# COS 284 TUTORIAL 4

CLASS TEST 3 RECAP

# PROVIDE THE MINIMAL **SUM-OF-PRODUCTS** FORM OF THE FUNCTION F REPRESENTED BY THIS KARNAUGH MAP

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| **00** | 1 | 0 | 1 | 1 |
| **01** | 0 | 0 | 0 | 1 |
| **11** | 1 | 1 | 1 | 1 |
| **10** | 0 | 0 | 0 | 0 |

# PROVIDE THE MINIMAL **SUM-OF-PRODUCTS** FORM OF THE FUNCTION F REPRESENTED BY THIS KARNAUGH MAP

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| **00** | 1 | 0 | 1 | 1 |
| **01** | 0 | 0 | 0 | 1 |
| **11** | 1 | 1 | 1 | 1 |
| **10** | 0 | 0 | 0 | 0 |

- find maximal groups of adjacent 1s where the number of 1s is a power of two
- all 1s must be covered by at least one group, overlapping of groups is allowed
- Each group forms a logical product
- The sum of the product of each group is the function represented by the map

# PROVIDE THE MINIMAL **SUM-OF-PRODUCTS** FORM OF THE FUNCTION F REPRESENTED BY THIS KARNAUGH MAP

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 0 | 1 | 1 |
| 01 | 0 | 0 | 0 | 1 |
| 11 | 1 | 1 | 1 | 1 |
| 10 | 0 | 0 | 0 | 0 |

AB

# PROVIDE THE MINIMAL **SUM-OF-PRODUCTS** FORM OF THE FUNCTION F REPRESENTED BY THIS KARNAUGH MAP

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 0 | 1 | 1 |
| 01 | 0 | 0 | 0 | 1 |
| 11 | 1 | 1 | 1 | 1 |
| 10 | 0 | 0 | 0 | 0 |

AB + A'B'D'

# PROVIDE THE MINIMAL **SUM-OF-PRODUCTS** FORM OF THE FUNCTION F REPRESENTED BY THIS KARNAUGH MAP

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 0 | 1 | 1 |
| 01 | 0 | 0 | 0 | 1 |
| 11 | 1 | 1 | 1 | 1 |
| 10 | 0 | 0 | 0 | 0 |

AB + A'B'D' + A'B'C

# PROVIDE THE MINIMAL **SUM-OF-PRODUCTS** FORM OF THE FUNCTION F REPRESENTED BY THIS KARNAUGH MAP

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 0 | 1 | 1 |
| 01 | 0 | 0 | 0 | 1 |
| 11 | 1 | 1 | 1 | 1 |
| 10 | 0 | 0 | 0 | 0 |

AB + A'B'D' + A'B'C + BCD'

# PROVIDE THE MINIMAL **PRODUCT-OF-SUMS** FORM OF THE FUNCTION F REPRESENTED BY THIS KARNAUGH MAP

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | 1 | 1 |
| 01 | 1 | 0 | 0 | 1 |
| 11 | 1 | 0 | 0 | 0 |
| 10 | 1 | 1 | 0 | 0 |

# PROVIDE THE MINIMAL **PRODUCT-OF-SUMS** FORM OF THE FUNCTION F REPRESENTED BY THIS KARNAUGH MAP

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | 1 | 1 |
| 01 | 1 | 0 | 0 | 1 |
| 11 | 1 | 0 | 0 | 0 |
| 10 | 1 | 1 | 0 | 0 |

A'B'

# PROVIDE THE MINIMAL **PRODUCT-OF-SUMS** FORM OF THE FUNCTION F REPRESENTED BY THIS KARNAUGH MAP

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | 1 | 1 |
| 01 | 1 | 0 | 0 | 1 |
| 11 | 1 | 0 | 0 | 0 |
| 10 | 1 | 1 | 0 | 0 |

A'B' + C'D'

# PROVIDE THE MINIMAL **PRODUCT-OF-SUMS** FORM OF THE FUNCTION F REPRESENTED BY THIS KARNAUGH MAP

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | 1 | 1 |
| 01 | 1 | 0 | 0 | 1 |
| 11 | 1 | 0 | 0 | 0 |
| 10 | 1 | 1 | 0 | 0 |

A'B' + C'D' + A'CD'

# PROVIDE THE MINIMAL **PRODUCT-OF-SUMS** FORM OF THE FUNCTION F REPRESENTED BY THIS KARNAUGH MAP

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | 1 | 1 |
| 01 | 1 | 0 | 0 | 1 |
| 11 | 1 | 0 | 0 | 0 |
| 10 | 1 | 1 | 0 | 0 |

A'B' + C'D' + A'CD' + AB'C'

# PROVIDE THE MINIMAL **PRODUCT-OF-SUMS** FORM OF THE FUNCTION F REPRESENTED BY THIS KARNAUGH MAP

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | 1 | 1 |
| 01 | 1 | 0 | 0 | 1 |
| 11 | 1 | 0 | 0 | 0 |
| 10 | 1 | 1 | 0 | 0 |

A'B' + C'D' + A'CD' + AB'C'

not a product of sums yet

# PROVIDE THE MINIMAL **PRODUCT-OF-SUMS** FORM OF THE FUNCTION F REPRESENTED BY THIS KARNAUGH MAP

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | 1 | 1 |
| 01 | 1 | 0 | 0 | 1 |
| 11 | 1 | 0 | 0 | 0 |
| 10 | 1 | 1 | 0 | 0 |

A'B' + C'D' + A'CD' + AB'C'          (apply distributive law x+yz = (x+y)(x+z))

# PROVIDE THE MINIMAL **PRODUCT-OF-SUMS** FORM OF THE FUNCTION F REPRESENTED BY THIS KARNAUGH MAP

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | 1 | 1 |
| 01 | 1 | 0 | 0 | 1 |
| 11 | 1 | 0 | 0 | 0 |
| 10 | 1 | 1 | 0 | 0 |

A'B' + C'D' + A'CD' + AB'C'  (apply distributive law x+yz = (x+y)(x+z))

= (A'B' + C')(A'B' + D')+ A'CD' + AB'C'

# PROVIDE THE MINIMAL **PRODUCT-OF-SUMS** FORM OF THE FUNCTION F REPRESENTED BY THIS KARNAUGH MAP

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | 1 | 1 |
| 01 | 1 | 0 | 0 | 1 |
| 11 | 1 | 0 | 0 | 0 |
| 10 | 1 | 1 | 0 | 0 |

$\underline{A'B' + C'D'} + A'CD' + AB'C'$ (apply distributive law $x+yz = (x+y)(x+z)$)

$= (A'B' + C')(A'B' + D') + A'CD' + AB'C'$

$= (A' + C')(B' + C')(A' + D')(B' + D') + A'CD' + AB'C' = \ldots$

# PROVIDE THE MINIMAL **PRODUCT-OF-SUMS** FORM OF THE FUNCTION F REPRESENTED BY THIS KARNAUGH MAP

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | 1 | 1 |
| 01 | 1 | 0 | 0 | 1 |
| 11 | 1 | 0 | 0 | 0 |
| 10 | 1 | 1 | 0 | 0 |

- a set of groups that cover all 1s allows us to derive the function F

# PROVIDE THE MINIMAL PRODUCT-OF-SUMS FORM OF THE FUNCTION F REPRESENTED BY THIS KARNAUGH MAP

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | 1 | 1 |
| 01 | 1 | 0 | 0 | 1 |
| 11 | 1 | 0 | 0 | 0 |
| 10 | 1 | 1 | 0 | 0 |

