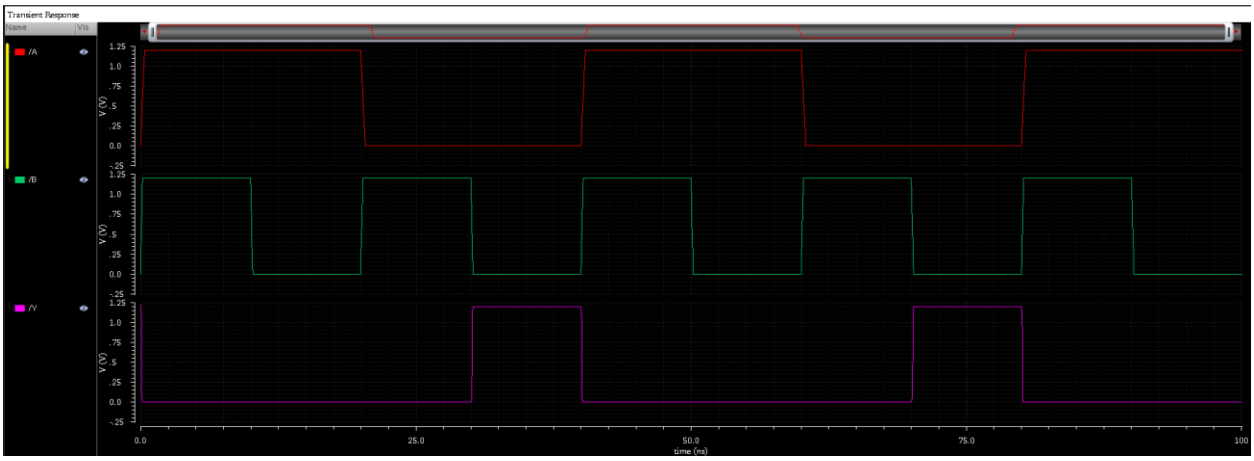


Simulation & Verification:

The simulation plot below shows the input waveforms for A and B and the resulting output waveform for Y. The output is low whenever at least one of the inputs is high, which is consistent with the NOR gate's truth table.

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

The simulation successfully validates the NOR gate's operation, confirming that the output matches the truth table for all possible input combinations.



OR2X1 (2-Input OR Gate)

Objective:

The objective is to design, implement, and verify a 2-input OR gate using the standard cells (NOR and Inverter) that were previously created.

Description:

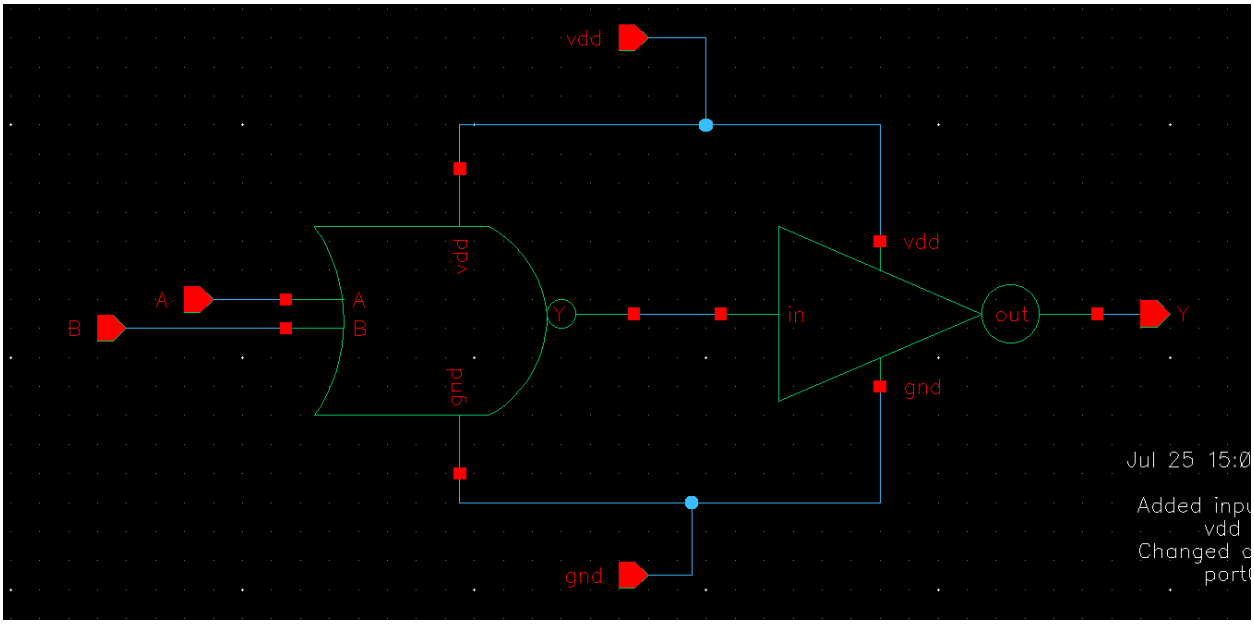
An OR gate is a fundamental digital logic gate that produces a high output (**logic 1**) if any of its inputs are high (**logic 1**). The output is low (**logic 0**) only when all inputs are low. The Boolean expression for a 2-input OR gate is Y=A+B. An efficient way to realize an OR gate in CMOS logic is to follow a NOR gate with an inverter. This method utilizes the property that the output of a NOR gate is A+B, and by inverting this, we get A+B, which simplifies to the desired A+B.

Design Steps:

1. **Hierarchical Schematic Design**: The OR gate is designed hierarchically by reusing the already verified NOR2X1 and Inverter standard cells.
2. **Schematic Capture**: Connect the two inputs, A and B, to the inputs of the NOR2X1 cell. The output of the NOR2X1 cell is then connected to the input of the Inverter cell. The output of the Inverter serves as the final output, Y, of the OR gate.
3. **Layout**: Create the layout for the OR gate by arranging the layouts of the NOR2X1 and Inverter cells. The cells are placed optimally to minimize the routing length and area. Connections are made between the cells, and the power and ground rails are routed.
4. **DRC/LVS**: Perform Design Rule Check (DRC) to ensure that the physical layout complies with the design rules of the gpdk90 PDK. Run Layout Versus Schematic (LVS) to verify that the layout and the schematic are logically identical.
5. **Simulation Setup**: Use the Cadence ADE L environment to set up a transient analysis. Apply pulsed input signals to A and B to test all four possible input combinations (00, 01, 10, 11).
6. **Simulation & Verification**: Run the simulation and examine the resulting waveforms. The output waveform is compared to the OR gate's truth table to confirm that the circuit functions as intended.

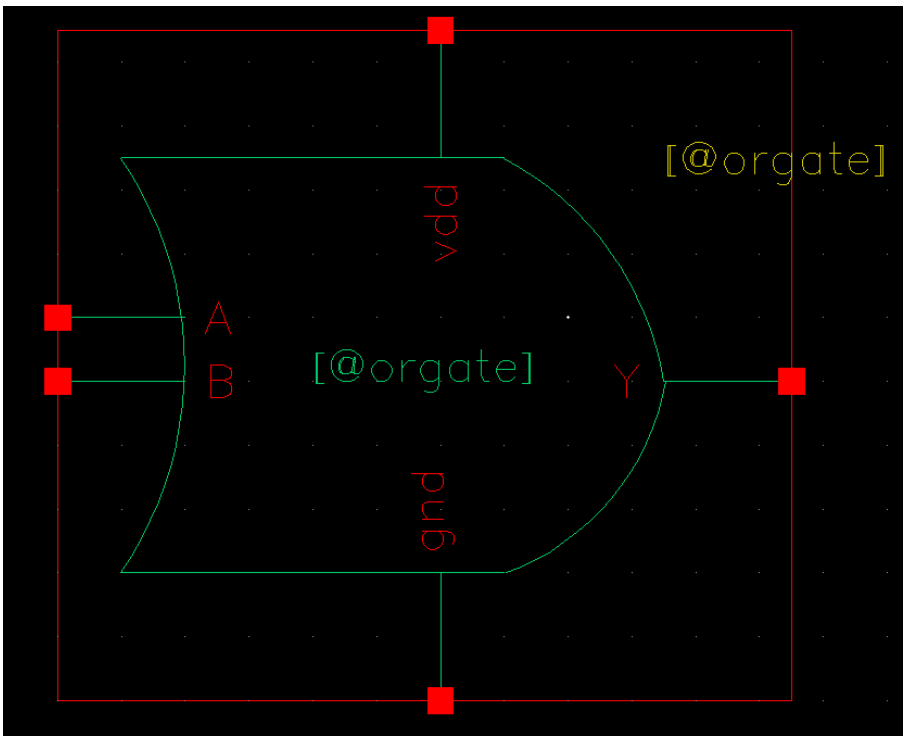
Schematic Diagram

This schematic illustrates the hierarchical design, showing the NOR2X1 cell followed by the Inverter cell.



