Date: **07/01/2026** **CS29204**: Switching Circuits Laboratory

Assignment 1

Building CMOS Gates and Simple Circuits

Group 36
Dipam Sen (24CS10059)
Mitali Laddha (24CS10111)

Part (a)

Aim

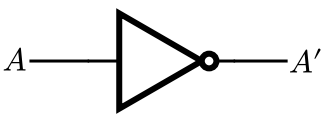
To realize a CMOS inverter using the IC 4007, study its transfer characteristics, and determine its high and low noise margins.

Apparatus

1. Breadboard
2. IC 4007 × 1 (CMOS IC)
3. Potentiometer
4. Digital Storage Oscilloscope (DSO)
5. Wires, wire cutter, tweezer

Theory

An inverter (NOT gate) is a basic logic gate which has one input and one output. It outputs the opposite logic level to its input. If the input is high (logic 1), the output is low (logic 0), and vice versa. Figure 1 shows the circuit symbol and truth table for the inverter.



A	A'
0	1
1	0

Figure 1: (a) Circuit symbol of NOT gate (b) Truth table for NOT

We can realise an inverter by using a pMOS and an nMOS transistor connected in a CMOS (complementary MOSFET) configuration as shown in Figure 2.

For low input (gate) voltages, the nMOS transistor is off while the pMOS transistor conducts, resulting in a high output voltage close to V_{DD}. Conversely, for high input voltages, the pMOS transistor is off while the nMOS transistor conducts, pulling the output voltage close to ground (0 V).

The transition region occurs when both transistors are partially on, leading to a rapid change in output voltage with respect to the input voltage. The transfer characteristics of the inverter can be plotted by measuring the output voltage (V_{OUT}) as a function of the input voltage (V_{IN}).

Circuit Diagrams

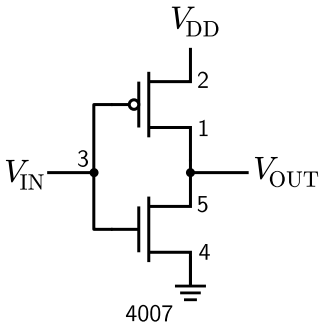


Figure 2: CMOS Inverter Circuit

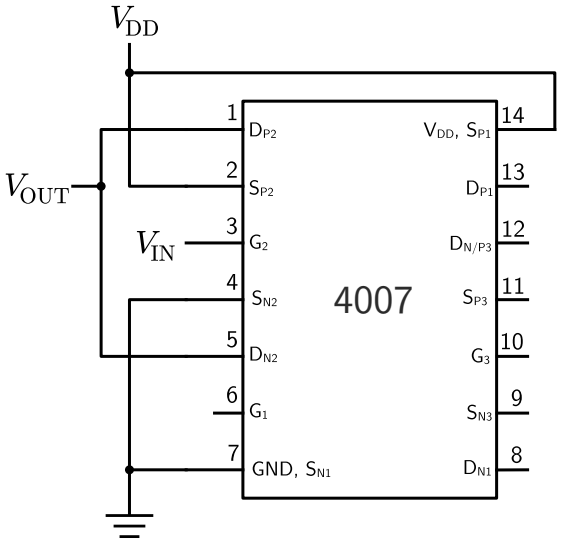


Figure 3: Connection diagram for a CMOS inverter

Procedure

1. Connect the CMOS inverter circuit on the breadboard as shown in Figure 3.
2. Connect the input of the inverter to a potentiometer to vary the input voltage from 0 V to V_{DD} (5 V).
3. Connect the input and output of the inverter to a Digital Storage Oscilloscope (DSO) and measure the voltages.
4. Vary the input voltage from 0 V to 5 V in small increments and record the corresponding output voltage.
5. Plot the transfer characteristics (V_{OUT} vs V_{IN}) of the CMOS inverter.
6. Determine the high and low noise margins from the transfer characteristics.