- a set of groups that cover all 1s allows us to derive the function F
- conversely, a set of groups that cover all 0s allows us to derive the function F’

# PROVIDE THE MINIMAL **PRODUCT-OF-SUMS** FORM OF THE FUNCTION F REPRESENTED BY THIS KARNAUGH MAP

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | 1 | 1 |
| 01 | 1 | 0 | 0 | 1 |
| 11 | 1 | 0 | 0 | 0 |
| 10 | 1 | 1 | 0 | 0 |

F' = BD + AC

# PROVIDE THE MINIMAL **PRODUCT-OF-SUMS** FORM OF THE FUNCTION F REPRESENTED BY THIS KARNAUGH MAP

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | 1 | 1 |
| 01 | 1 | 0 | 0 | 1 |
| 11 | 1 | 0 | 0 | 0 |
| 10 | 1 | 1 | 0 | 0 |

F’ = BD + AC

F = F’’ = (BD + AC)’

# PROVIDE THE MINIMAL **PRODUCT-OF-SUMS** FORM OF THE FUNCTION F REPRESENTED BY THIS KARNAUGH MAP

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | 1 | 1 |
| 01 | 1 | 0 | 0 | 1 |
| 11 | 1 | 0 | 0 | 0 |
| 10 | 1 | 1 | 0 | 0 |

F’ = BD + AC

F = F’’ = (BD + AC)’ = (BD)’(AC)’

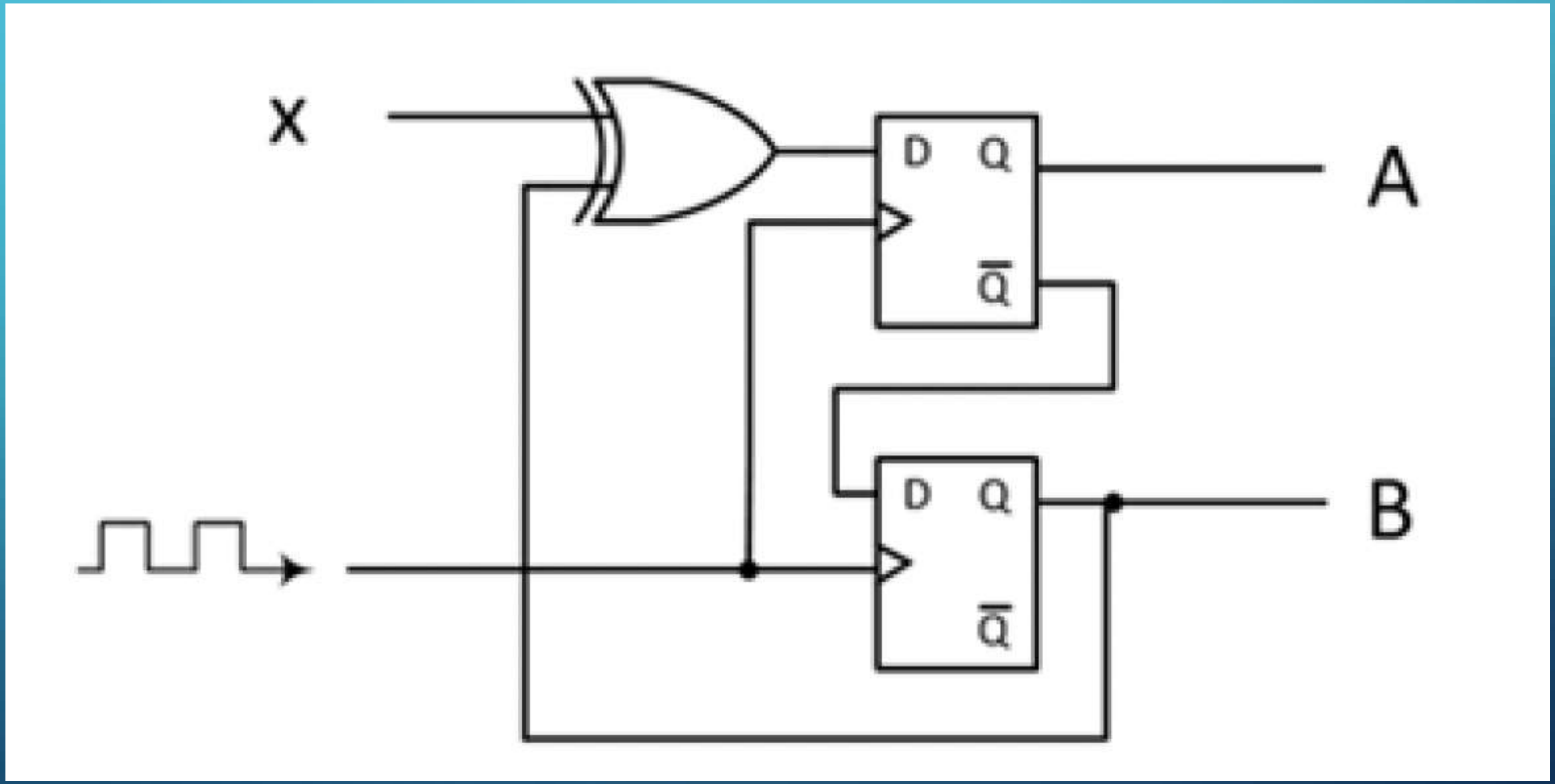
# PROVIDE THE MINIMAL **PRODUCT-OF-SUMS** FORM OF THE FUNCTION F REPRESENTED BY THIS KARNAUGH MAP

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| **00** | 1 | 1 | 1 | 1 |
| **01** | 1 | 0 | 0 | 1 |
| **11** | 1 | 0 | 0 | 0 |
| **10** | 1 | 1 | 0 | 0 |