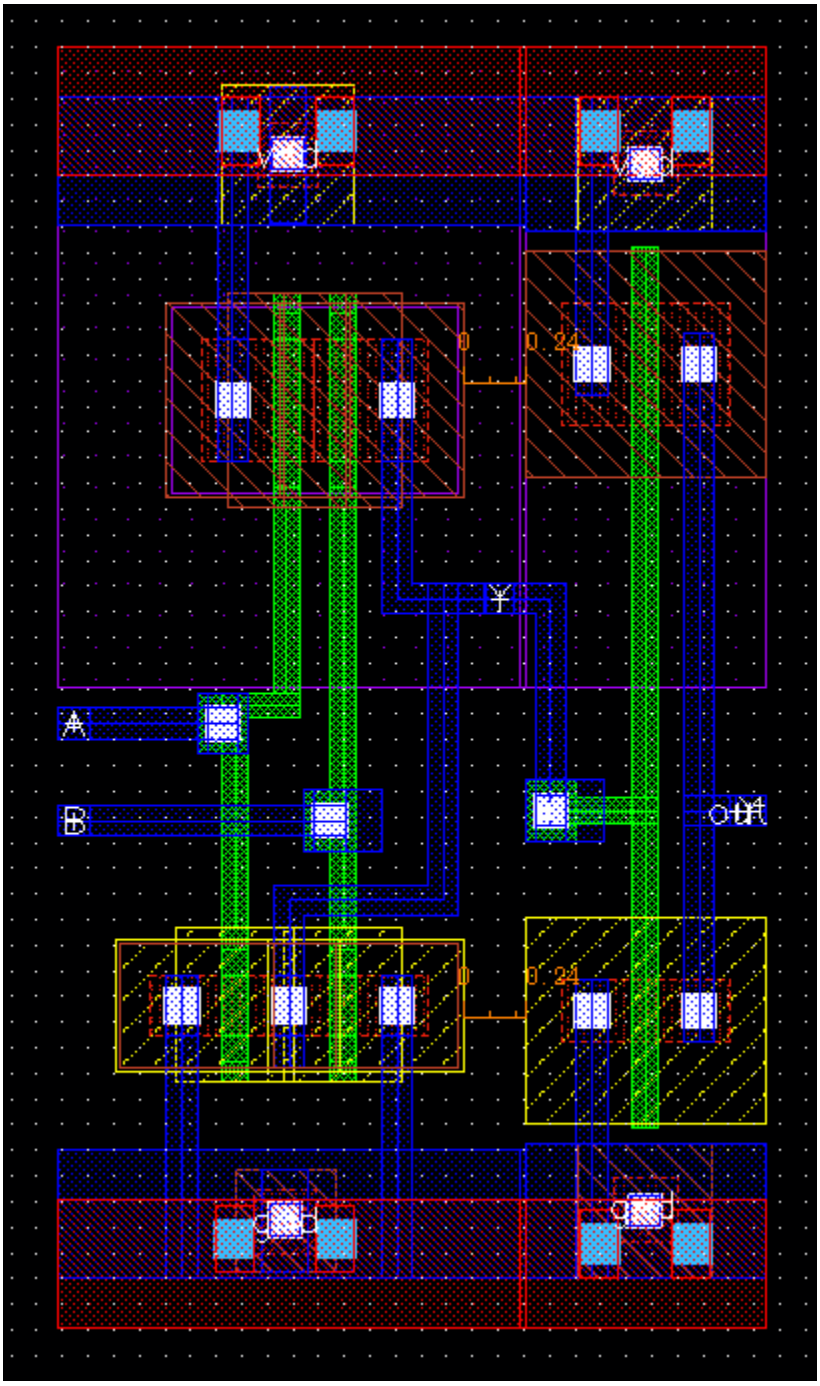
Symbol

The symbol represents the 2-input OR gate with a standard logical representation, including inputs A and B and a single output Y, making it easy to integrate into larger circuits.



Layout

This layout shows the physical arrangement of the NOR and Inverter cell layouts, along with the routing of power, ground, and signal lines.

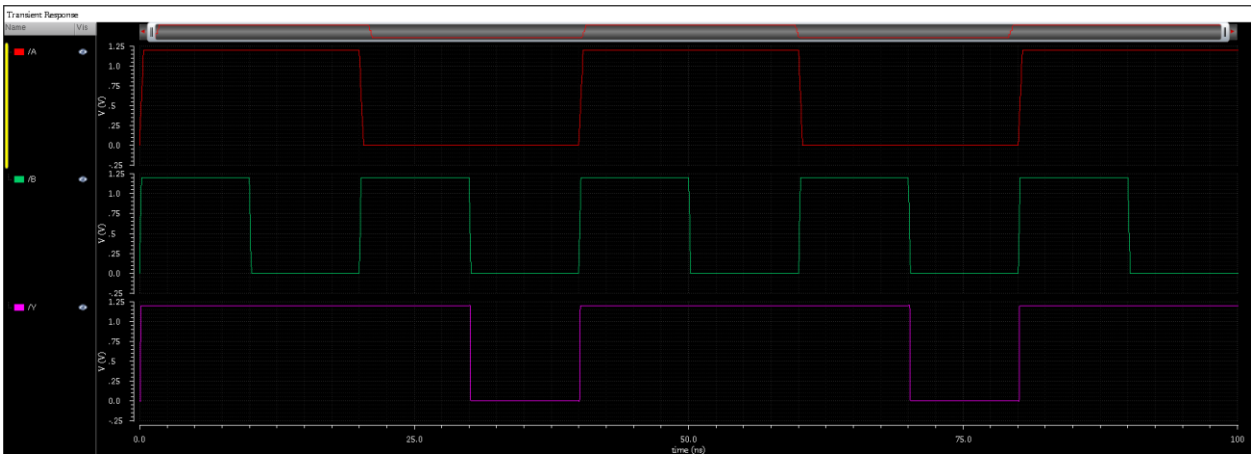


Simulation & Verification

The transient simulation waveforms below show the output Y is high whenever either or both of the inputs (A or B) are high, which is the defining characteristic of an OR gate. The output is only low when both inputs are low.

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

The simulation confirms that the designed circuit successfully operates as an OR gate, validating its functionality against the truth table for all input combinations.



M1_NWELL

Objective:

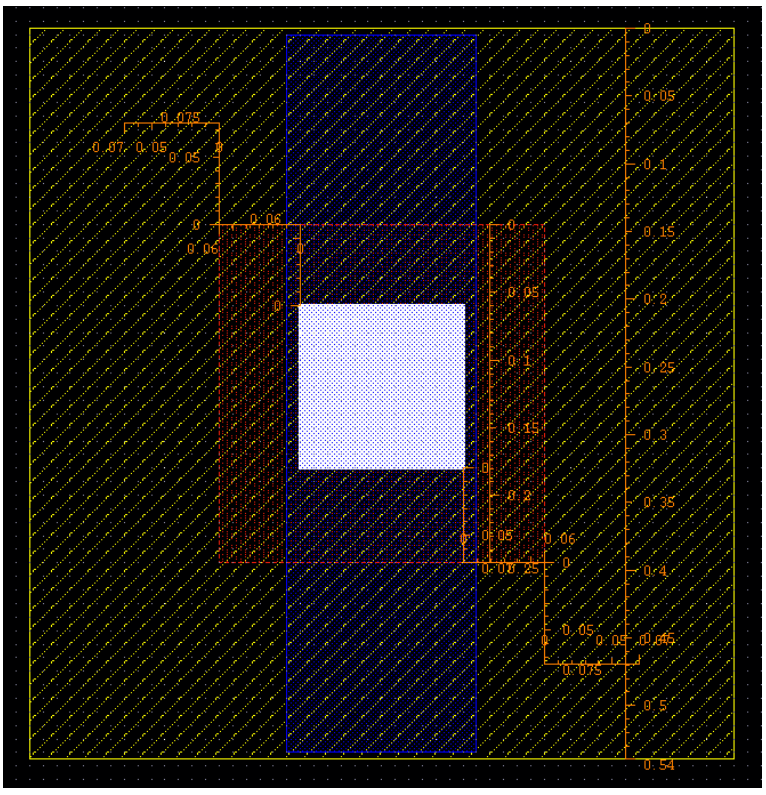
The objective is to understand the purpose and physical representation of the M1_NWELL layer in the layout of a CMOS integrated circuit.

Description:

The M1_NWELL layer is a foundational layer in CMOS technology, representing the **N-Well**, which is a region of the silicon substrate that has been doped with n-type impurities. This layer is specifically created to house **PMOS transistors**. Because PMOS transistors require a p-type source and drain built within an n-type body (the N-Well), this layer provides the necessary environment. The N-Well must be connected to the highest potential in the circuit (VDD) to prevent the formation of parasitic transistors and to ensure proper operation.