Observations and Calculations

V_{IN} (V)	V_{OUT} (V)
0.07	5.04
1.24	5.04
1.90	4.96
2.22	4.71
2.36	4.52
2.49	4.12
2.55	3.71
2.58	2.61
2.60	1.21
2.63	0.80
2.68	0.50
2.82	0.32
3.04	0.14
3.41	0.04
4.11	0.06
5.04	0.06

Table 1: Input-output voltage measurements for the CMOS Inverter

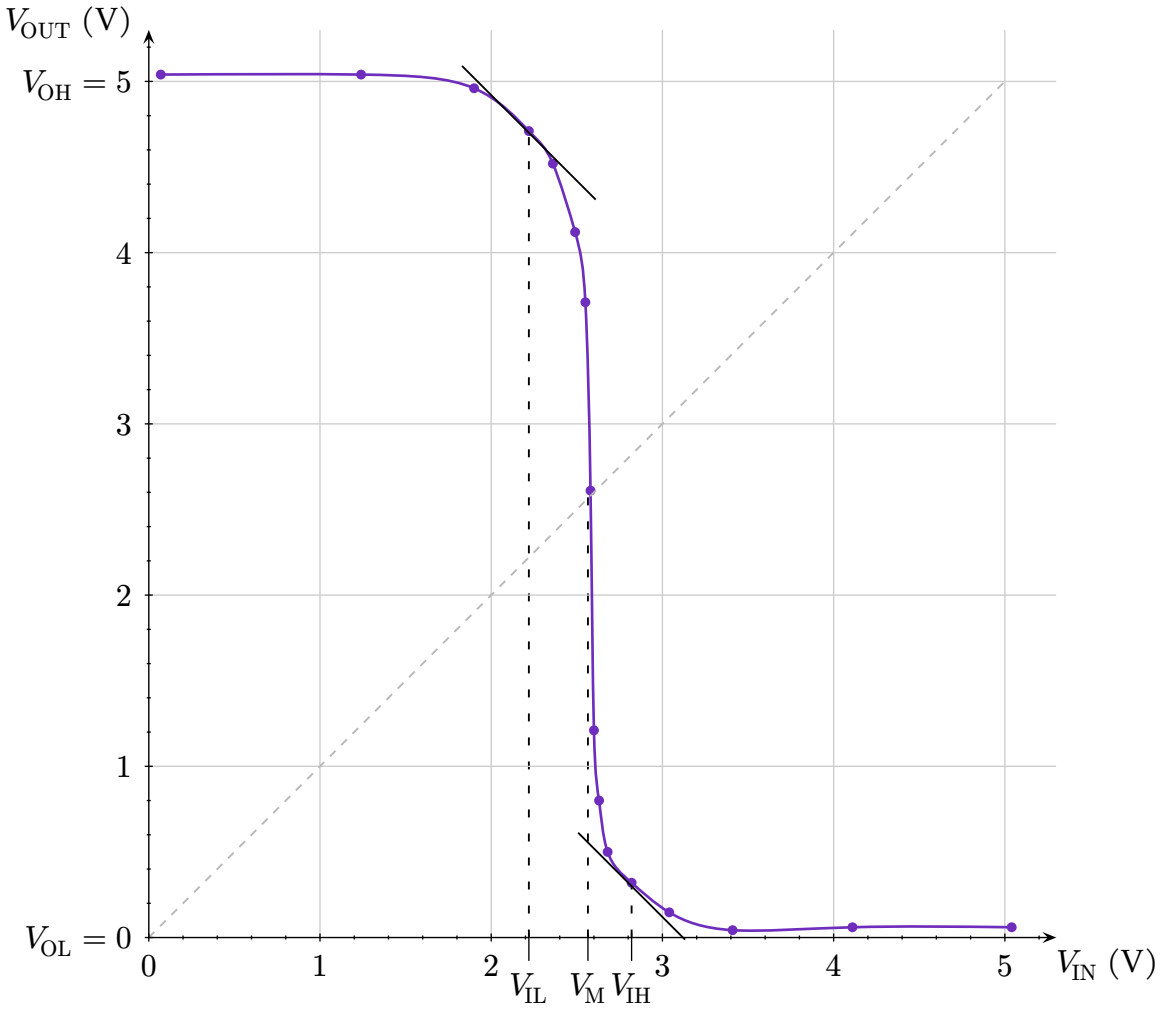


Figure 4: Voltage Transfer Characteristics of CMOS Inverter

From the voltage transfer characteristics graph (Figure 4), we can determine the following voltages:

$$\begin{aligned} V_{\text{OH}} &= 5\ \text{V} \quad && \text{(output high)} \\ V_{\text{OL}} &= 0\ \text{V} \quad && \text{(output low)} \\ V_{\text{IH}} &= 2.82\ \text{V} \quad && \text{(input high threshold)} \\ V_{\text{IL}} &= 2.22\ \text{V} \quad && \text{(input low threshold)} \end{aligned}$$

The noise margins can be calculated as follows:

$$\begin{aligned} \text{NM}_H &= V_{\text{OH}} - V_{\text{IH}} = 5 - 2.82 = 2.18\ \text{V} \\ \text{NM}_L &= V_{\text{IL}} - V_{\text{OL}} = 2.22 - 0 = 2.22\ \text{V} \end{aligned}$$

Result

A CMOS inverter was successfully built using the IC 4007. The transfer characteristics were plotted, and the high and low noise margins were determined as:

$$\begin{aligned} \text{NM}_H &= 2.18\ \text{V} \\ \text{NM}_L &= 2.22\ \text{V} \end{aligned}$$

Discussion

The CMOS inverter exhibits the expected behaviour of inverting the input signal. Near logic low inputs (0 V), the output remains high (close to 5 V), and near logic high inputs (5 V), the output drops to low (close to 0 V).

In the transition region between 2 V and 3 V, the output voltage changes rapidly from a high to a low state. In this region, both the pMOS and nMOS are partially conducting, leading to the sharp change in output voltage. The sharp transition also indicates that the inverter has good switching characteristics. The calculated noise margins, $\mathrm{NM}_H = 2.18\,\mathrm{V}$ and $\mathrm{NM}_L = 2.22\,\mathrm{V}$, suggest that the inverter can tolerate a reasonable amount of noise on its input without causing erroneous output states.

Conclusion

The circuit realised using the IC 4007 functioned correctly as an inverter. The experimentally observed transfer characteristics are consistent with the expected behaviour.

Part (b)

Aim

To realise a 3-input CMOS NAND gate and a 3-input CMOS NOR gate using the IC 4007 and verify their truth tables.

Apparatus

1. Breadboard
2. IC 4007 × 2 (CMOS IC)
3. Wires, wire cutter, tweezer

Theory

A NAND gate is a digital logic gate that outputs a low (0) only when all its inputs are high (1). A NOR gate outputs a high (1) only when all its inputs are low (0). Table 2 and Table 3 show the truth tables for 3-input NAND and NOR gates, respectively.

A	B	C	$Y = (A \cdot B \cdot C)'$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Table 2: Truth table for 3-input NAND gate

A	B	C	$Y = (A + B + C)'$
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Table 3: Truth table for 3-input NOR gate

We can realise 3-input NAND and NOR gates using three pairs of pMOS and nMOS transistors connected in CMOS configuration.

In the NAND gate, the pMOS transistors connected in parallel form the pull-up network, whereas the nMOS transistors connected in series form the pull-down network. For the output to be pulled low, all nMOS transistors must conduct, meaning that all inputs must be high. Conversely, if any input is low, at least one pMOS transistor will conduct, pulling the output high.

Similarly, the NOR gate operates in the exact opposite manner, with the pull-up network formed by pMOS transistors in series and the pull-down network formed by nMOS transistors in parallel. The output is high only when all inputs are low.