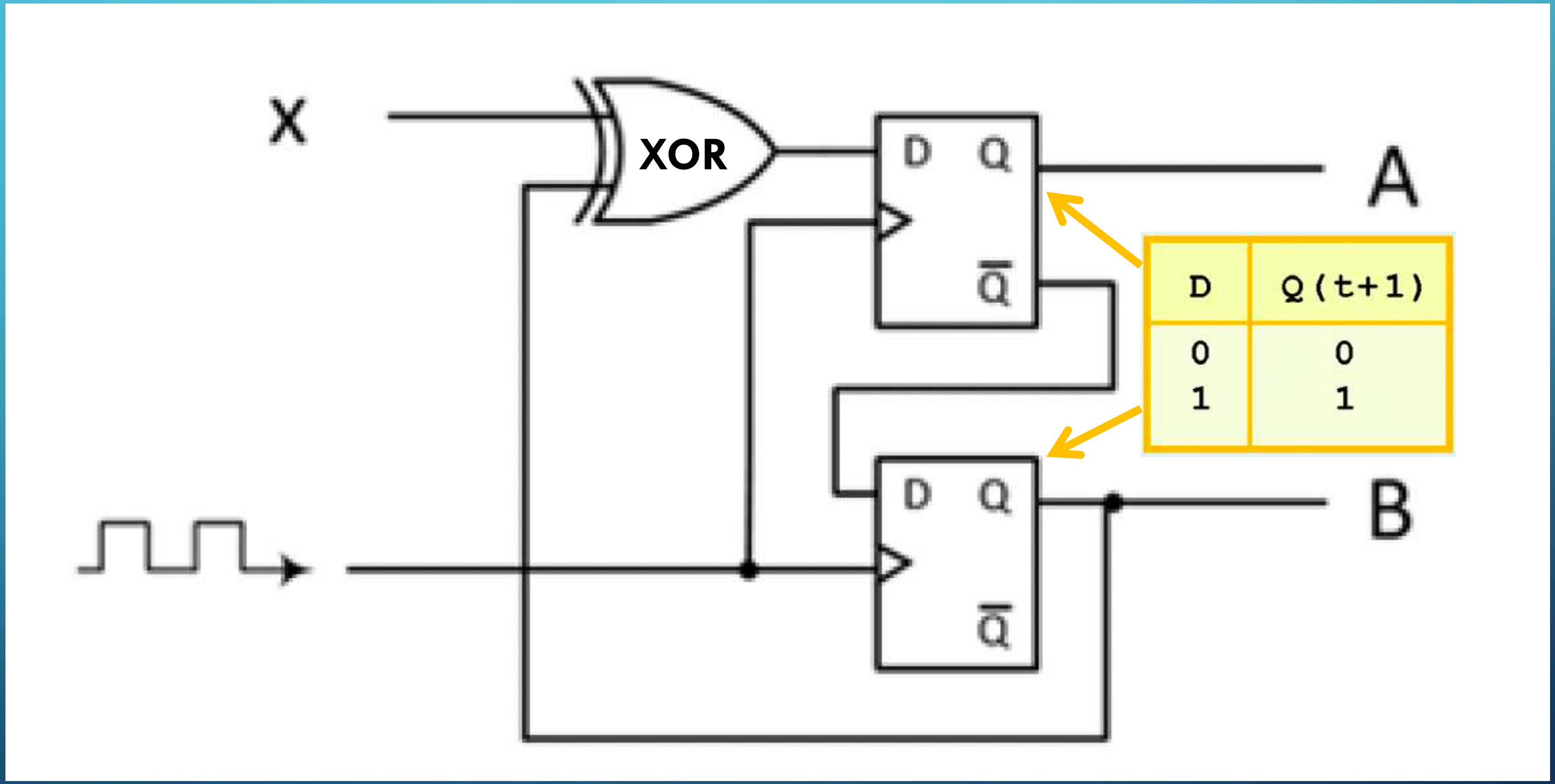
F’ = BD + AC

F = F’’ = (BD + AC)’ = (BD)’(AC)’ = (B’ + D’)(A’ + C’)