Layout:

In the layout, the M1_NWELL layer is a large, colored polygon that encloses all the PMOS transistors within a specific circuit block. Its shape and size are determined by the placement of the PMOS devices and the design rules regarding minimum well area and spacing. All PMOS devices within a single well share the same body connection, which is tied to the VDD power rail via a **P-type tap** (P_TAP).



M1_PSUB

Objective:

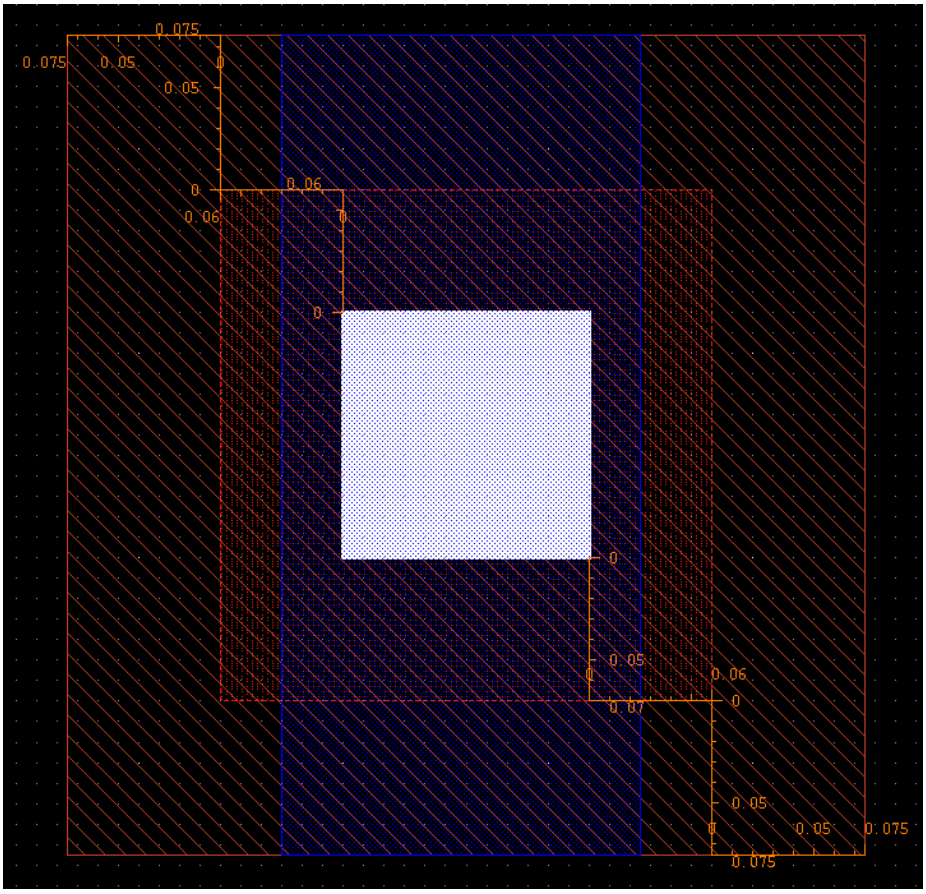
The objective is to understand the purpose and physical representation of the M1_PSUB layer, which is essential for the layout of NMOS transistors.

Description:

The M1_PSUB layer represents the **P-Substrate** connection. It is not a separate well like the N-Well but rather a tap into the bulk p-type silicon wafer itself. This layer is crucial for providing a stable body potential for all **NMOS transistors**, which are built directly on the P-Substrate. The P-Substrate must always be connected to the lowest potential in the circuit (GND) to prevent device latch-up and ensure that the NMOS transistors function correctly.

Layout:

In the layout, the M1_PSUB layer is represented by a specific contact that connects a metal layer to the P-Substrate. These connections, called **N-type taps** (N_TAP), are placed at regular intervals around the NMOS transistors. The M1_PSUB layer ensures that the body of every NMOS transistor is held at ground potential, providing a stable reference for the circuit.



Section 2: Digital Blocks and Systems

Each Digital Blocks and System is presented with the following format:

- Objective
- Description
- Design Steps
- Schematic Diagram
- Layout
- Symbol
- Simulation & Verification

Half Adder (with NAND Gates)

Objective:

The objective is to design, implement, and verify a 1-bit half adder circuit using only 2-input NAND gates. This demonstrates the universality of the NAND gate and its ability to construct any other logic function.

Description:

A half adder is a combinational logic circuit that performs the addition of two single-bit binary numbers, A and B. It produces two outputs: a **Sum (Y)** and a **Carry (C)**. The Sum output represents the least significant bit of the sum, and the Carry output represents a carry-out to the next bit position. The Boolean expressions for the half adder are:

- **Sum (Y)** = $A \oplus B$
- **Carry (C)** = $A \cdot B$

We can implement these functions using only NAND gates. The **Carry (C)** output, which is $A \cdot B$, can be derived from the NAND function $A \cdot B$ by using an inverter (which is itself made of a single NAND gate). The **Sum (S)** output, which is $A \oplus B$, requires a more complex arrangement of NAND gates. The expression for the XOR function in terms of NAND is $S=(A \cdot A \cdot B)+(B \cdot A \cdot B)$, which can be implemented with four NAND gates.

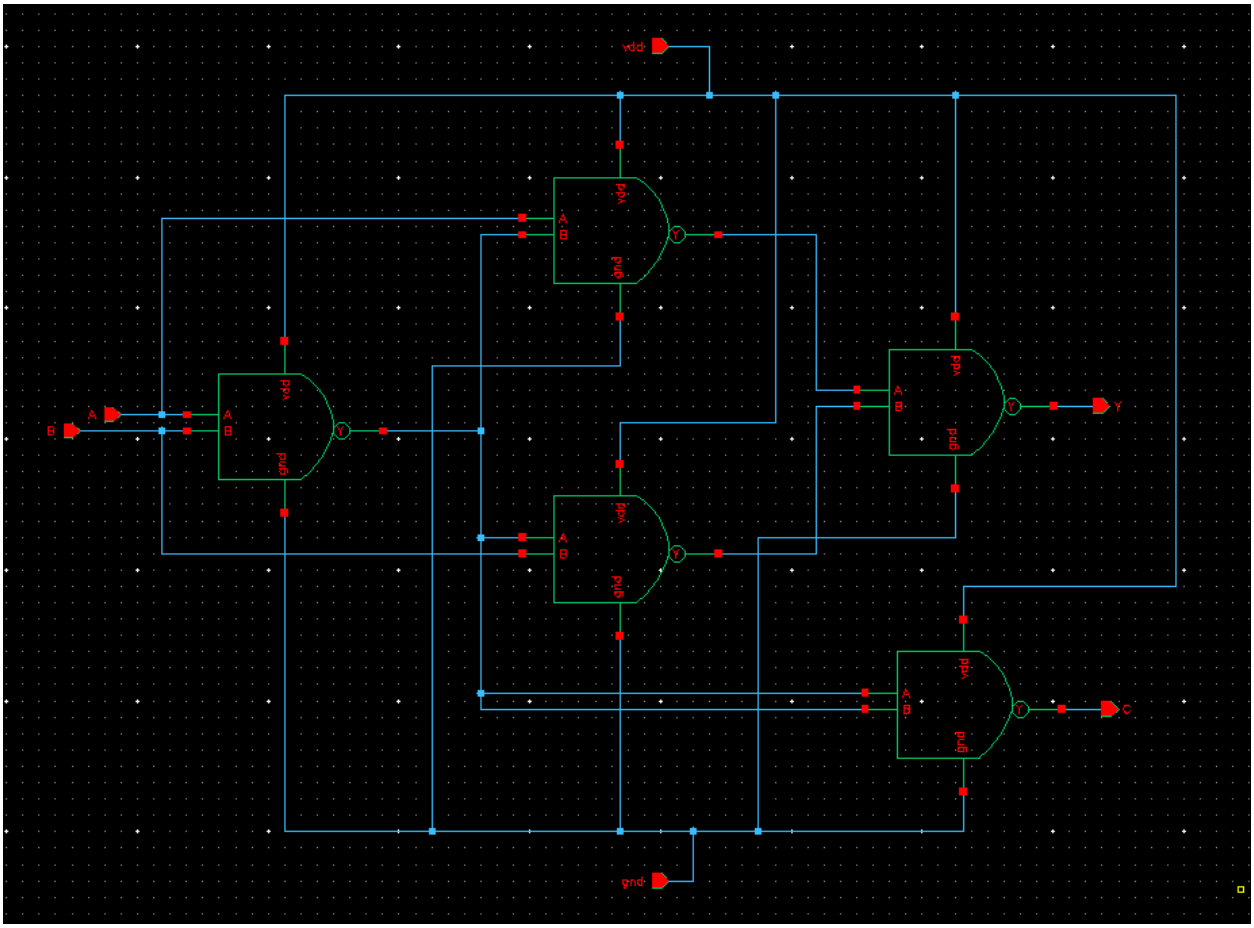
Design Steps:

1. **Schematic Design**: Instantiate the NAND2X1 standard cell as the building block.
    a. To get the **Carry (C)** output, connect the inputs A and B to a single NAND2X1 gate, and then connect the output of that gate to another NAND2X1 gate with both of its inputs tied together to create an inverter.
    b. To get the **Sum (Y)** output, a more complex configuration is required. A classic implementation uses four NAND gates. Connect inputs A and B to the first NAND gate. The output of this gate feeds into two separate NAND gates, one with input A and the other with input B. The outputs of these two gates are then fed into a final NAND gate to produce the Sum output.
2. **Layout Generation**: Place the NAND gate layouts in a compact and organized manner. Carefully route the connections to minimize signal path length and avoid design rule violations.
3. **DRC/LVS**: Run Design Rule Check (DRC) to ensure the layout adheres to the manufacturing rules. Perform Layout Versus Schematic (LVS) to confirm that the physical layout matches the logic of the schematic.

