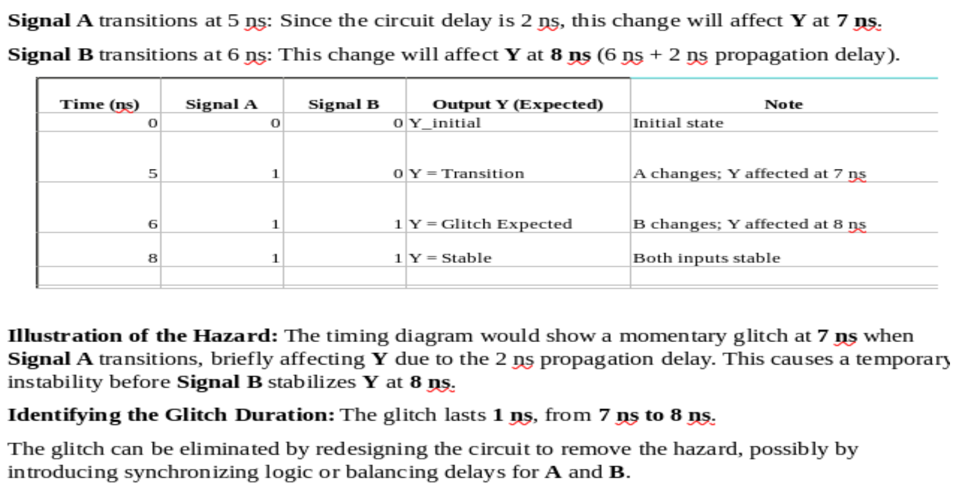
**Q1. A timing diagram shows a combinational logic circuit with two input signals, A and B, and one output, Y. Signal A transitions at 5 ns, and signal B transitions at 6 ns. The propagation delay of the circuit is 2 ns. A glitch occurs at the output due to a timing hazard. Illustrate the hazard in the timing diagram by drawing the expected output and identifying the glitch duration.**



**Signal A** transitions at 5 ns: Since the circuit delay is 2 ns, this change will affect **Y** at **7 ns.**

**Signal B** transitions at 6 ns: This change will affect **Y** at **8 ns** (6 ns + 2 ns propagation delay).

| Time (ns) | Signal A | Signal B | Output Y (Expected) | Note |
|---|---|---|---|---|
| 0 | 0 | 0 | Y_initial | Initial state |
| 5 | 1 | 0 | Y = Transition | A changes; Y affected at 7 ns |
| 6 | 1 | 1 | Y = Glitch Expected | B changes; Y affected at 8 ns |
| 8 | 1 | 1 | Y = Stable | Both inputs stable |
|  |  |  |  |  |

**Illustration of the Hazard:** The timing diagram would show a momentary glitch at **7 ns** when **Signal A** transitions, briefly affecting **Y** due to the 2 ns propagation delay. This causes a temporary instability before **Signal B** stabilizes **Y** at **8 ns.**