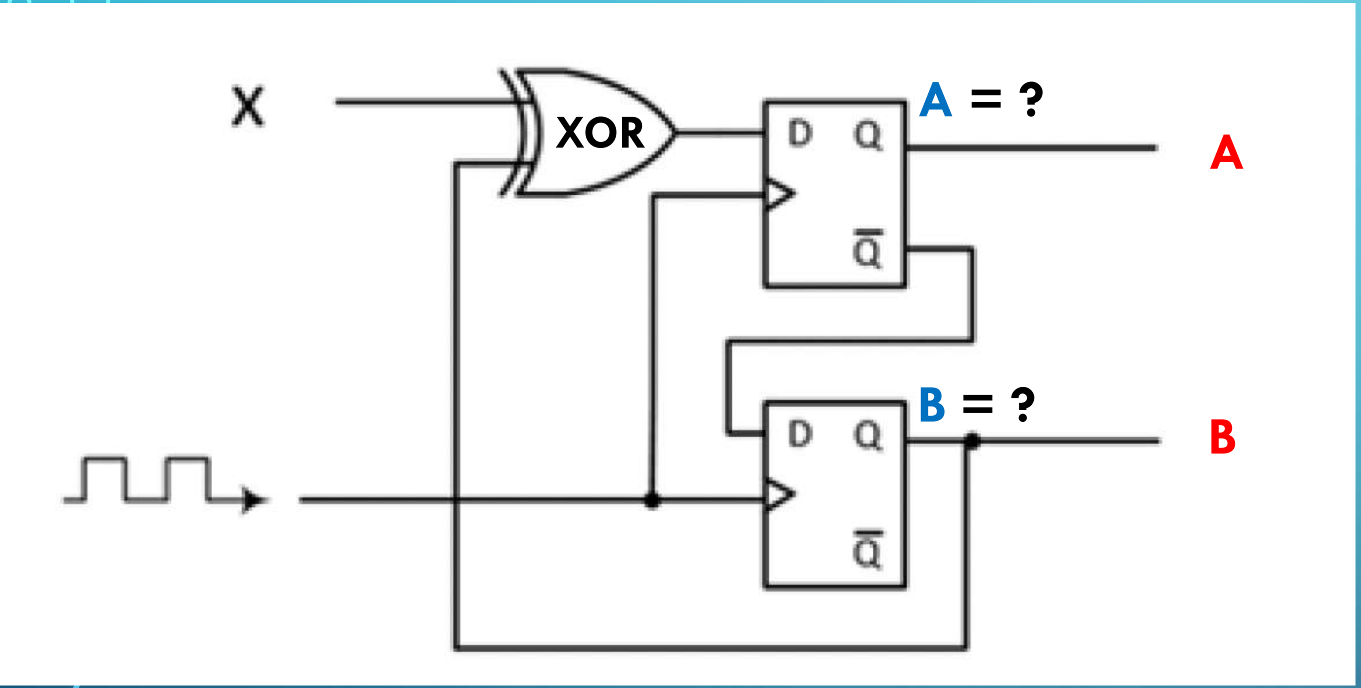
# COMPLETE THE CHARACTERISTIC TABLE FOR THE SEQUENTIAL CIRCUIT



# COMPLETE THE CHARACTERISTIC TABLE FOR THE SEQUENTIAL CIRCUIT

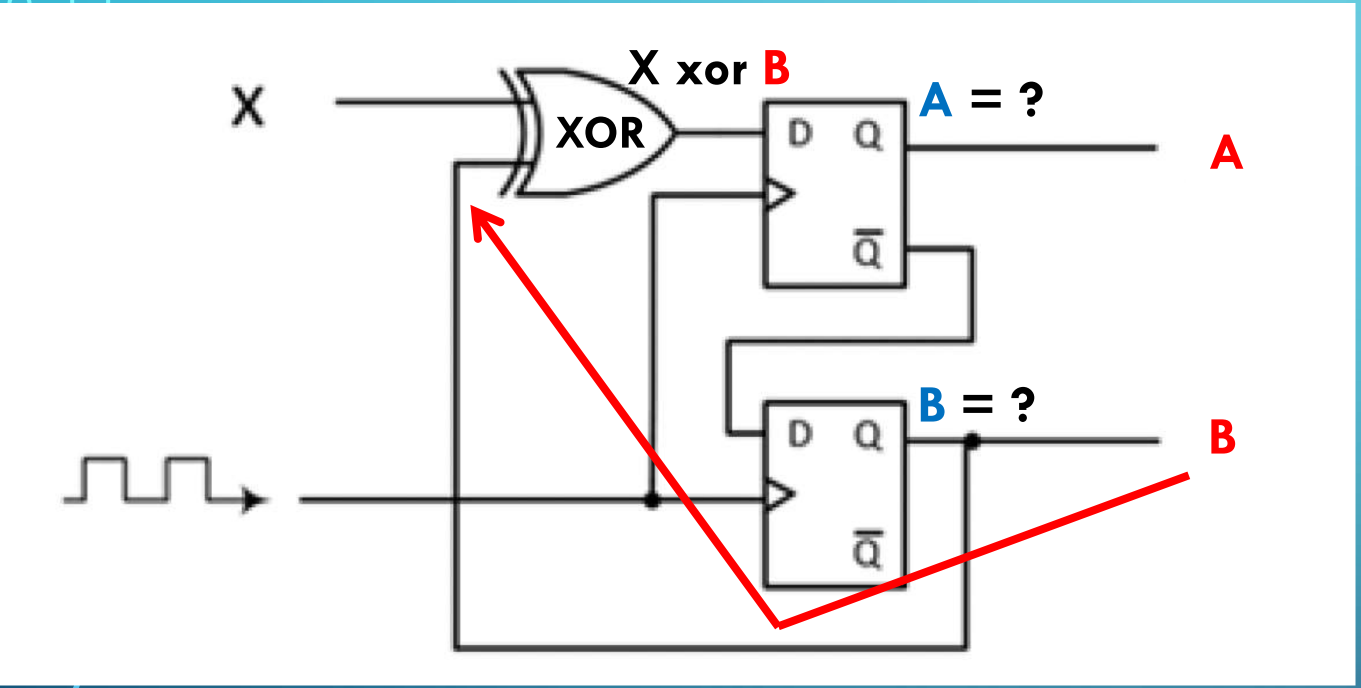


# COMPLETE THE CHARACTERISTIC TABLE FOR THE SEQUENTIAL CIRCUIT



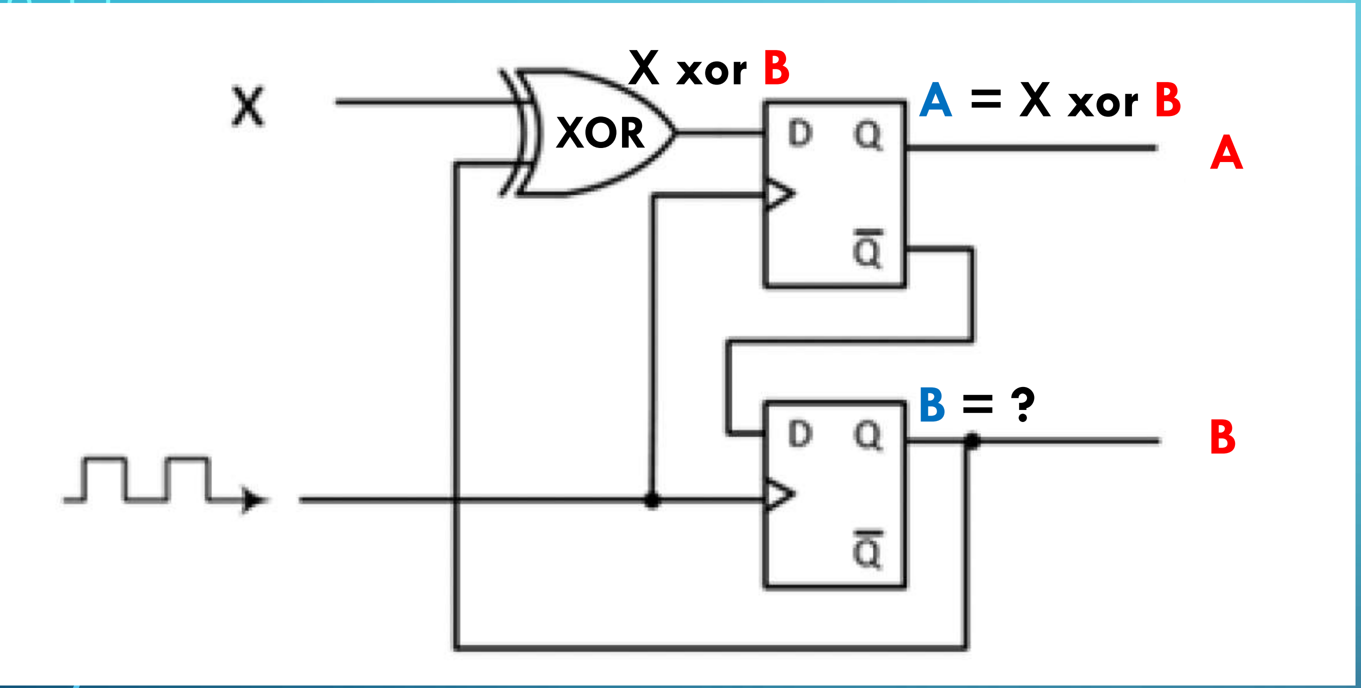

| X | A (Current) | B (Current) | A (Next) | B (Next) |
|---|---|---|---|---|
| 0 | 0 | 0 | | |
| 0 | 0 | 1 | | |
| 0 | 1 | 0 | | |
| 0 | 1 | 1 | | |
| 1 | 0 | 0 | | |
| 1 | 0 | 1 | | |
| 1 | 1 | 0 | | |
| 1 | 1 | 1 | | |