4. **Simulation Setup**: Configure the Cadence ADE L environment for a transient analysis. Apply pulsed voltage sources to inputs A and B to test all four possible input combinations (00, 01, 10, 11).
5. **Simulation & Verification**: Run the simulation and examine the waveforms for the Sum and Carry outputs. Compare the results against the half adder's truth table to confirm correct functionality.

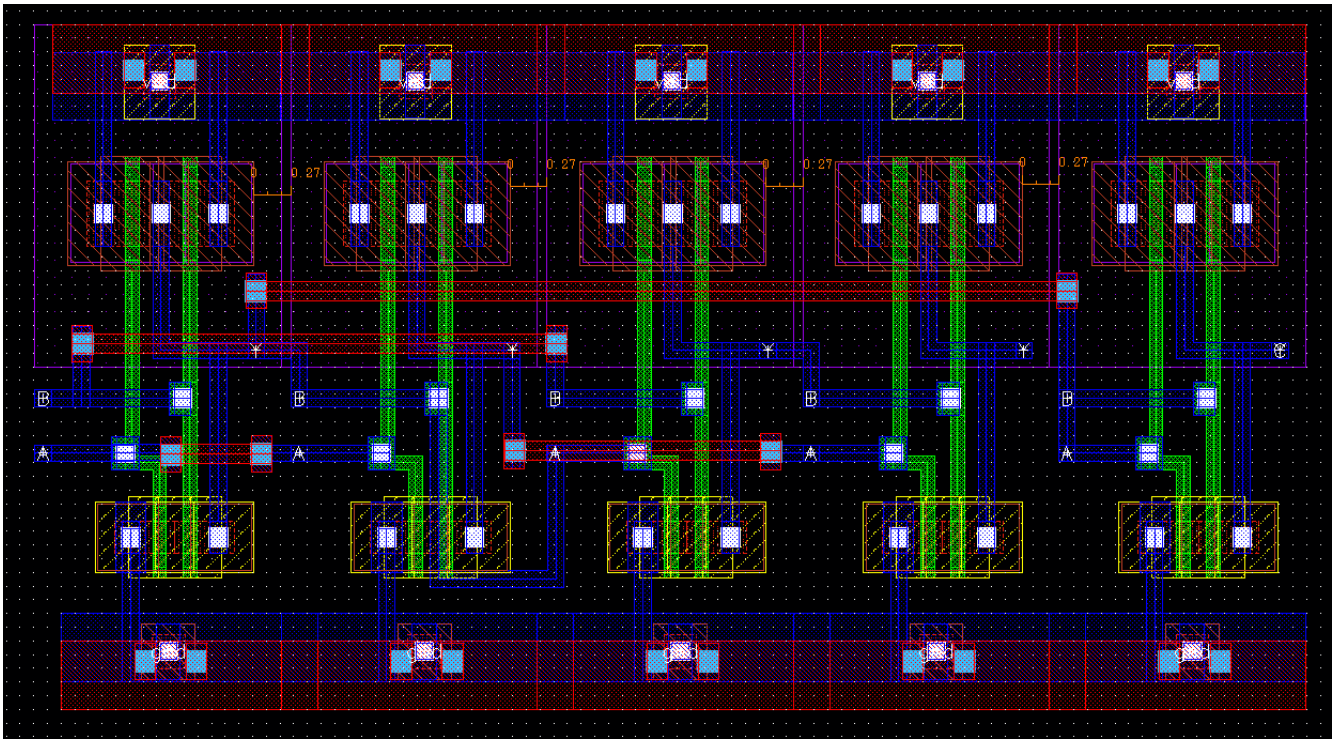
Schematic Diagram:

The schematic shows the two branches of the circuit: one for the Sum (Y) using four NAND gates and one for the Carry (C) using two NAND gates.



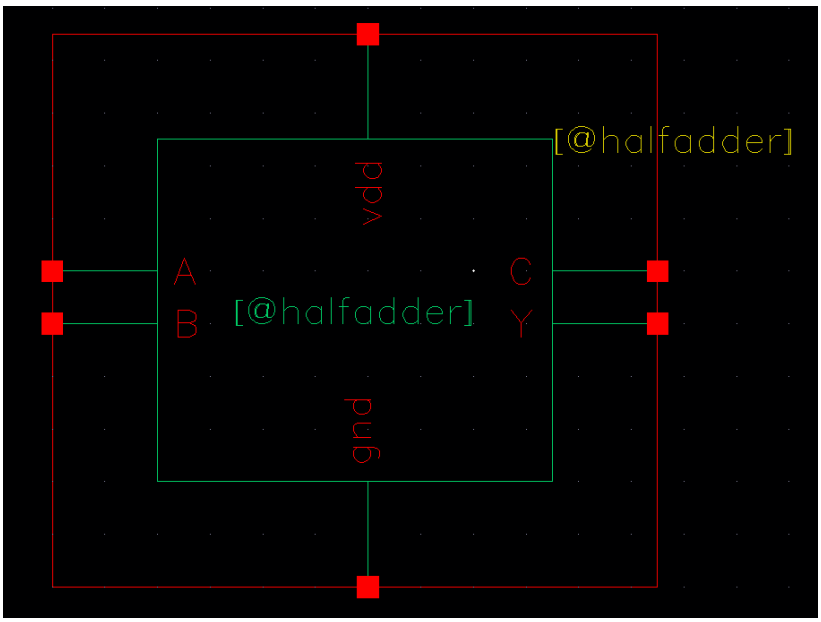
Layout:

This layout shows the physical placement of the six NAND gates and the routing of the interconnections.



Symbol:

The symbol provides a high-level representation of the half adder, with two inputs, A and B, and two outputs, Sum and Carry.

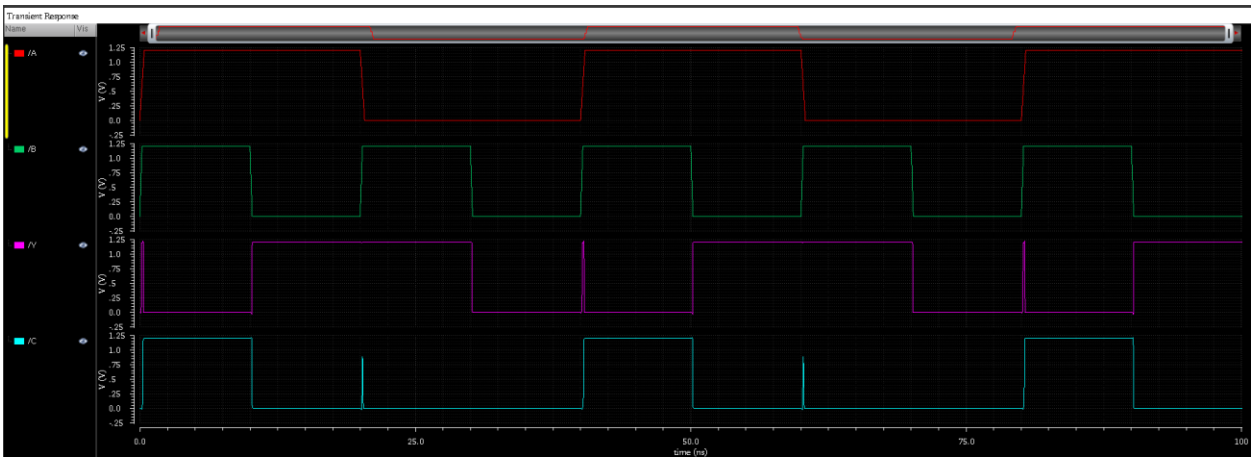


Simulation & Verification:

The simulation plot below shows the input waveforms for A and B, along with the resulting Sum and Carry outputs. The results perfectly match the half adder's truth table.