**Identifying the Glitch Duration:** The glitch lasts **1 ns**, from **7 ns to 8 ns.**

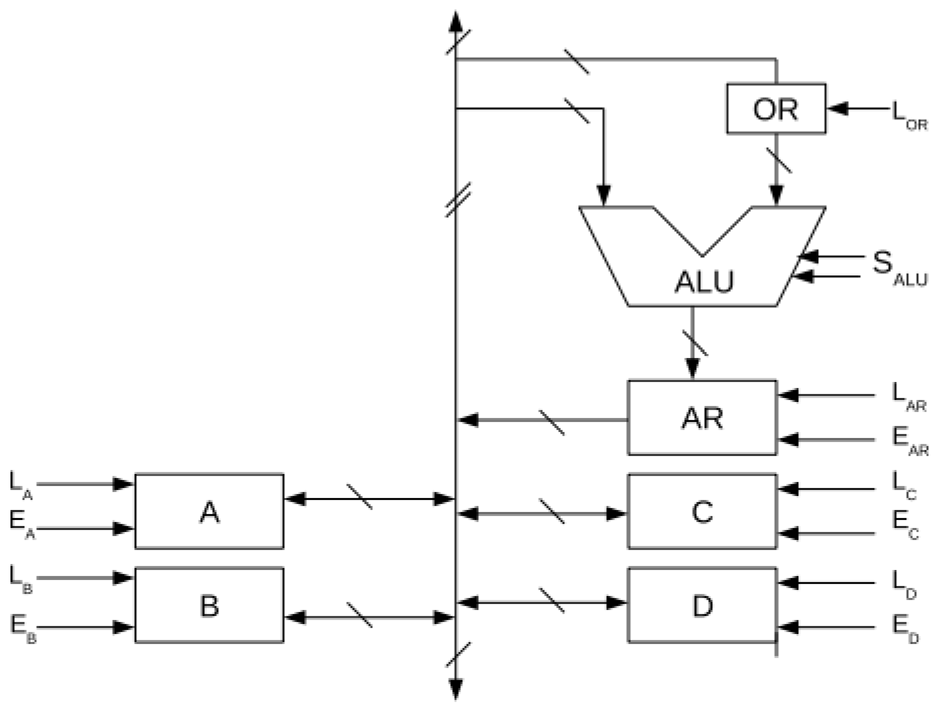
The glitch can be eliminated by redesigning the circuit to remove the hazard, possibly by introducing synchronizing logic or balancing delays for **A** and **B**.

Answer:

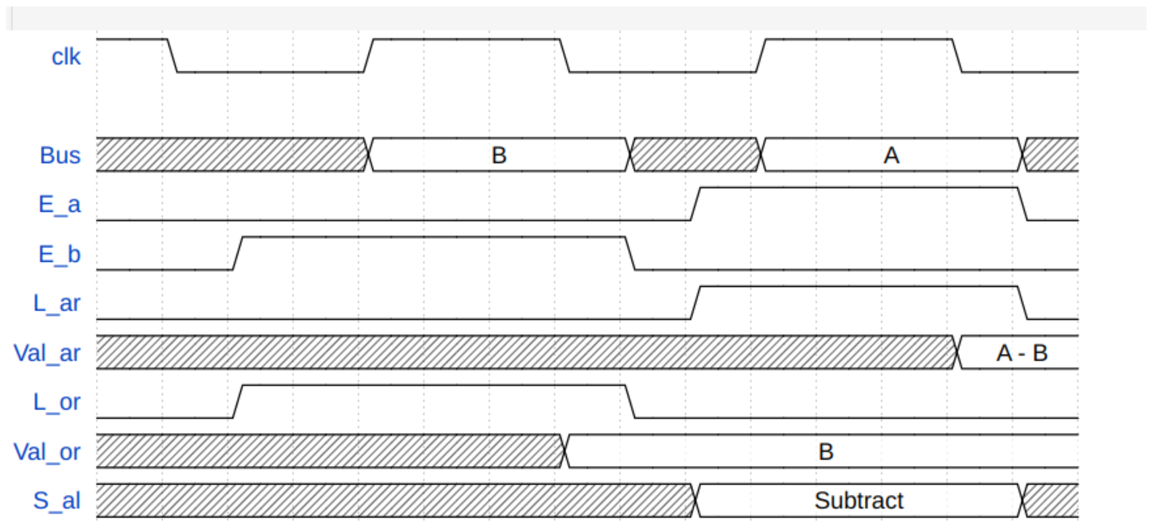
Q2: **In a system with a single bus shared by the CPU and a printer, both devices need access to the bus. If both try to access the bus at the same time, what could happen?**

Answer: In a system where multiple devices (like the CPU and a printer) share a single bus, bus contention can occur when both devices try to access the bus at the same time. This results in delays or conflicts, and the system needs an arbitration scheme to determine which device gets access to the bus.