# COMPLETE THE CHARACTERISTIC TABLE FOR THE SEQUENTIAL CIRCUIT



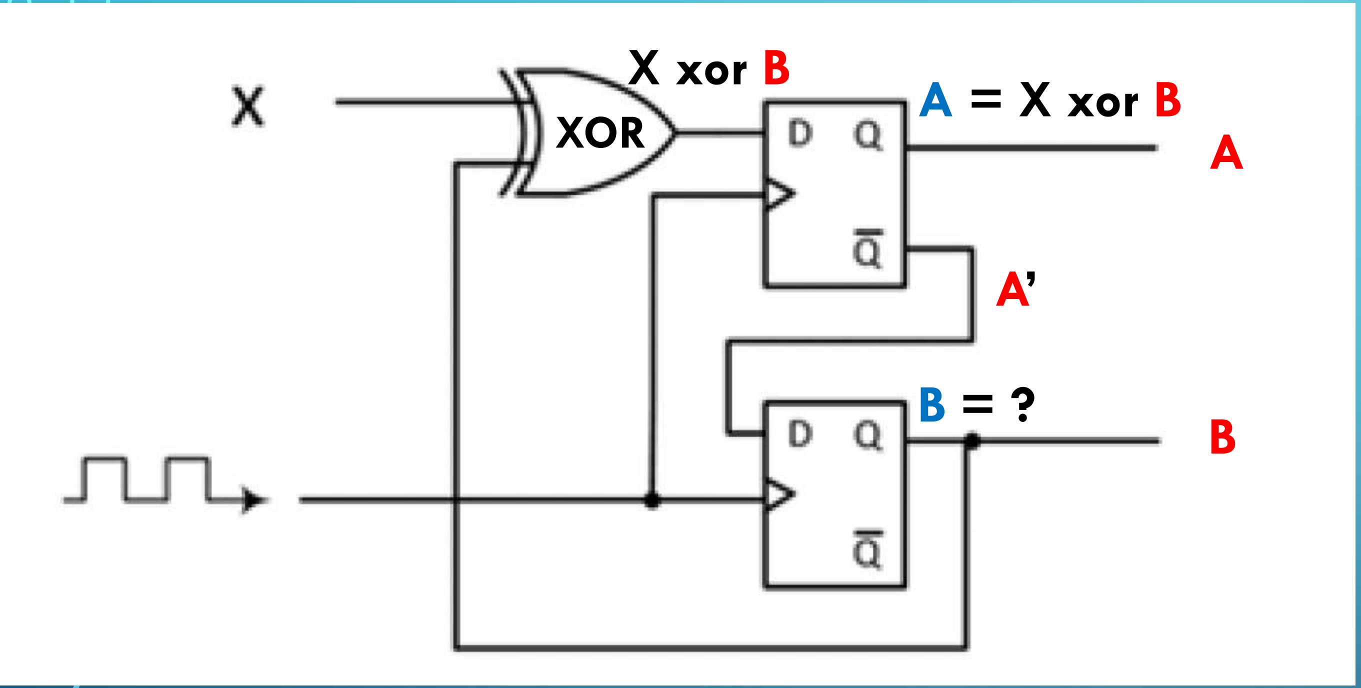

| X | A (Current) | B (Current) | A (Next) | B (Next) |
|---|---|---|---|---|
| 0 | 0 | 0 | | |
| 0 | 0 | 1 | | |
| 0 | 1 | 0 | | |
| 0 | 1 | 1 | | |
| 1 | 0 | 0 | | |
| 1 | 0 | 1 | | |
| 1 | 1 | 0 | | |
| 1 | 1 | 1 | | |

# COMPLETE THE CHARACTERISTIC TABLE FOR THE SEQUENTIAL CIRCUIT



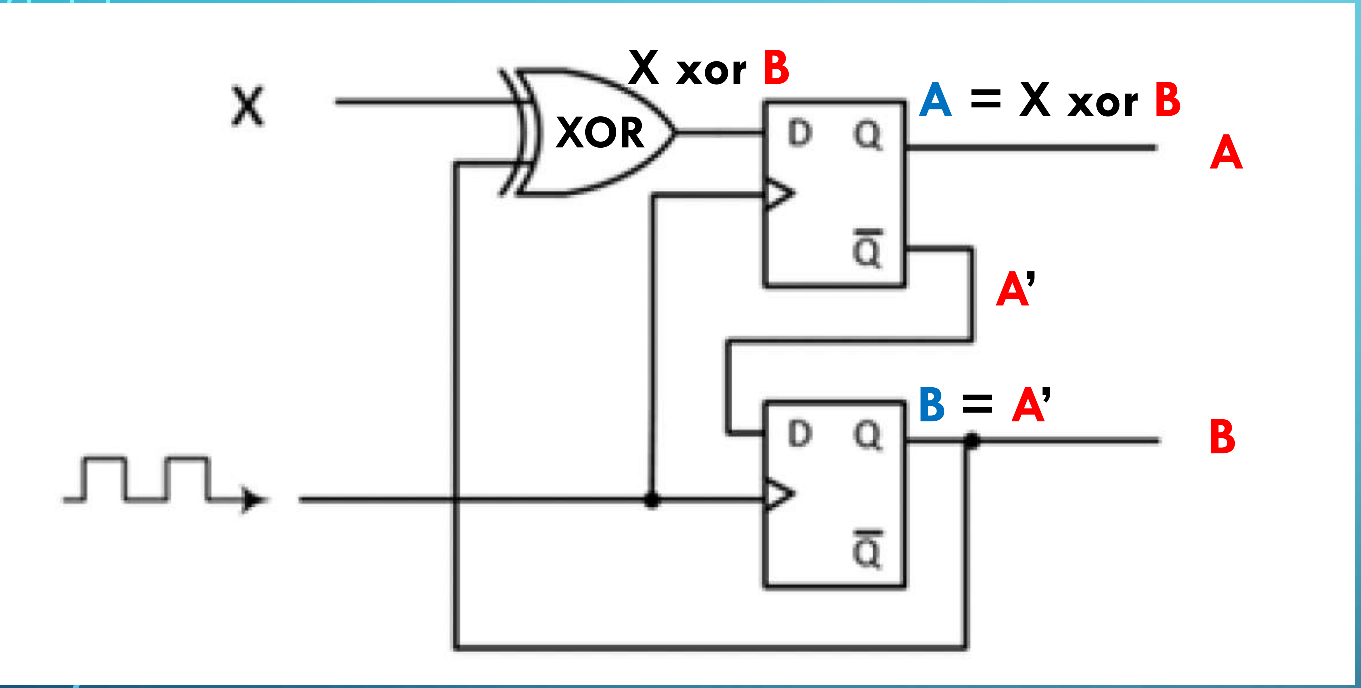

| X | A (Current) | B (Current) | A (Next) | B (Next) |
|---|---|---|---|---|
| 0 | 0 | 0 | | |
| 0 | 0 | 1 | | |
| 0 | 1 | 0 | | |
| 0 | 1 | 1 | | |
| 1 | 0 | 0 | | |
| 1 | 0 | 1 | | |
| 1 | 1 | 0 | | |
| 1 | 1 | 1 | | |