Circuit Diagrams

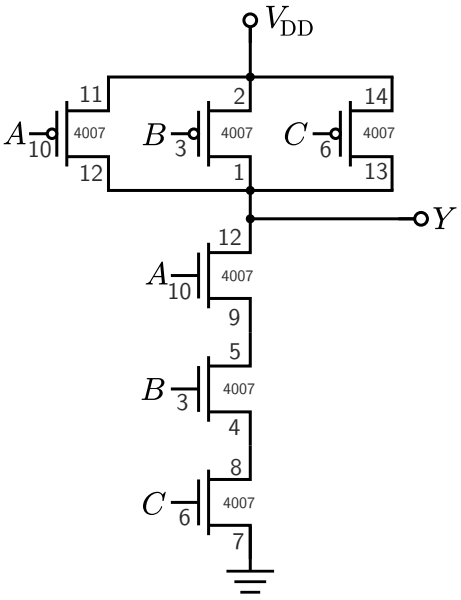


Figure 5: CMOS 3-input NAND Gate Circuit

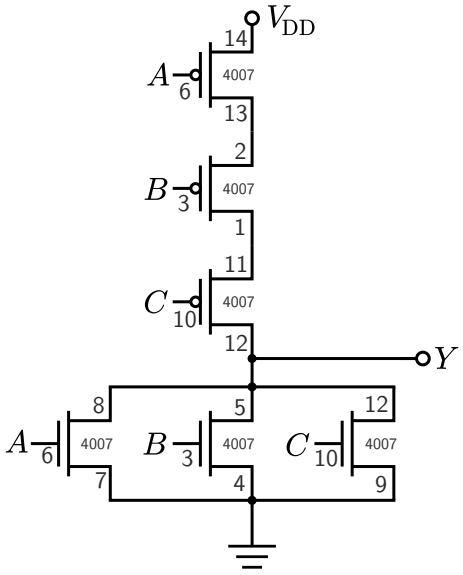


Figure 6: CMOS 3-input NOR Gate Circuit

Procedure

1. Connect the CMOS NAND gate circuit on the breadboard as shown in Figure 5.
2. Connect the inputs A, B, and C to logic high (5 V) or logic low (0 V) using wires.
3. Observe the output Y for all possible combinations of inputs and verify the truth table.
4. Repeat steps 1-3 for the CMOS NOR gate circuit as shown in Figure 6.

Result

The 3-input CMOS NAND and NOR gates were successfully realised using the IC 4007, and their truth tables were verified experimentally.

Discussion

The CMOS NAND and NOR gates functioned correctly according to their truth tables. For the NAND gate, the output was low only when all three inputs were high, while for the NOR gate, the output was high only when all three inputs were low. This behaviour is consistent with the expected operation of these logic gates.

Conclusion

The CMOS NAND and NOR gates functioned correctly according to their truth tables when implemented using the IC 4007.

Part (c)

Aim

To design and implement a ring oscillator by using an odd number of TTL and CMOS NOT gates, measure its oscillation frequency and estimate the propagation delay of the gates used, for $N = 3$ and $N = 5$.

Apparatus

1. Breadboard
2. IC 7404 × 1 (TTL Hex Inverter)
3. IC 4009 × 1 (CMOS Hex Inverter)
4. Digital Storage Oscilloscope (DSO)
5. Wires, wire cutter, tweezer

Theory

A ring oscillator is an electronic circuit that consists of an odd number of inverters (NOT gates) connected in a series, with the output of the last inverter fed back to the input of the first. This configuration creates a feedback loop that causes the signal at any point in the loop to oscillate between high and low states. Figure 7 shows the circuit diagram of a 3-stage ring oscillator.

The oscillation frequency of a ring oscillator is determined by the number of inverters (N) and the propagation delay (T) of each inverter. The total delay for one complete oscillation cycle is given by:

$$P = 2NT$$

where P is the period of oscillation. The frequency (f) can be calculated as:

$$f = \frac{1}{P} = \frac{1}{2NT}$$

By measuring the oscillation frequency, we can estimate the propagation delay of the gates used in the ring oscillator.