Q3: **Given the following ALU design, Give the timing diagram for performing the operation (A-B) and storing it in AR: (Dont use registers C and D)**



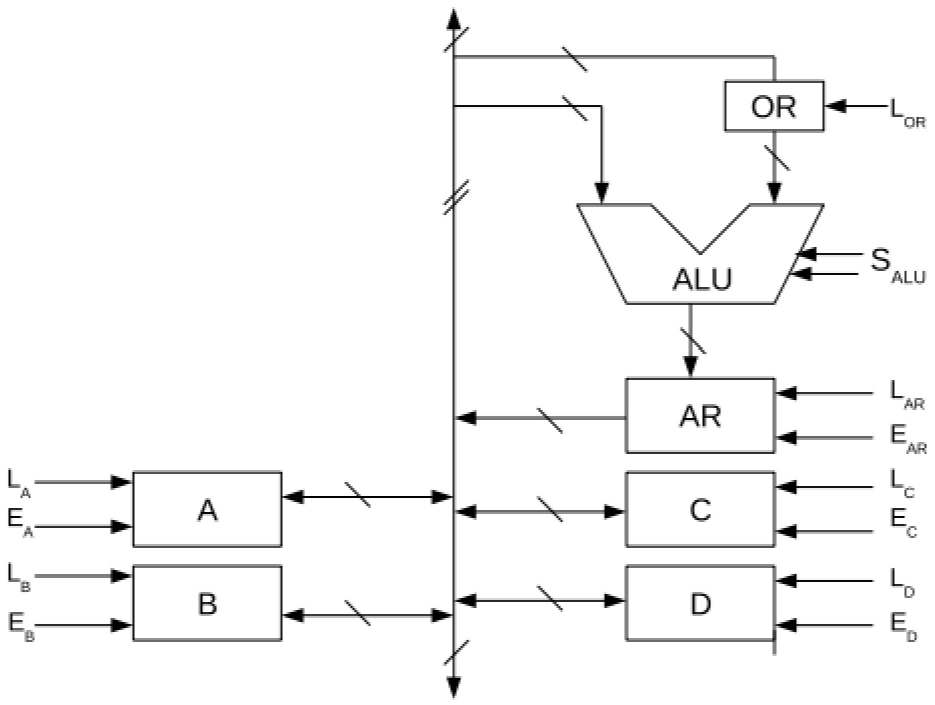
## Ans



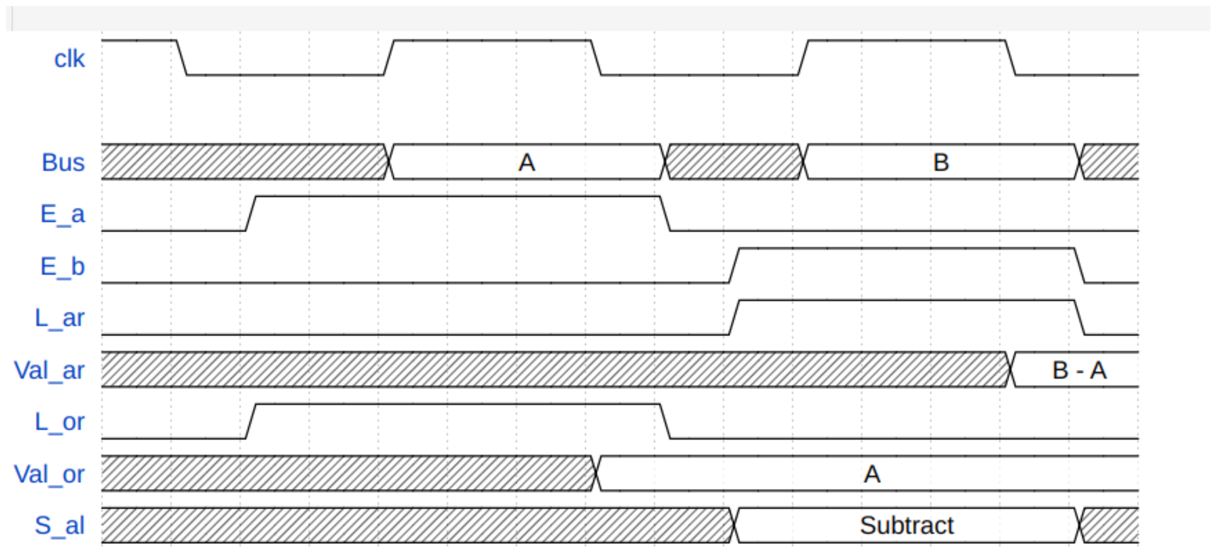
Clock Cycle 1: E_b, L_or
Clock Cycle 2: E_a, L_ar, S_al = SUBTRACTS