# COMPLETE THE CHARACTERISTIC TABLE FOR THE SEQUENTIAL CIRCUIT



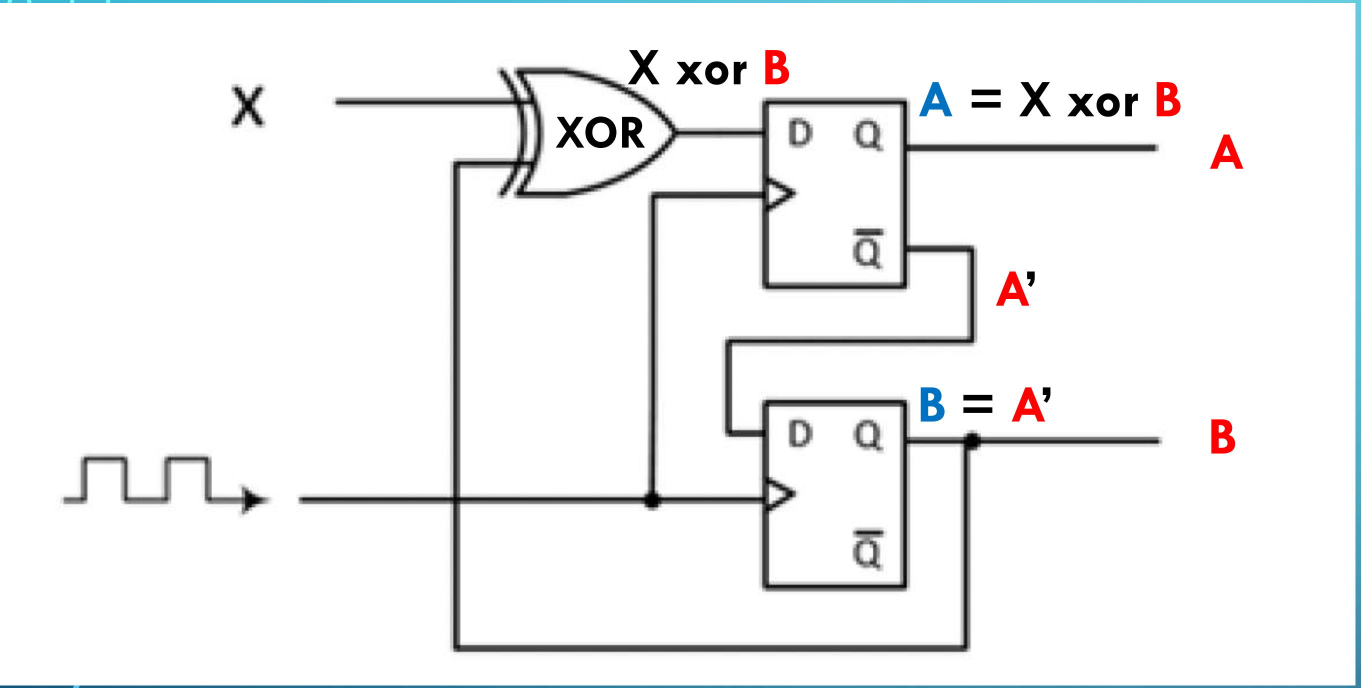

| X | A (Current) | B (Current) | A (Next) | B (Next) |
|---|---|---|---|---|
| 0 | 0 | 0 | | |
| 0 | 0 | 1 | | |
| 0 | 1 | 0 | | |
| 0 | 1 | 1 | | |
| 1 | 0 | 0 | | |
| 1 | 0 | 1 | | |
| 1 | 1 | 0 | | |
| 1 | 1 | 1 | | |

# COMPLETE THE CHARACTERISTIC TABLE FOR THE SEQUENTIAL CIRCUIT



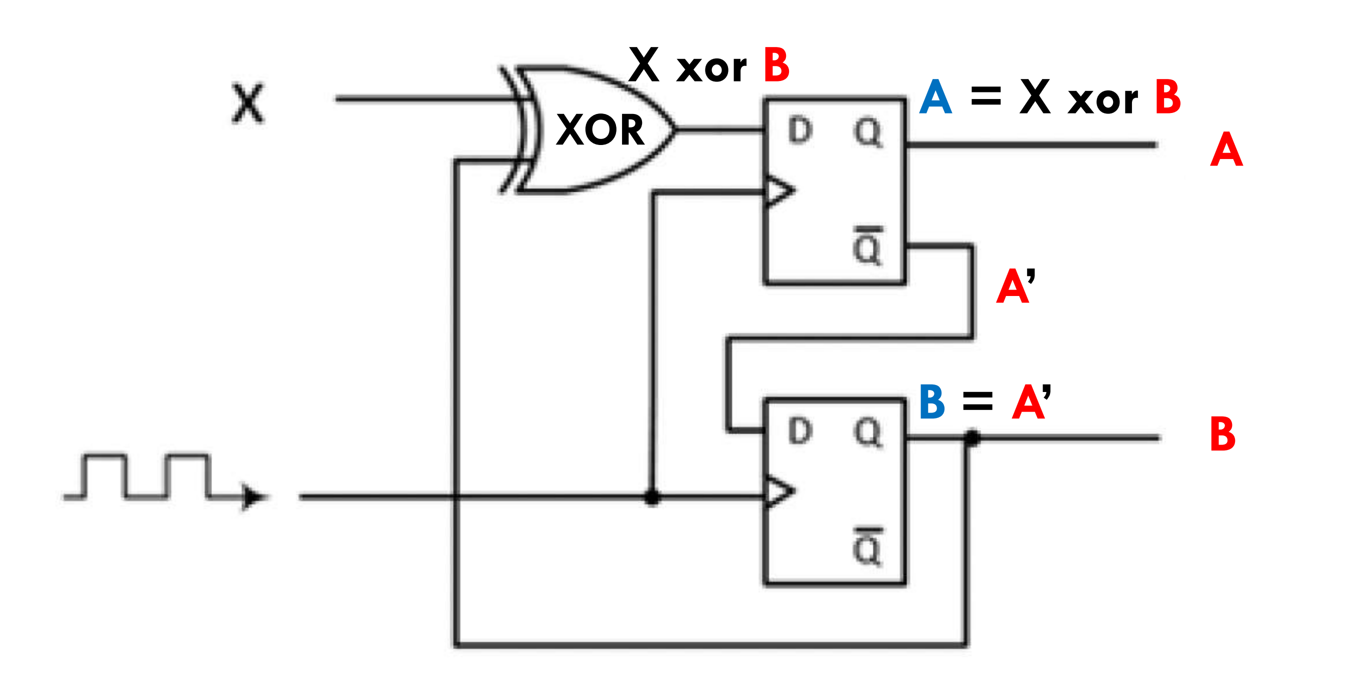

| X | A (Current) | B (Current) | A (Next) | B (Next) |
|---|---|---|---|---|
| 0 | 0 | 0 | | |
| 0 | 0 | 1 | | |
| 0 | 1 | 0 | | |
| 0 | 1 | 1 | | |
| 1 | 0 | 0 | | |
| 1 | 0 | 1 | | |
| 1 | 1 | 0 | | |
| 1 | 1 | 1 | | |