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

The simulation successfully validates the design, confirming that the circuit performs the correct binary addition.



Full Adder

Objective:

The objective is to design, implement, and verify a 1-bit full adder circuit using only 2-input NAND gates. This task highlights the ability to create complex combinational logic by leveraging the universal property of the NAND gate.

Description:

A full adder is a combinational logic circuit that adds three single-bit binary numbers: two data inputs, **A** and **B**, and a carry-in, Cin. It produces two outputs: a **Sum (Y)** and a **Carry-out (Cout)**. The circuit is crucial for creating multi-bit adders.

The Boolean expressions for a full adder are:

- **Sum (Y)** = $A \oplus B \oplus Cin$
- **Carry-out (Cout)** = $(A \cdot B)+(Cin \cdot (A \oplus B))$

These functions can be realized entirely with NAND gates by leveraging the fact that an XOR gate can be made from four NAND gates and an AND gate can be made from two NAND gates. By connecting these NAND-based sub-circuits, a full adder can be constructed.

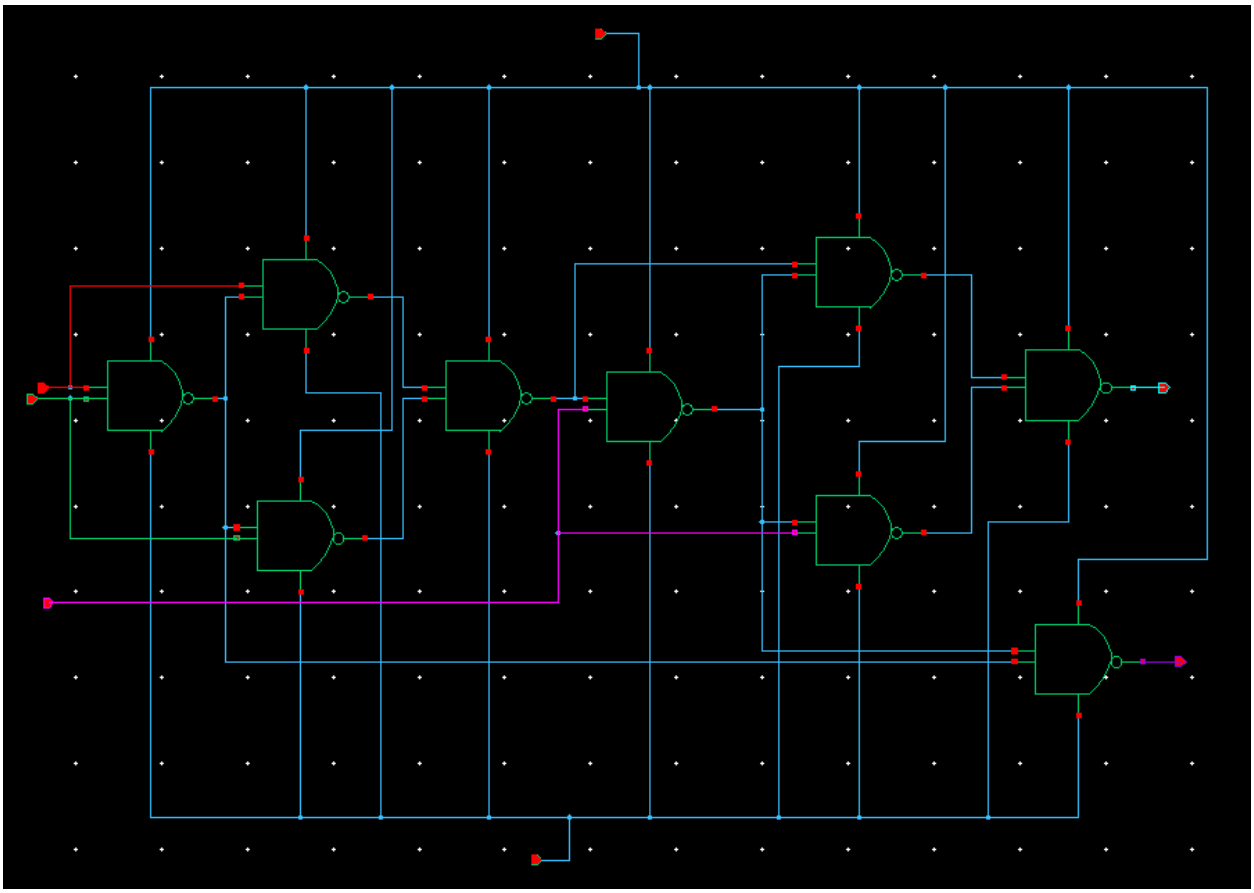
Design Steps:

1. **Hierarchical Schematic Design**: The full adder is designed hierarchically by instantiating two half adders (constructed with NAND gates) and an OR gate (constructed with NAND gates).
2. **Schematic Capture**:
    - Connect the inputs **A** and **B** to the first NAND-based half adder. The outputs of this half adder are Sum1 and Carry1.
    - Connect Sum1 and the carry-in Cin to the inputs of the second NAND-based half adder. The sum output of this second half adder is the final **Sum (Y)** output of the full adder.
    - The Carry1 output from the first half adder and the carry output from the second half adder (Cout2) are combined using an OR gate. This OR gate is implemented with three NAND gates. The output of this OR gate is the final **Carry-out (Cout)**.

3. **Layout Generation**: Place the layouts of the NAND gates (and the half adder and OR gate sub-blocks) in a compact layout. Route all the interconnections, being careful to minimize wire length and avoid violations.
4. **DRC/LVS**: Run Design Rule Check (DRC) to ensure the physical design conforms to the PDK specifications. Perform Layout Versus Schematic (LVS) to verify that the layout and the schematic are logically identical.
5. **Simulation Setup**: In the ADE L environment, set up a transient analysis. Apply pulsed voltage sources to all three inputs **A**, **B**, and Cin to cover all eight possible input combinations.
6. **Simulation & Verification**: Run the simulation and plot the input and output waveforms. Compare the simulated results for **Sum** and **Carry-out** against the full adder's truth table to confirm that the circuit is working as expected.

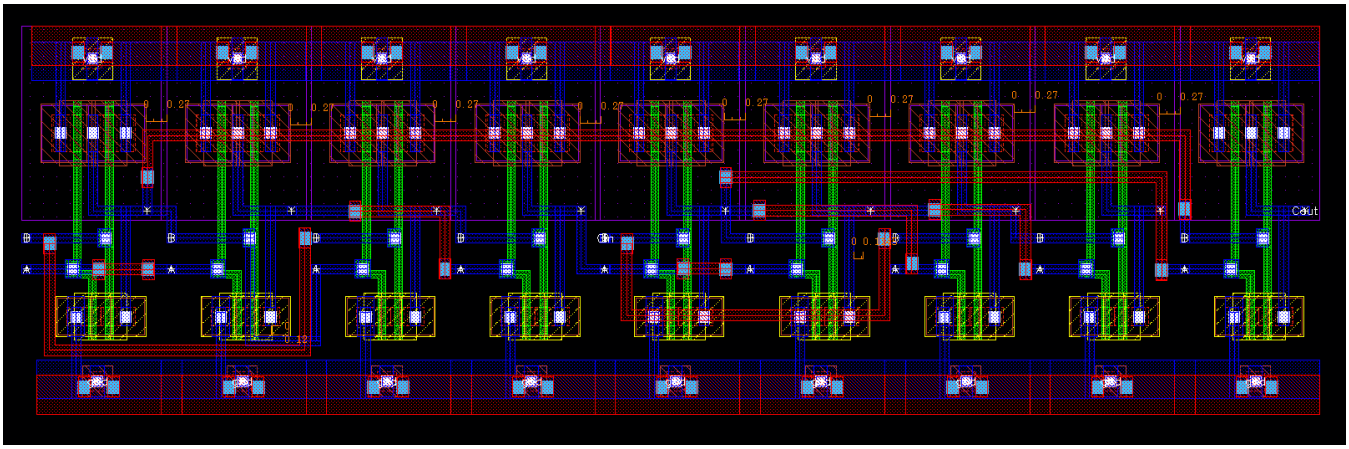
Schematic Diagram:

This schematic shows how two NAND-based half adders and a NAND-based OR gate are interconnected to form the full adder.