## Q4:

**Given the following ALU design, Give the timing diagram for performing the operation (B-A) and storing it in AR: (Dont use registers C and D)**



## Ans



Clock cycle 1: E_a, L_or
Clock cycle 2: E_b, L_ar, S_al = SUBTRACT

## Q5:

Assume that the registers $A$ and $B$ hold the values 10 and 11 initially. What is the value of $A$ and $B$ after the following clock cycles?

$$\begin{aligned} C_1 &: I_B \\ C_2 &: E_B, L_A, I_B \\ C_3 &: I_A \end{aligned}$$

Ans:

Explanation of each clock cycle:

- $C_1 : I_B$
  $B$ will increment to 12.
- $C_2 : E_B, L_A, I_B$
  $B$ increments to 13, while $A$ is loaded with the old $B$ value of 12.
- $C_3 : I_A$
  $A$ is incremented to 13.

Therefore, the final value of both $A$ and $B$ registers is 13.

## Q6:

**Custom Operation Sequence:** - Implement a three-step operation in a single-bus system where:

- The contents of $R3$ are moved to the operand register $OR$.
- The value in $OR$ is subtracted from $AR$, and the result is stored back in $AR$.
- Store the result from $AR$ into $R5$.

Write out the control signals needed for each clock cycle in this operation sequence

Ans:

$$\begin{aligned} &\text{Cycle 1:} \quad E_{RG}, L_{OR}, S_{RG} \leftarrow 3 \\ &\text{Cycle 2:} \quad E_{AR}, L_{AR}, S_{ALU} \leftarrow \text{SUB} \\ &\text{Cycle 3:} \quad E_{AR}, L_{RG}, S_{RG} \leftarrow 5 \end{aligned}$$

**Q7: Take a n-bit ALU. How does it implement the following set of operations with only the following components:**

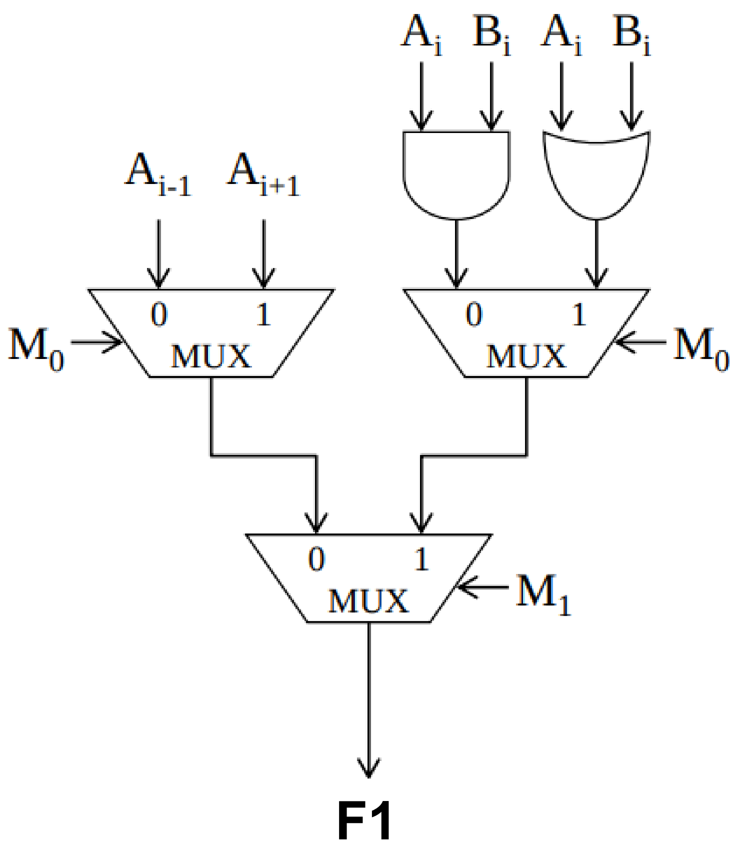
- **1 bit Full Adders**
- **2 input AND/OR/XOR gates**
- **Inverters**
- **2:1 MUX**