Circuit Diagrams

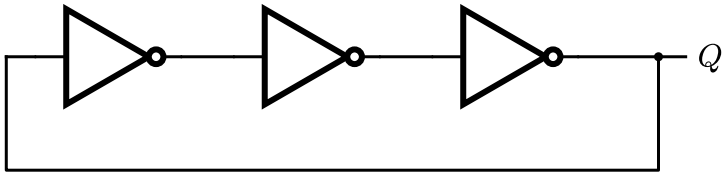


Figure 7: Circuit diagram of a 3-stage ring oscillator

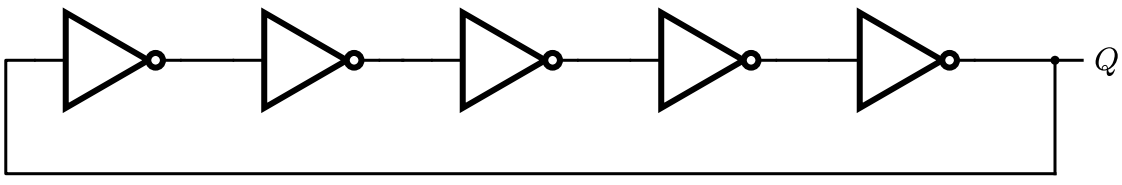


Figure 8: Circuit diagram of a 5-stage ring oscillator

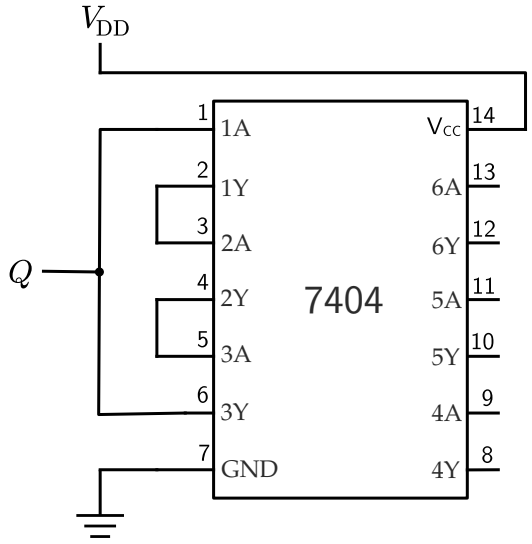


Figure 9: Connection diagram for a 3-stage TTL ring oscillator

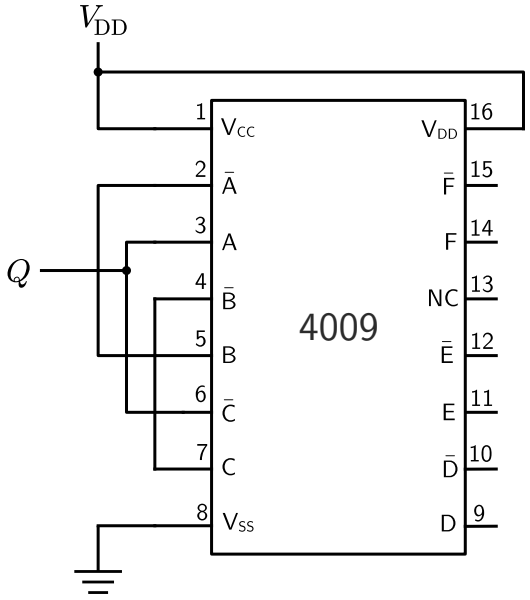


Figure 10: Connection diagram for a 3-stage CMOS ring oscillator

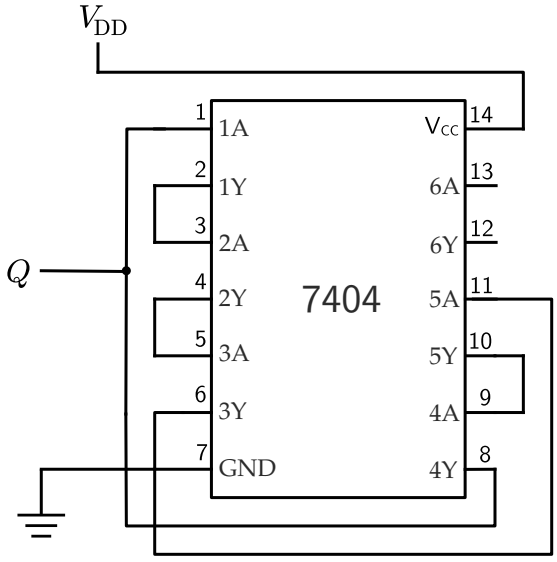


Figure 11: Connection diagram for a 5-stage TTL ring oscillator

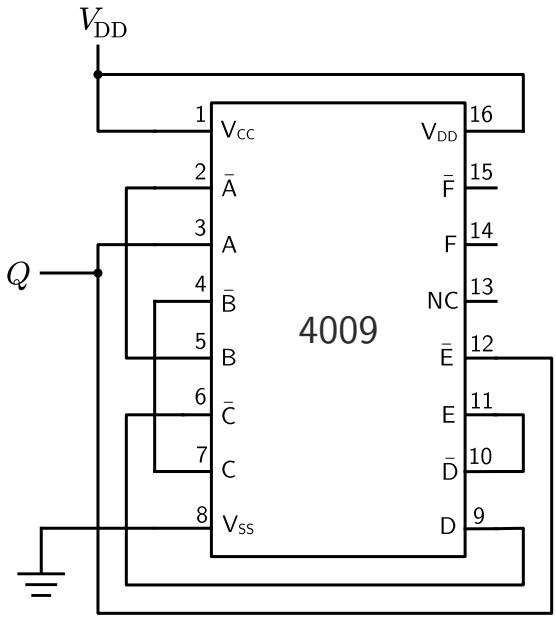


Figure 12: Connection diagram for a 5-stage CMOS ring oscillator

Procedure

1. Connect the ring oscillator circuit on the breadboard using TTL NOT gates as shown in Figure 9.
2. Connect one point in the ring oscillator to a Digital Storage Oscilloscope (DSO) and measure the oscillation frequency using the DSO.
3. Calculate the propagation delay of the gates using the measured frequency.
4. Repeat the procedure for both $N = 3$ and $N = 5$ stages.
5. Repeat the entire procedure using CMOS NOT gates.

Observations and Calculations

	N	f	$T = \frac{1}{2Nf}$	T_{avg}
TTL IC 7404	3	51.25 MHz	3.269 ns	3.257 ns
	5	30.59 MHz	3.252 ns	
CMOS IC 4009	3	6.115 MHz	27.25 ns	26.415 ns
	5	3.909 MHz	25.58 ns	

Table 4: Measured frequencies and calculated propagation delays for ring oscillators

- Average propagation delay for TTL NOT gate (IC 7404): $T_{\text{avg,TTL}} = 3.257$ ns
- Average propagation delay for CMOS NOT gate (IC 4009): $T_{\text{avg,CMOS}} = 26.415$ ns

Result

Ring oscillators using both TTL and CMOS NOT gates were successfully implemented for $N = 3$ and $N = 5$. The oscillation frequencies were measured, and the propagation delays were found to be:

$$T_{\text{avg,TTL}} = 3.257 \text{ ns}$$
$$T_{\text{avg,CMOS}} = 26.415 \text{ ns}$$

Discussion

When an odd number of NOT gates are cascaded in a ring configuration, the output of the last gate is fed back to the input of the first gate. Due to this feedback loop, the system cannot settle into a stable state, but instead the voltage levels at any point in the loop continuously oscillates between the high and low states. By measuring this oscillation frequency, we can estimate the propagation delay of the gates used in the ring oscillator.

The measured propagation delays for the NOT gates indicate that TTL gates have a significantly lower propagation delay compared to CMOS gates. TTL technology offers faster switching speeds, as it uses BJT transistors. In contrast, CMOS technology, while being more power-efficient, has higher propagation delays due to the capacitive nature of MOSFETs.

Conclusion

Ring oscillators using TTL and CMOS NOT gates were successfully realised, and their oscillation frequencies were measured. The propagation delays were estimated, confirming the expected performance differences between TTL and CMOS technologies.

Part (d)

Aim

To implement (i) a 4-bit binary-to-Gray code converter and (ii) a 4-bit Gray-to-binary code converter using TTL XOR IC 7486 and verify their truth tables.

Apparatus

1. Breadboard
2. IC 7486 × 2 (TTL Quad 2-input XOR gate)
3. Wires, wire cutter, tweezer

Theory

Gray code is a type of non weighted, cyclic binary code, where two successive values differ in only one bit position. This property makes Gray code particularly useful in various applications where minimizing errors during transitions is crucial.