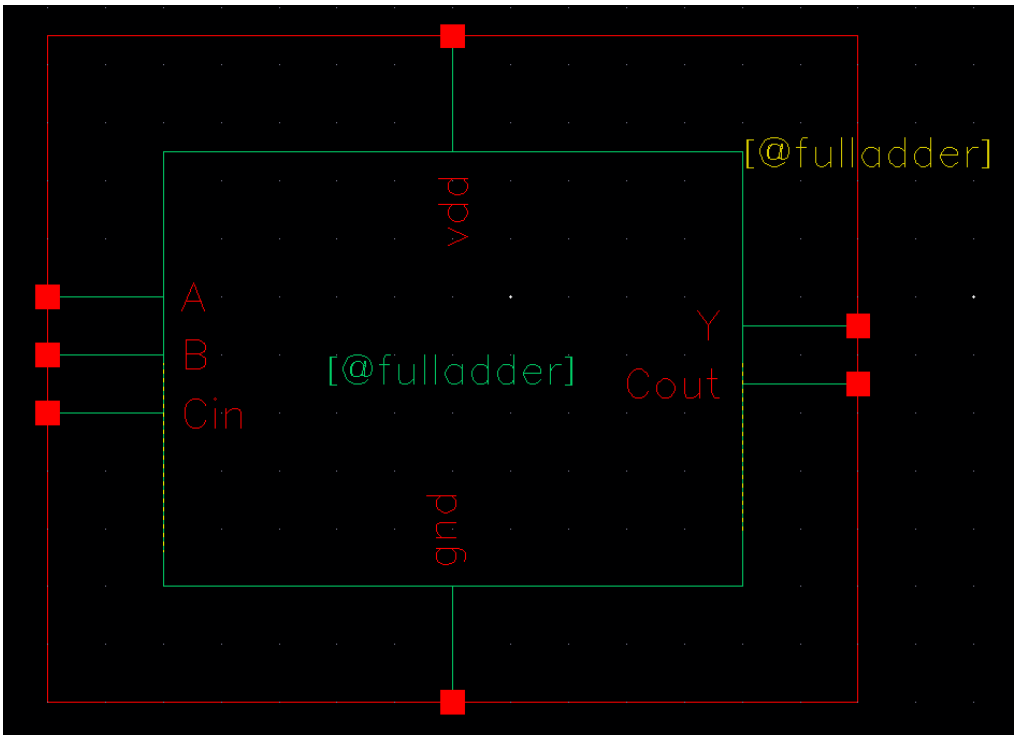
Layout:

This layout displays the physical arrangement of all the individual NAND gates and the complex routing required to connect them.



Symbol:

The symbol for the full adder is a standard representation with three inputs, **A**, **B**, and Cin, and two outputs, **S** and Cout.

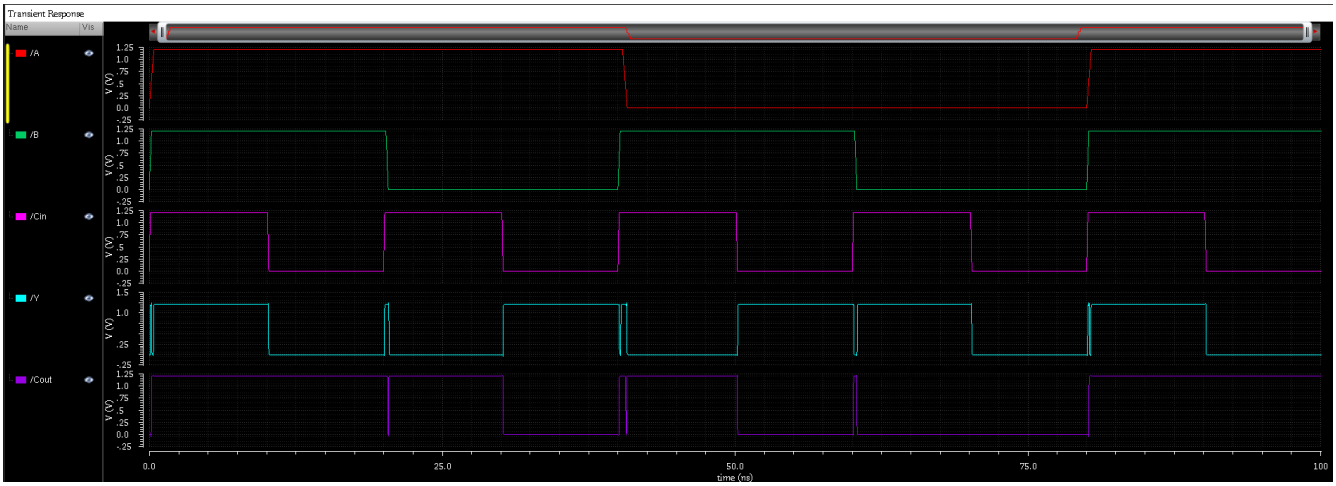


Simulation & Verification:

The simulation plot below shows the waveforms for the inputs and outputs. The results confirm that the circuit correctly performs 1-bit binary addition with a carry-in.

A	B	Cin	Y	Cout
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

The simulation successfully validates the full adder's operation against its truth table for all possible input combinations, confirming the successful implementation of this complex logic circuit using only NAND gates.



1 x 2 De-multiplexer

Objective:

The objective is to design, implement, and verify a 1x2 demultiplexer circuit using only 2-input NAND gates. This design showcases how a universal gate can be used to build a fundamental data distribution circuit.

Description:

A demultiplexer (DEMUX) is a combinational logic circuit that takes a single data input and a set of select lines, and then routes the data to one of several outputs. A 1x2 demultiplexer has one data input (A), one select line (S), and two outputs (Y0 and Y1). The select line determines which of the two outputs the data input will be connected to.

- When S=0, the data input D is routed to output Y0.
- When S=1, the data input D is routed to output Y1.

The Boolean expressions for the outputs are:

- $Y0=A \cdot S$
- $Y1=A \cdot S$

Since an AND gate can be created with two NAND gates and a NOT gate can be created with a single NAND gate, this circuit can be constructed entirely with NAND gates.

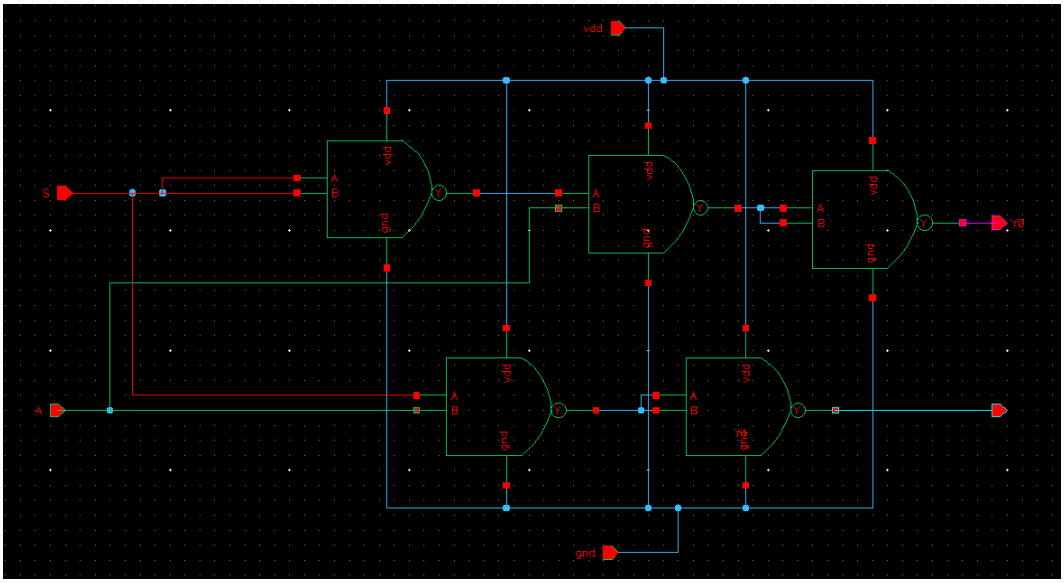
Design Steps:

- **Schematic Design**: Instantiate the NAND2X1 standard cell as the primary building block.
    - To implement $Y0=D \cdot S$, first create an inverter using a single NAND gate with its inputs tied together. This produces S. Then, use two more NAND gates to form an AND gate. The inputs to this AND gate will be D and S.
    - To implement $Y1=D \cdot S$, use two NAND gates to form an AND gate. The inputs to this AND gate will be D and S.
- **Layout Generation**: Arrange the NAND gate layouts in a logical and compact manner. Route the power, ground, and signal lines to ensure all connections are made without violating design rules.
- **DRC/LVS**: Perform Design Rule Check (DRC) to confirm that the physical layout complies with the gpdk90 PDK specifications. Run Layout Versus Schematic (LVS) to verify that the layout and the schematic are functionally identical.
- **Simulation Setup**: In the ADE L environment, set up a transient analysis. Apply pulsed voltage sources to the data input (D) and the select line (S).

- **Simulation & Verification**: Run the simulation and examine the waveforms. Verify that the outputs Y0 and Y1 behave according to the demultiplexer's truth table, based on the value of the select line S.

