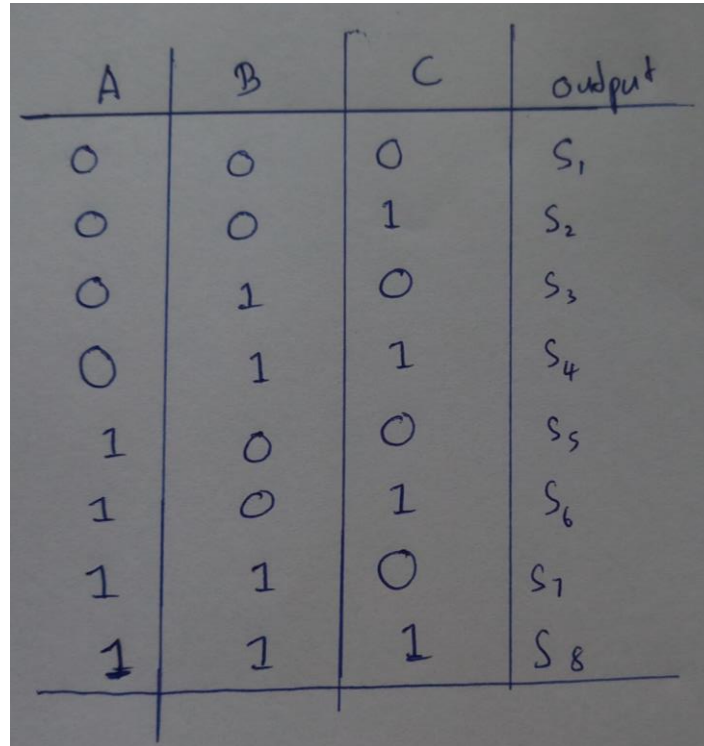


### Circuit\_2\_Part\_3

The task was to design a PLD circuit capable of implementing any three input combinational



| A | B | C | output         |
|---|---|---|----------------|
| 0 | 0 | 0 | S <sub>1</sub> |
| 0 | 0 | 1 | S <sub>2</sub> |
| 0 | 1 | 0 | S <sub>3</sub> |
| 0 | 1 | 1 | S <sub>4</sub> |
| 1 | 0 | 0 | S <sub>5</sub> |
| 1 | 0 | 1 | S <sub>6</sub> |
| 1 | 1 | 0 | S <sub>7</sub> |
| 1 | 1 | 1 | S <sub>8</sub> |

circuit. The truth-table of any three input combinational circuits will be as below.

Figure 3.3.1

Truth-table of any three input combinational circuit

Outputs S<sub>1</sub>, S<sub>2</sub>,..., S<sub>8</sub> differ with the combinational circuit. So we can write an expression for the combinational logic circuit using the 8 minterms. Which minterms to be selected differ according to the S<sub>1</sub>, S<sub>2</sub>,..., S<sub>8</sub>. If any S<sub>i</sub> is 1 then the corresponding minterm is taken into the sum of products expression. If S<sub>i</sub> is 0 that corresponding minterm is discarded.

So we can build the PLD with a fixed AND plane which has all eight minterms and a programmable OR plane which can be programmed using S<sub>i</sub> terms. So our PLD becomes a PROM.

Before building the PLD the AND plane and OR plane should be created. For the fixed AND plane, we need eight minterms. A minterm is a product of any three of A, A', B, B', C, or C'. So we need three input and gate. We configured a three-input AND gate using NAND, NOR, and NOT gates as below for better efficiency.

$$\begin{aligned}
 & A \cdot B \cdot C \\
 \Rightarrow & \overline{\overline{A \cdot B \cdot C}} \\
 \Rightarrow & \overline{\overline{A} \cdot \overline{B} \cdot \overline{C}}
 \end{aligned}$$

Using this expression we constructed the 3 input AND gates using a minimum number of logic gates.

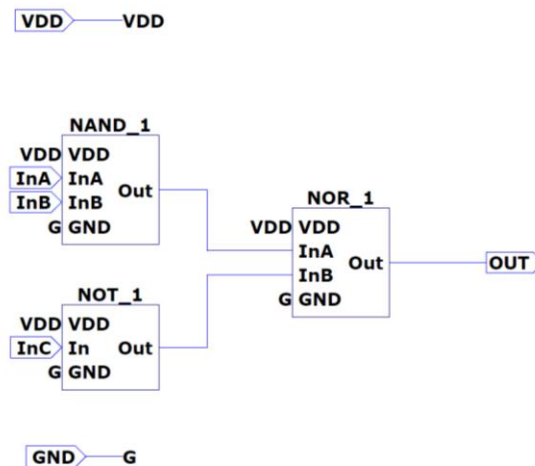


Figure 3.3.2

Implementing the three-input AND gate using NOR, AND, and NOT gates.

Using seven separate OR gates( 7 NOR gates + 7 NOT gates) to implement the OR plane, increases complexity and the latency of the circuit by a huge factor. Instead, we can simplify the expression and use a minimum number of gates as below.

$$\begin{aligned}
 & S_1 + S_2 + S_3 + S_4 + S_5 + S_6 + S_7 + S_8 \\
 \Rightarrow & \overline{\overline{S_1 + S_2 + S_3 + S_4 + S_5 + S_6 + S_7 + S_8}} \\
 \Rightarrow & \overline{S_1 + S_2 + S_3 + S_4} \cdot \overline{S_5 + S_6 + S_7 + S_8} \\
 \Rightarrow & \overline{S_1 + S_2} \cdot \overline{S_3 + S_4} \cdot \overline{S_5 + S_6} \cdot \overline{S_7 + S_8} \\
 \Rightarrow & \overline{S_1 + S_2} \cdot \overline{S_3 + S_4} + \overline{S_5 + S_6} \cdot \overline{S_7 + S_8} \\
 \Rightarrow & \overline{S_1 + S_2} \cdot \overline{S_3 + S_4} + \overline{S_5 + S_6} \cdot \overline{S_7 + S_8}
 \end{aligned}$$

Using this expression we were able to build an OR plane with a minimum number of components as below.

