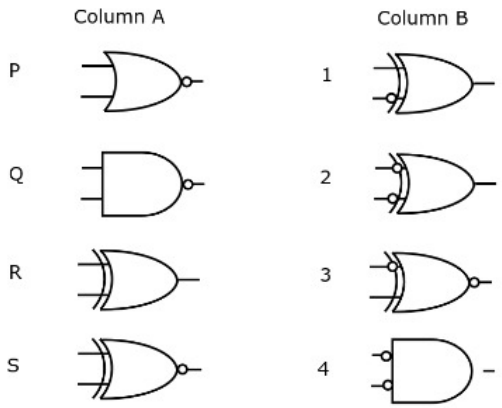
**Name:** Ashok Kumar Reddy K
**Batch:** 2
**ID:** cometfwc016
**Date:** 9th July 2025
**Email:** kothapalli.fwc1@iiitb.ac.in
**Institute:** IIITB – COMET Foundation

# GATE Question Paper 2010, EC Question Number 411

## Question 11 Analysis

**Question:**

match the logic gates in **column A** with equivalent to the **column 2**




(A) P-2,Q-4,R-1,S-3 (B) P-4,Q-2.R-1,S-3
(C) P-2,Q-4,R-3,S-1 (D) P-4,Q-2,R-3,S-1

## Truth Tables for Custom Logic Gates

1. **1st Gate:** Output = NOT$(A \vee B)$
2. **2nd Gate:** Output = NOT$(A \wedge B)$
3. **3rd Gate:** Output = $A \oplus B$
4. **4th Gate:** Output = NOT$(A \oplus B)$
5. **5th Gate:** Output = $A \oplus$ NOT$(B)$
6. **6th Gate:** Output = NOT$(A) \vee$ NOT$(B)$
7. **7th Gate:** Output = NOT(NOT$(A) \oplus B$)
8. **8th Gate:** Output = NOT$(A) \wedge$ NOT$(B)$

## Truth Tables

- **1st Gate**

| A | B | OUT |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

- **2nd Gate**

| A | B | OUT |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

- **3rd Gate**

| A | B | OUT |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

- **4th Gate**

| A | B | OUT |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

- **5th Gate**

| A | B | OUT |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

- **6th Gate**

| A | B | OUT |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

- **7th Gate**

| A | B | OUT |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

- **8th Gate**

| A | B | OUT |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

## Matching Gate Outputs

- 1st == 8th
- 2nd == 6th
- 3rd == 7th
- 4th == 5th

the correct answer is (D) P-4,Q-2,R-3,S-1

## Brief Discussion

The question involves matching logic gates from **Column A** with equivalent gates from **Column B**. The hardware implementation validates the equivalence of each gate by comparing outputs using LED indicators on an Arduino Uno board.

## Abstract

This project implements four logic gates shown in Column A and verifies their output against equivalent gates in Column B using an Arduino Uno. Each logic gate is recreated using basic gate logic with push-button inputs $A$ and $B$. The outputs are displayed using LEDs, enabling direct observation and comparison.

## Hardware Requirements

| S.No | Component |
|---|---|
| 1 | Arduino Uno Board |
| 2 | Breadboard |
| 3 | Push Buttons (2) |
| 4 | LEDs (4) |
| 5 | Resistors: 220Ω, 10kΩ |
| 6 | Jumper Wires |
| 7 | USB Cable |

**Table 1: Arduino Hardware Components**

## Pin Connections

| Component | Arduino Pin |
|---|---|
| Input A (Button 1) | Digital 2 |
| Input B (Button 2) | Digital 3 |
| LED for Gate P Output | Digital 8 |
| LED for Gate Q Output | Digital 9 |
| LED for Gate R Output | Digital 10 |
| LED for Gate S Output | Digital 11 |
| GND | GND |
| VCC | 5V |

**Table 2: Pin Mapping for Arduino Uno**

## Upload Steps

1. Connect Arduino Uno to mobile using OTG cable.
2. Open the ArduinoDroid app.
3. Select “Arduino Uno” as board type.
4. Paste the sketch to implement each gate logic (NAND, NOR, XOR, etc.).
5. Compile and upload the code to Arduino.
6. Press input buttons and observe LED outputs.