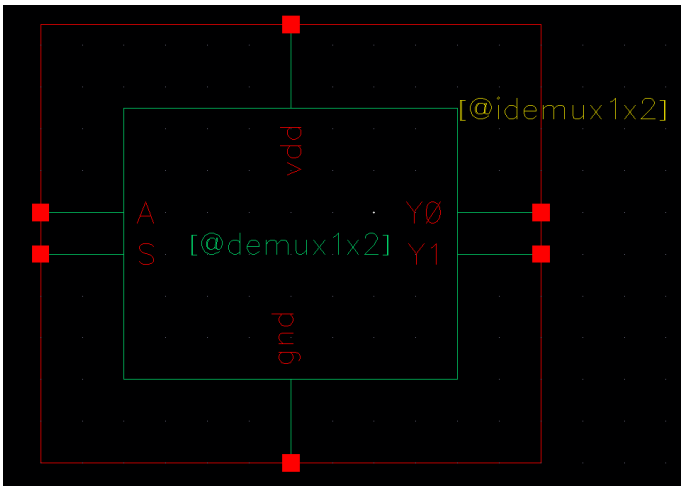
Schematic Diagram:

This schematic shows how the data input and select line are connected to the NAND gates to correctly route the signal to one of the two outputs.



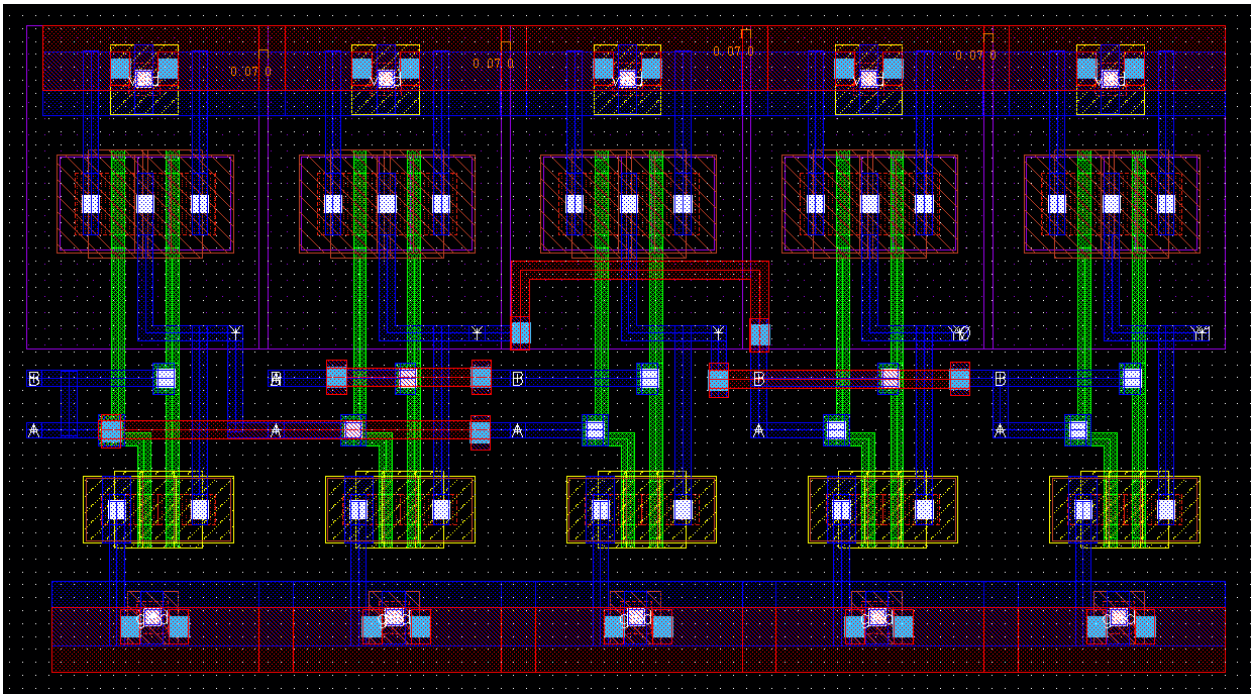
Symbol:

The symbol provides a high-level representation with a single data input (D), a select line (S), and two outputs (Y0 and Y1).



Layout:

This layout displays the physical arrangement of the NAND gates and the routing of the interconnections between them.

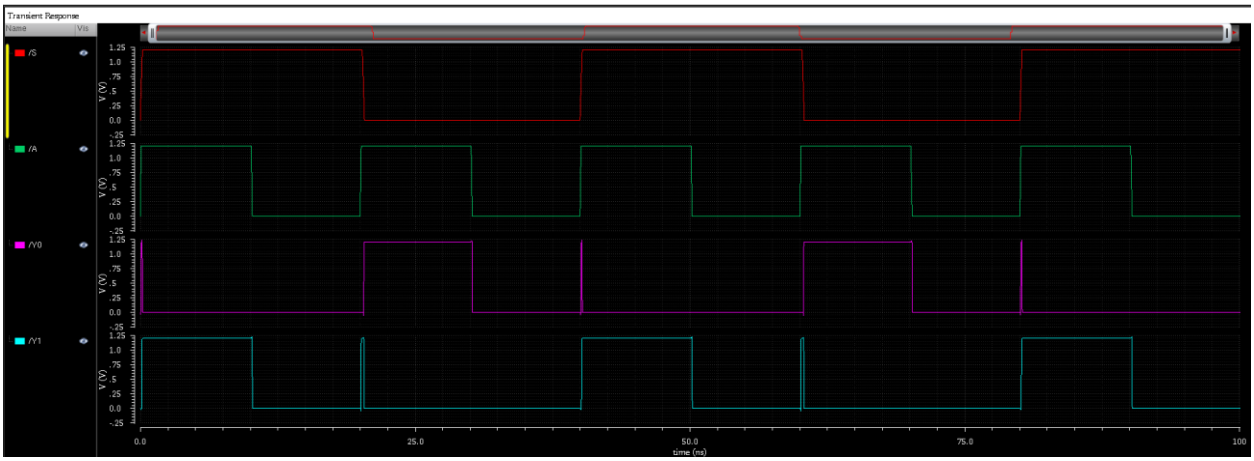


Simulation & Verification:

The simulation plot below shows the input waveforms and the resulting output waveforms. The results confirm that the data from input A is correctly routed to Y0 when S=0 and to Y1 when S=1.

S	A	Y0	Y1
0	0	0	0
0	1	1	0
1	0	0	0
1	1	0	1

The simulation successfully validates the design, confirming that the circuit performs the correct demultiplexing operation for all input combinations.



1 x 8 De-Multiplexer

Objective:

The objective is to design, implement, and verify a 1-to-8 demultiplexer (DEMUX) circuit using a hierarchical approach, building it entirely from pre-existing 1x2 demultiplexer standard cells. This process demonstrates the principles of hierarchical design and cell reuse to create more complex digital circuits.

Description:

A 1x8 demultiplexer is a combinational logic circuit with one data input (D), three select lines (S0, S1, S2), and eight outputs (Y0 to Y7). Its purpose is to route the single data input to one of the eight outputs, as determined by the binary value of the select lines.

The circuit is built by cascading 1x2 demultiplexer cells in a tree structure. A 1x8 DEMUX requires a total of seven 1x2 DEMUX units arranged in three stages:

- **Stage 1:** One 1x2 DEMUX with data input D and select line S2.
- **Stage 2:** Two 1x2 DEMUX units, with their inputs connected to the outputs of the Stage 1 DEMUX. The select line for these is S1.
- **Stage 3:** Four 1x2 DEMUX units, with their inputs connected to the outputs of the Stage 2 DEMUXs. The select line for these is S0.