# COMPLETE THE CHARACTERISTIC TABLE FOR THE SEQUENTIAL CIRCUIT



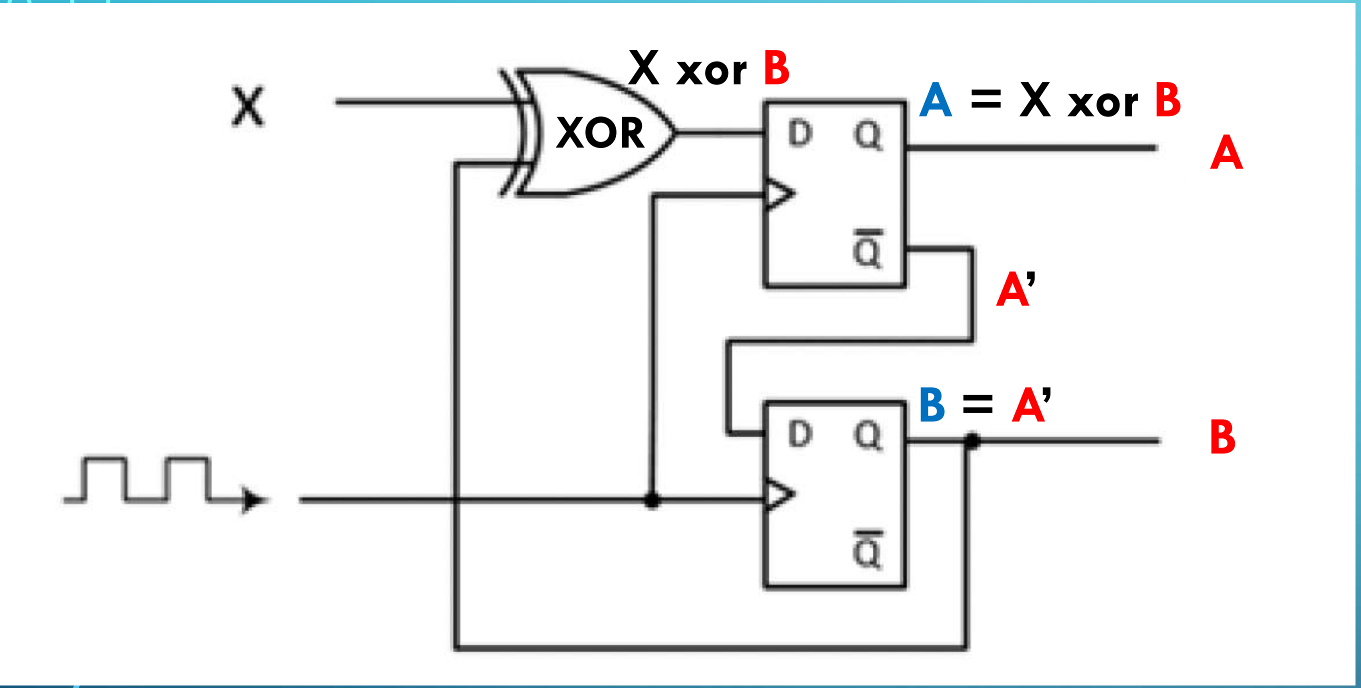

| X | A (Current) | B (Current) | A (Next) | B (Next) |
|---|---|---|---|---|
| 0 | 0 | 0 | X xor B | A' |
| 0 | 0 | 1 | X xor B | A' |
| 0 | 1 | 0 | X xor B | A' |
| 0 | 1 | 1 | X xor B | A' |
| 1 | 0 | 0 | X xor B | A' |
| 1 | 0 | 1 | X xor B | A' |
| 1 | 1 | 0 | X xor B | A' |
| 1 | 1 | 1 | X xor B | A' |

# COMPLETE THE CHARACTERISTIC TABLE FOR THE SEQUENTIAL CIRCUIT



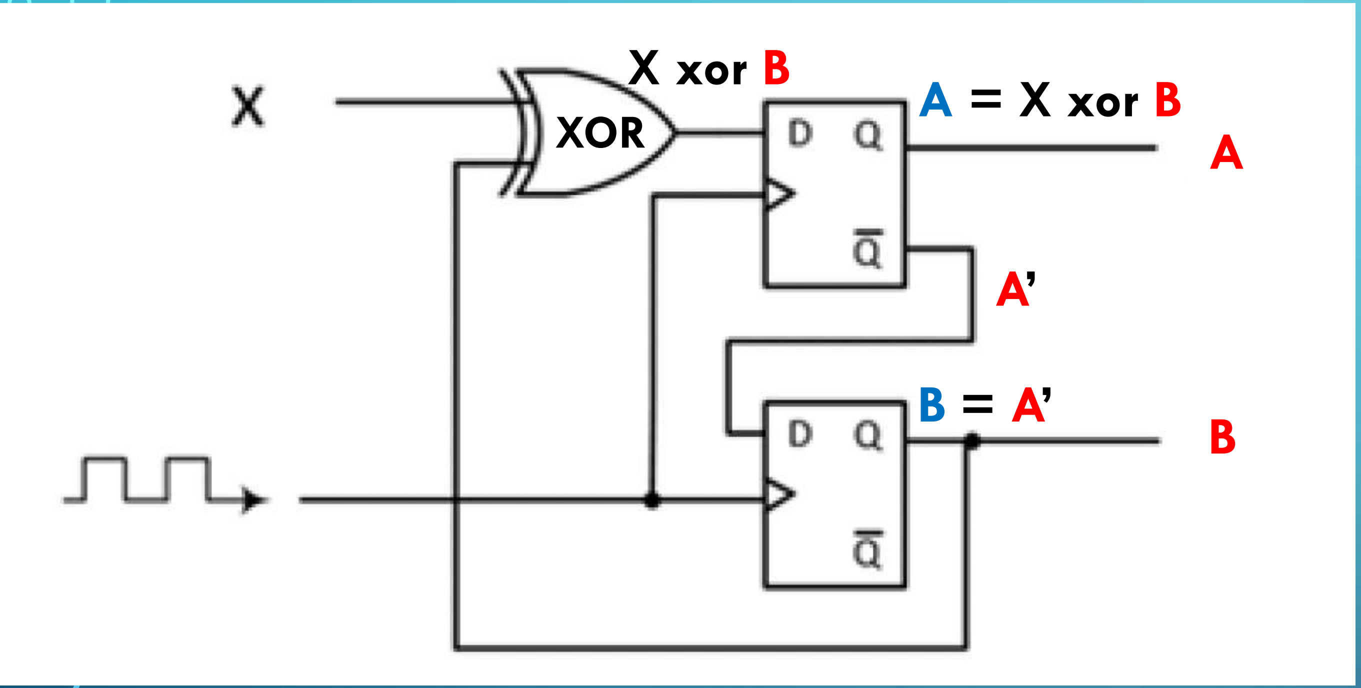

| X | A (Current) | B (Current) | A (Next) | B (Next) |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | X xor B | A' |
| 0 | 1 | 0 | X xor B | A' |
| 0 | 1 | 1 | X xor B | A' |
| 1 | 0 | 0 | X xor B | A' |
| 1 | 0 | 1 | X xor B | A' |
| 1 | 1 | 0 | X xor B | A' |
| 1 | 1 | 1 | X xor B | A' |

# COMPLETE THE CHARACTERISTIC TABLE FOR THE SEQUENTIAL CIRCUIT




| X | A (Current) | B (Current) | A (Next) | B (Next) |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | X xor B | A' |
| 0 | 1 | 1 | X xor B | A' |
| 1 | 0 | 0 | X xor B | A' |
| 1 | 0 | 1 | X xor B | A' |
| 1 | 1 | 0 | X xor B | A' |
| 1 | 1 | 1 | X xor B | A' |

# COMPLETE THE CHARACTERISTIC TABLE FOR THE SEQUENTIAL CIRCUIT




| X | A (Current) | B (Current) | A (Next) | B (Next) |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |

# MARIE PROGRAM: THE USER ENTERS A NATURAL NUMBER X, THE PROGRAM DECIDES IF THE NUMBER IS EVEN OR ODD BY CONTINUALLY SUBTRACTING 1 FROM IT.

# MARIE PROGRAM: THE USER ENTERS A NATURAL NUMBER X, THE PROGRAM DECIDES IF THE NUMBER IS EVEN OR ODD BY CONTINUALLY SUBTRACTING 1 FROM IT.

**Pseudocode Solution:**

```
input x
loop forever do
        if x == 0 then
                return 0 (Even)
        else
                x--
                if x == 0 then
                        return 1 (Odd)
                else
                        x--
```

# MARIE PROGRAM: THE USER ENTERS A NATURAL NUMBER X, THE PROGRAM DECIDES IF THE NUMBER IS EVEN OR ODD BY CONTINUALLY SUBTRACTING 1 FROM IT.

**Pseudocode Solution:**

```
input x
loop forever do
      if x == 0 then
            return 0 (Even)
      else
            x--
            if x == 0 then
                  return 1 (Odd)
            else
                  x--
```

| Instruction | Meaning |
|---|---|
| Load X | Load contents of address X into AC. |
| Store X | Store the contents of AC at address X. |
| Add X | Add the contents of address X to AC. |
| Subt X | Subtract the contents of address X from AC. |
| Input | Input a value from the keyboard into AC. |
| Output | Output the value in AC to the display. |
| Halt | Terminate program. |
| Skipcond 800 | Skip next instruction if AC is positive |
| Jump X | Load the value of X into PC. |

# MARIE PROGRAM: THE USER ENTERS A NATURAL NUMBER X, THE PROGRAM DECIDES IF THE NUMBER IS EVEN OR ODD BY CONTINUALLY SUBTRACTING 1 FROM IT.