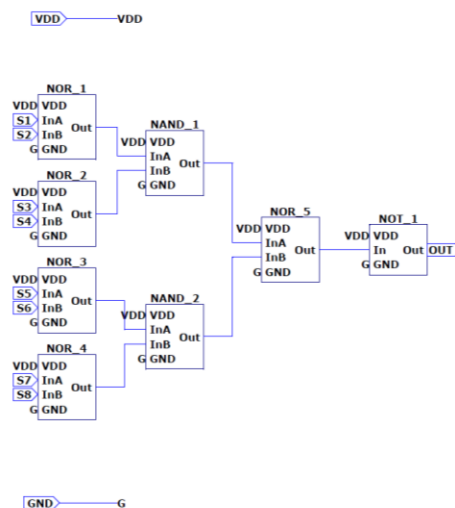


Figure 3.3.3

Implementing the OR plane

Instead of using a total of 14 logic gates, now we have implemented it using only 8 logic gates. This reduces the latency and complexity by a huge factor.

PROM is constructed as below

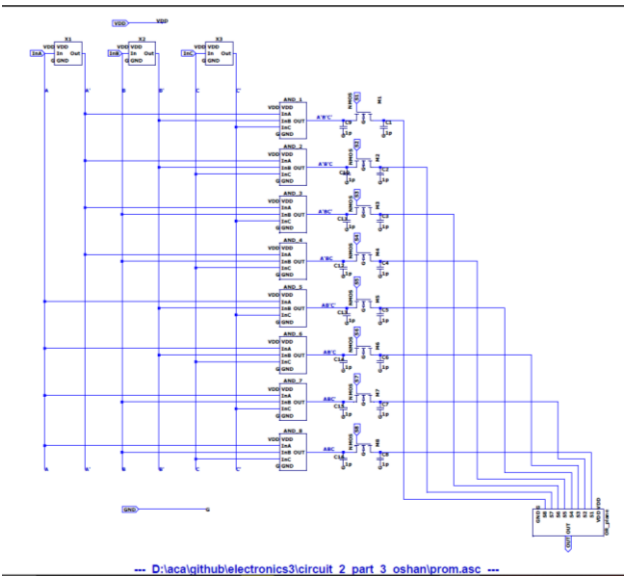


Figure 3.3.4

PROM circuit

We have used nmos transistors as switches which choose, which minterms are taken into the sum of products. We didn't choose passgates as switches as it increases the complexity and the latency of the circuit.

We tested the circuit for different combinational circuits by configuring Si switches. Below we have configured the PROM as a simple NOR gate.

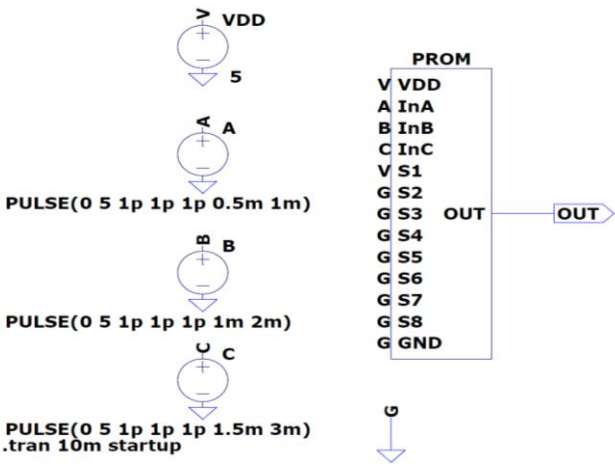


Figure 3.3.5

PROM configured as a NOR gate

Results were as below,

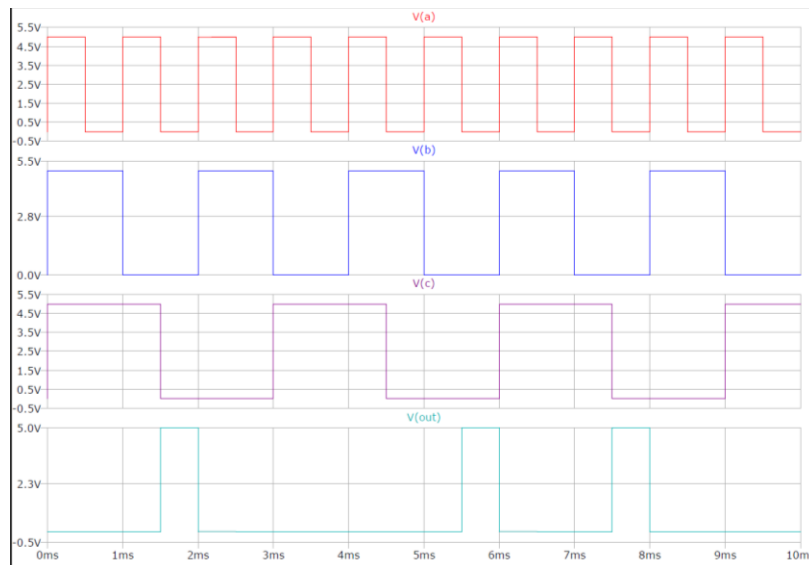


Figure 3.3.6

Results of PROM configured as a NOR gate

We can observe that the PROM is functioning correctly.