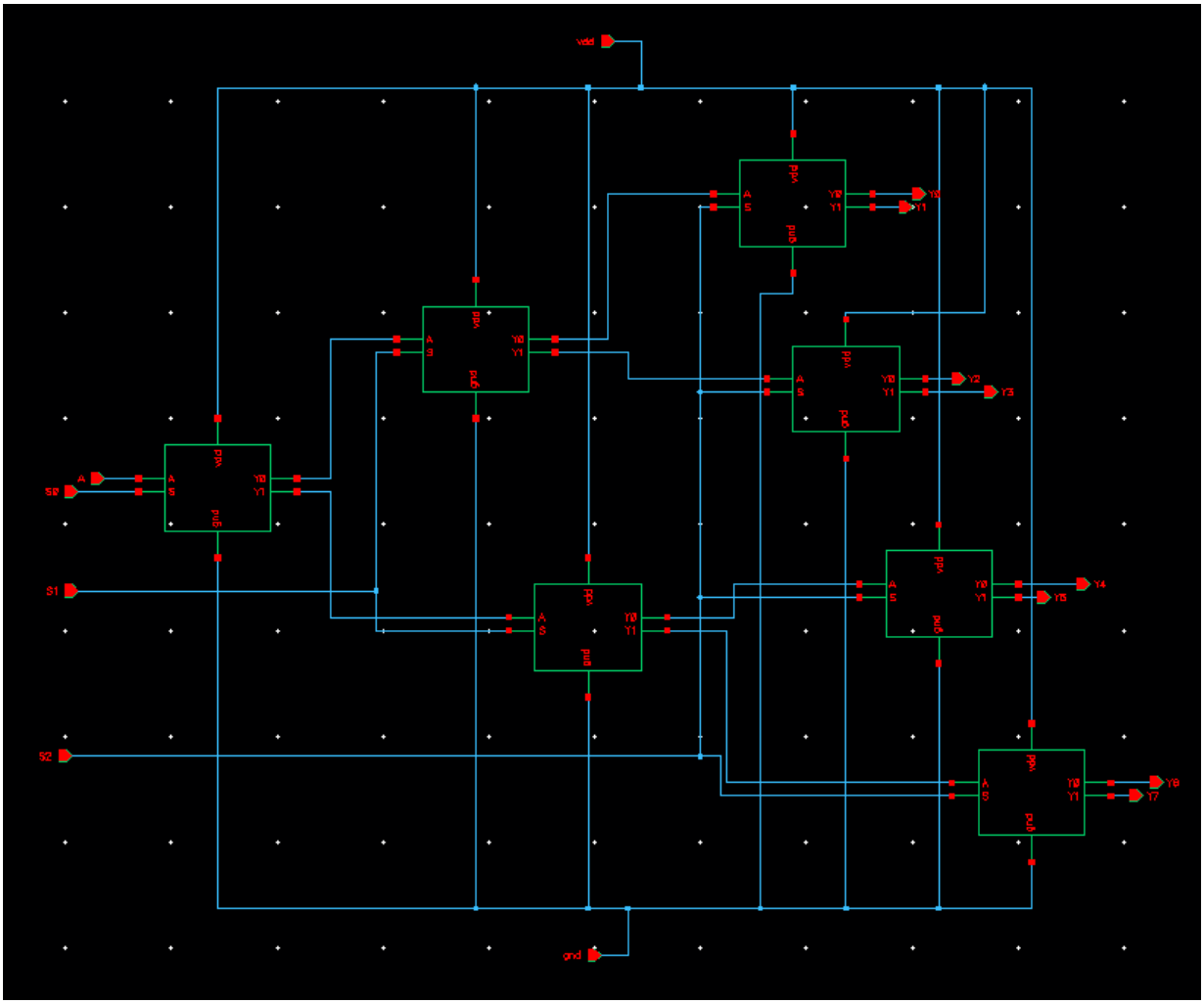
Design Steps:

1. **Hierarchical Schematic Design**: The design is created by instantiating and connecting seven of the pre-built DEMUX1X2 cells. The top-level schematic will show these cells in a clear, tree-like arrangement.
2. **Schematic Capture**: The data input **D** is connected to the first stage **DEMUX1X2**'s data pin, and the select line S2 is connected to its select pin. The two outputs of this first stage are routed to the data inputs of the two **DEMUX1X2** cells in the second stage. The select line for these two is S1. The four outputs from the second stage feed the data inputs of the four **DEMUX1X2** cells in the final, third stage, which are controlled by select line S0.
3. **Layout**: The layout is created by placing the seven DEMUX1X2 layouts. The cells are arranged to follow the hierarchical tree structure for clean and efficient routing. The select lines (S0, S1, S2) and the data input (D) are routed to minimize wire length and parasitic effects.

4. **DRC/LVS**: Perform a Design Rule Check (DRC) to ensure the layout adheres to all manufacturing rules. Execute a Layout Versus Schematic (LVS) to verify that the physical layout accurately represents the hierarchical schematic.
5. **Simulation Setup**: In the Cadence ADE L environment, a transient analysis is configured. Pulsed voltage sources are applied to the data input **D** and the three select lines S0, S1, and S2. All 8 combinations of the select lines are tested to ensure full functionality.
6. **Simulation & Verification**: The simulation is run, and the output waveforms for Y0 through Y7 are observed. The output on the selected line should mirror the input data, while all other outputs remain low. This is compared against the demultiplexer's truth table.

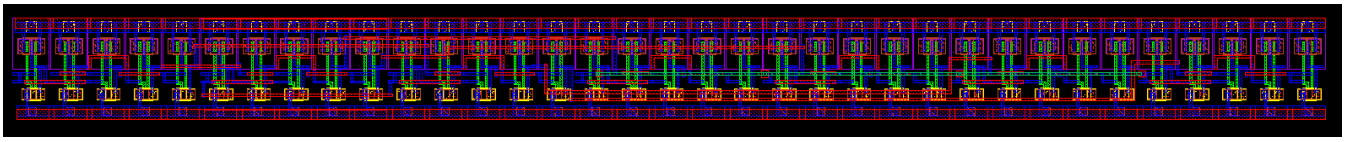
Schematic Diagram:

The schematic diagram clearly shows the cascading structure of the seven 1x2 DEMUX cells, with the select lines controlling each stage of the tree.



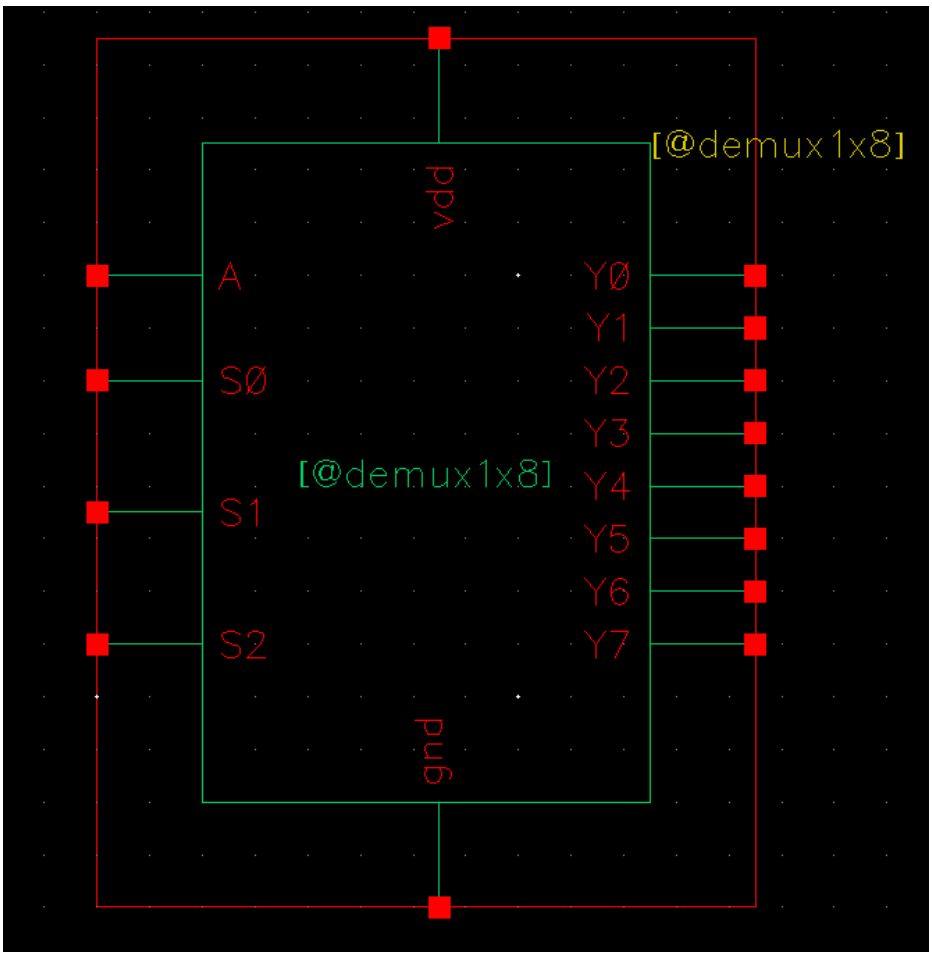
Layout:

The layout visually represents the physical placement of the DEMUX1X2 cells and the careful routing of signal and power lines to achieve a compact and functional design. (35 Transistors)



Symbol:

The symbol is a top-level representation of the 1x8 demultiplexer, showing its single data input, three select lines, and eight outputs.



Simulation & Verification:

The simulation plot below shows the waveforms for the inputs and outputs, confirming that the circuit performs the correct demultiplexing function. The output goes high only on the line corresponding to the binary value of the select lines. For example, when S2S1S0=011 (binary 3), the data is routed to output Y3.

S2	S1	S0	D	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
0	0	0	1	1	0	0	0	0	0	0	0
0	0	1	1	0	1	0	0	0	0	0	0
0	1	0	1	0	0	1	0	0	0	0	0
0	1	1	1	0	0	0	1	0	0	0	0
1	0	0	1	0	0	0	0	1	0	0	0
1	0	1	1	0	0	0	0	0	1	0	0
1	1	0	1	0	0	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0	1

The simulation confirms that the designed circuit successfully routes the data input to the selected output, validating its functionality for all eight possible input combinations.