**Pseudocode Solution:**

```
input x
loop forever do
    if x == 0 then
        return 0
    else
        x--
        if x == 0 then
            return 1
        else
            x--
```

| Instruction | Meaning |
|---|---|
| Load X | Load contents of address X into AC. |
| Store X | Store the contents of AC at address X. |
| Add X | Add the contents of address X to AC. |
| Subt X | Subtract the contents of address X from AC. |
| Input | Input a value from the keyboard into AC. |
| Output | Output the value in AC to the display. |
| Halt | Terminate program. |
| Skipcond 800 | Skip next instruction if AC is positive |
| Jump X | Load the value of X into PC. |

| Label | Instruction |
|---|---|
| Read | Input |
| Loop | |
| - | |
| - | |
| - | |
| - | |
| - | |
| - | |
| Even | |
| - | |
| - | Halt |
| Odd | |
| - | |
| - | Halt |
| One | Dec 1 |
| Zero | Dec 0 |

# MARIE PROGRAM: THE USER ENTERS A NATURAL NUMBER X, THE PROGRAM DECIDES IF THE NUMBER IS EVEN OR ODD BY CONTINUALLY SUBTRACTING 1 FROM IT.

**Pseudocode Solution:**

```
input x
loop forever do
    if x == 0 then
        return 0
    else
        x--
        if x == 0 then
            return 1
        else
            x--
```

| Instruction | Meaning |
|---|---|
| Load X | Load contents of address X into AC. |
| Store X | Store the contents of AC at address X. |
| Add X | Add the contents of address X to AC. |
| Subt X | Subtract the contents of address X from AC. |
| Input | Input a value from the keyboard into AC. |
| Output | Output the value in AC to the display. |
| Halt | Terminate program. |
| Skipcond 800 | Skip next instruction if AC is positive |
| Jump X | Load the value of X into PC. |

| Label | Instruction |
|---|---|
| Read | Input |
| Loop | Skipcond 800 |
| - | Jump Even |
| - | Subt One |
| - | |
| - | |
| - | |
| - | |
| Even | |
| - | |
| - | Halt |
| Odd | |
| - | |
| - | Halt |
| One | Dec 1 |
| Zero | Dec 0 |

# MARIE PROGRAM: THE USER ENTERS A NATURAL NUMBER X, THE PROGRAM DECIDES IF THE NUMBER IS EVEN OR ODD BY CONTINUALLY SUBTRACTING 1 FROM IT.

**Pseudocode Solution:**

```
input x
loop forever do
    if x == 0 then
        return 0
    else
        x--
        if x == 0 then
            return 1
        else
            x--
```

| Instruction | Meaning |
|---|---|
| Load X | Load contents of address X into AC. |
| Store X | Store the contents of AC at address X. |
| Add X | Add the contents of address X to AC. |
| Subt X | Subtract the contents of address X from AC. |
| Input | Input a value from the keyboard into AC. |
| Output | Output the value in AC to the display. |
| Halt | Terminate program. |
| Skipcond 800 | Skip next instruction if AC is positive |
| Jump X | Load the value of X into PC. |

| Label | Instruction |
|---|---|
| Read | Input |
| Loop | Skipcond 800 |
| - | Jump Even |
| - | Subt One |
| - | |
| - | |
| - | |
| - | |
| Even | Load Zero |
| - | Output |
| - | Halt |
| Odd | |
| - | |
| - | Halt |
| One | Dec 1 |
| Zero | Dec 0 |

# MARIE PROGRAM: THE USER ENTERS A NATURAL NUMBER X, THE PROGRAM DECIDES IF THE NUMBER IS EVEN OR ODD BY CONTINUALLY SUBTRACTING 1 FROM IT.

**Pseudocode Solution:**

```
input x
loop forever do
    if x == 0 then
        return 0
    else
        x--
        if x == 0 then
            return 1
        else
            x--
```

| Instruction | Meaning |
|---|---|
| Load X | Load contents of address X into AC. |
| Store X | Store the contents of AC at address X. |
| Add X | Add the contents of address X to AC. |
| Subt X | Subtract the contents of address X from AC |
| Input | Input a value from the keyboard into AC. |
| Output | Output the value in AC to the display. |
| Halt | Terminate program. |
| Skipcond 800 | Skip next instruction if AC is positive |
| Jump X | Load the value of X into PC. |

| Label | Instruction |
|---|---|
| Read | Input |
| Loop | Skipcond 800 |
| - | Jump Even |
| - | Subt One |
| - | Skipcond 800 |
| - | Jump Odd |
| - | Subt One |
| - | |
| Even | Load Zero |
| - | Output |
| - | Halt |
| Odd | |
| - | |
| - | Halt |
| One | Dec 1 |
| Zero | Dec 0 |

# MARIE PROGRAM: THE USER ENTERS A NATURAL NUMBER X, THE PROGRAM DECIDES IF THE NUMBER IS EVEN OR ODD BY CONTINUALLY SUBTRACTING 1 FROM IT.

**Pseudocode Solution:**

```
input x
loop forever do
    if x == 0 then
        return 0
    else
        x--
        if x == 0 then
            return 1
        else
            x--
```

| Instruction | Meaning |
|---|---|
| Load X | Load contents of address X into AC. |
| Store X | Store the contents of AC at address X. |
| Add X | Add the contents of address X to AC. |
| Subt X | Subtract the contents of address X from AC |
| Input | Input a value from the keyboard into AC. |
| Output | Output the value in AC to the display. |
| Halt | Terminate program. |
| Skipcond 800 | Skip next instruction if AC is positive |
| Jump X | Load the value of X into PC. |

| Label | Instruction |
|---|---|
| Read | Input |
| Loop | Skipcond 800 |
| - | Jump Even |
| - | Subt One |
| - | Skipcond 800 |
| - | Jump Odd |
| - | Subt One |
| - | |
| Even | Load Zero |
| - | Output |
| - | Halt |
| Odd | Load One |
| - | Output |
| - | Halt |
| One | Dec 1 |
| Zero | Dec 0 |

# MARIE PROGRAM: THE USER ENTERS A NATURAL NUMBER X, THE PROGRAM DECIDES IF THE NUMBER IS EVEN OR ODD BY CONTINUALLY SUBTRACTING 1 FROM IT.

**Pseudocode Solution:**

```
input x
loop forever do
    if x == 0 then
        return 0
    else
        x--
        if x == 0 then
            return 1
        else
            x--
```

| Instruction | Meaning |
|---|---|
| Load X | Load contents of address X into AC. |
| Store X | Store the contents of AC at address X. |
| Add X | Add the contents of address X to AC. |
| Subt X | Subtract the contents of address X from AC. |
| Input | Input a value from the keyboard into AC. |
| Output | Output the value in AC to the display. |
| Halt | Terminate program. |
| Skipcond 800 | Skip next instruction if AC is positive |
| Jump X | Load the value of X into PC. |

| Label | Instruction |
|---|---|
| Read | Input |
| Loop | Skipcond 800 |
| - | Jump Even |
| - | Subt One |
| - | Skipcond 800 |
| - | Jump Odd |
| - | Subt One |
| - | Jump Loop |
| Even | Load Zero |
| - | Output |
| - | Halt |
| Odd | Load One |
| - | Output |
| - | Halt |
| One | Dec 1 |
| Zero | Dec 0 |