## Observations (Truth Table)

| A | B | Gate P | Gate Q | Gate R | Gate S |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |

**Table 3: Logic Gate Output using Arduino LEDs**

## Matching Gates

- **P: NOR** → Gate 1 (OR + NOT)
- **Q: NAND** → Gate 2 (AND + NOT)
- **R: XOR** → Gate 3 (Exclusive OR)
- **S: XNOR** → Gate 4 (NOT of XOR)

**Correct Option: (D) P-4, Q-2, R-3, S-1**

## Conclusion

- Each gate in Column A was implemented and tested using Arduino Uno.
- The observed outputs matched the expected behavior of equivalent gates in Column B.
- This confirms the correct matching as: **P-4, Q-2, R-3, S-1**.

## Brief Discussion

This experiment matches logic gates from Column A to equivalent logic gates in Column B based on their truth tables. The circuit is implemented practically using Raspberry Pi Pico 2 W and 4 LEDs to verify the logic.

## Abstract

This document presents a hardware-based comparison of four logic gates using the Raspberry Pi Pico 2 W board. Each gate from Column A is implemented, and its output is verified using LEDs. The behavior is matched against equivalent gates from Column B, and the experiment confirms the logic gate identities using MicroPython.

## Implementation using Raspberry Pi Pico 2 W

**Hardware Requirements:**

| S.No | Component |
|---|---|
| 1 | Raspberry Pi Pico 2 W |
| 2 | Breadboard |
| 3 | Push Buttons (2) |
| 4 | LEDs (4) |
| 5 | Resistors: 220Ω, 10kΩ |
| 6 | Jumper Wires |
| 7 | Micro USB Cable |

**Table 1: Pico 2 W Hardware Components**

**Pin Connections:**

| Component | Pico 2 W Pin |
|---|---|
| Input A (Button 1) | GP14 |
| Input B (Button 2) | GP15 |
| LED for Gate P (NOR) | GP10 |
| LED for Gate Q (NAND) | GP11 |
| LED for Gate R (XOR) | GP12 |
| LED for Gate S (XNOR) | GP13 |
| GND | GND |
| VCC | 3.3V |

**Table 2: GPIO Pin Mapping for Pico 2 W**

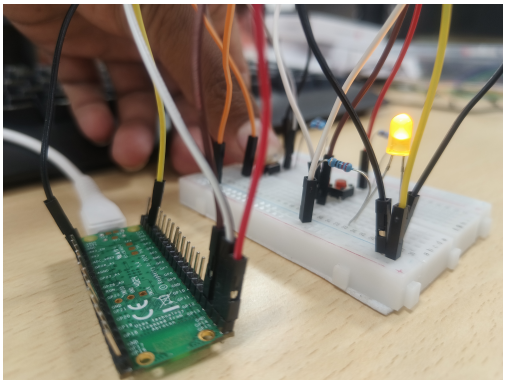
## Upload Steps

1. Connect Pico 2 W to mobile via OTG while pressing BOOTSEL.
2. Flash MicroPython firmware (.uf2) onto the board.
3. Open MicroREPL or uPyCraft on mobile.
4. Connect to the Pico 2 W and paste MicroPython logic gate implementation.
5. Assemble the circuit on breadboard as per pin mapping.
6. Press button inputs and observe LEDs for each gate output.

## Observations (Truth Table)

| A | B | Gate P | Gate Q | Gate R | Gate S |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |

**Table 3: Gate Output on Pico 2 W using LEDs**



## Conclusion

- Each logic gate in Column A was successfully implemented using Pico 2 W.
- The outputs observed on LEDs matched their equivalent Column B gate behavior.
- Correct matching: **P-4, Q-2, R-3, S-1** (Option D).
- The Raspberry Pi Pico 2 W setup validated theoretical logic gate behavior through practical observation.

**GitHub Repository:** `https://github.com/ashok-kumar-reddy-17/Ashok_FWC`