**Just draw the design for ith bit and assume that Ai-1, Ai+1, Bi-1,Bi+1 exist.**

a)

| M1 | M0 | Function Name | F = |
|---|---|---|---|
| 0 | 0 | Multiply by 2 | A*2 |
| 0 | 1 | Divide by 2 | A/2 |
| 1 | 0 | Bitwise AND | A AND B |
| 1 | 1 | Bitwise OR | A OR B |

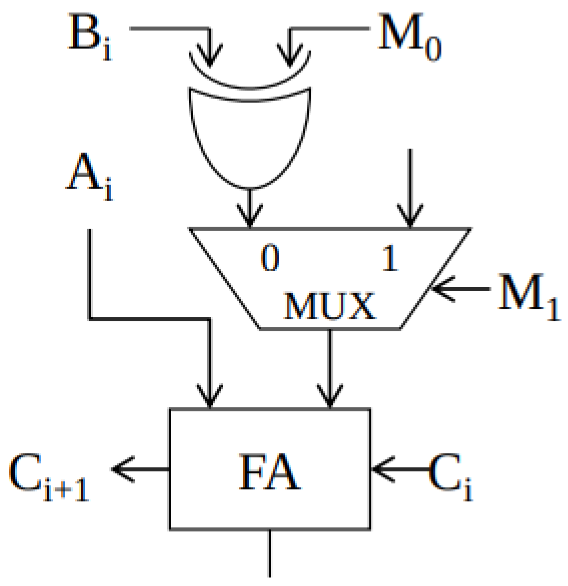
Ans)



**b)**

| M1 | M0 | Function Name | F = |
|---|---|---|---|
| 0 | 0 | ADD | A+B |
| 0 | 1 | SUBTRACT | A-B |
| 1 | 0 | INCREMENT | A+1 |
| 1 | 1 | DECREMENT | A-1 |

Ans) Let us take C0 to be M0 XOR M1



**F1**

**For decrement operation, In mathematical terms we're doing A – 0 – 1. Converted to an addition this becomes A + ~0 + 1 – 1. Simplifying that we get A + ~0 + 0. So we just need to set our carry in to zero.**

Q9. **In an accumulator-based architecture, the arithmetic unit relies heavily on the accumulator register to perform operations. If an 8-bit accumulator initially holds the value 0x2F (in hexadecimal), and it receives an instruction to add 0x3A to the accumulator's value, what will be the final value in the accumulator after the operation? Assume no carry-in from previous operations.**

Ans:

To solve this, we perform an 8-bit addition of the hexadecimal values in the accumulator.

Convert the hexadecimal values to binary or decimal for clarity:

0x2F in binary: 0010 1111
0x3A in binary: 0011 1010

Adding them will give 0110 1001 => 0x69

Thus, the final value in the accumulator is 0x69.

Q10. **In a single-bus architecture, data transfer between components takes place over a single shared bus, which can introduce delays due to the sequential nature of transfers. Consider a simple addition operation that follows these steps in a single-bus system:**

**Step 1: Load operand A from memory to a register.**
**Step 2: Load operand B from memory to a register.**
**Step 3: Perform addition in the arithmetic unit using operand A and operand B.**
**Step 4: Store the result back to memory.**
**If each data transfer over the bus (either loading from or storing to memory) takes 2 cycles and the arithmetic operation itself takes 1**

**cycle, calculate the total number of cycles required to complete this addition operation.**
Ans:

To determine the total cycles, add the cycles required for each step:

Step 1: Load operand A from memory to the register — 2 cycles.
Step 2: Load operand B from memory to the register — 2 cycles.
Step 3: Perform addition in the arithmetic unit — 1 cycle.
Step 4: Store the result back to memory — 2 cycles.

Total cycles required:
$2+2+1+2=7$ cycles

Answer: The addition operation requires a total of 7 cycles to complete.