A binary-to-Gray code converter transforms a binary number into its corresponding Gray code representation. Conversely, a Gray-to-binary code converter transforms a Gray code back into its binary equivalent. Both conversions can be implemented using XOR gates.

The conversion from 4-bit binary to Gray code can be achieved using the following equations:

$$g_3 = b_3$$
$$g_2 = b_3 \oplus b_2$$
$$g_1 = b_2 \oplus b_1$$
$$g_0 = b_1 \oplus b_0$$

where b_3, b_2, b_1, and b_0 are the bits of the binary number, and g_3, g_2, g_1, and g_0 are the bits of the Gray code.

Similarly, the conversion from Gray code to binary can be achieved using the following equations:

$$b_3 = g_3$$
$$b_2 = b_3 \oplus g_2$$
$$b_1 = b_2 \oplus g_1$$
$$b_0 = b_1 \oplus g_0$$

b_3	b_2	b_1	b_0	g_3	g_2	g_1	g_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

Table 5: 4-bit binary numbers, and their Gray code equivalents

Circuit Diagrams

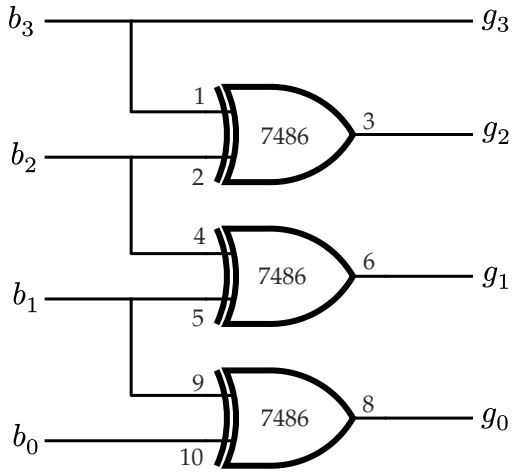


Figure 13: Circuit diagram of 4-bit Binary to Gray code converter

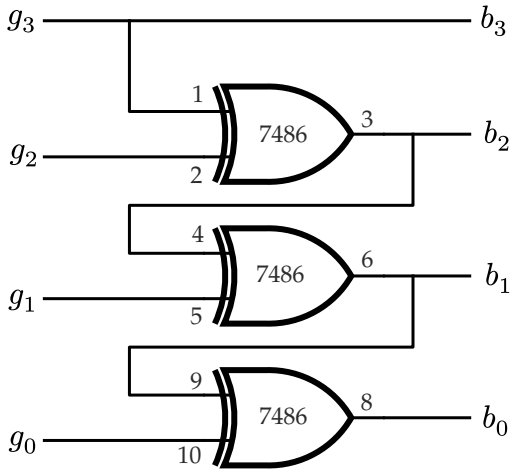


Figure 14: Circuit diagram of 4-bit Gray to Binary code converter

Procedure

1. Connect the 4-bit binary-to-Gray code converter circuit on the breadboard as shown in Figure 13.
2. Connect the binary inputs (b_3, b_2, b_1, b_0) to the input lines using wires.
3. Observe the Gray code outputs (g_3, g_2, g_1, g_0) for all possible combinations of binary inputs and verify the truth table (Table 5).
4. Repeat steps 1-3 for the 4-bit Gray-to-binary code converter circuit as shown in Figure 14.

Result

The 4-bit binary-to-Gray code converter and Gray-to-binary code converter were successfully realised using the IC 7486, and their truth tables were verified experimentally.

Discussion

The binary-to-Gray code converter functioned correctly according to its truth table. For all input combinations, the output Gray code matched the expected values

derived from the conversion equations. Similarly, the Gray-to-binary code converter also functioned correctly, accurately converting Gray code inputs back to their binary equivalents.

The use of XOR gates in both converters allowed for efficient implementation of the conversion logic.

Conclusion

The 4-bit binary-to-Gray code converter and Gray-to-binary code converter functioned correctly according to their truth tables when implemented using the IC 7486.