# Sequential Design: Example 2

- **Design a sequential modulo 3 accumulator for 2-bit operands**
- **Definitions:**
  - **Modulo *n* adder - an adder that gives the result of the addition as the remainder of the sum divided by *n***
    - **Example: (2 + 2) modulo 3 = remainder of 4/3 = 1**
  - **Accumulator - a circuit that "accumulates" the sum of its input operands over time - it adds each input operand to the stored sum, which is initially 0.**
- **Stored sum: ($Y_1$,$Y_0$), Input: ($X_1$,$X_0$), Output: ($Z_1$,$Z_0$)**

# Modulo 3 Accumulator

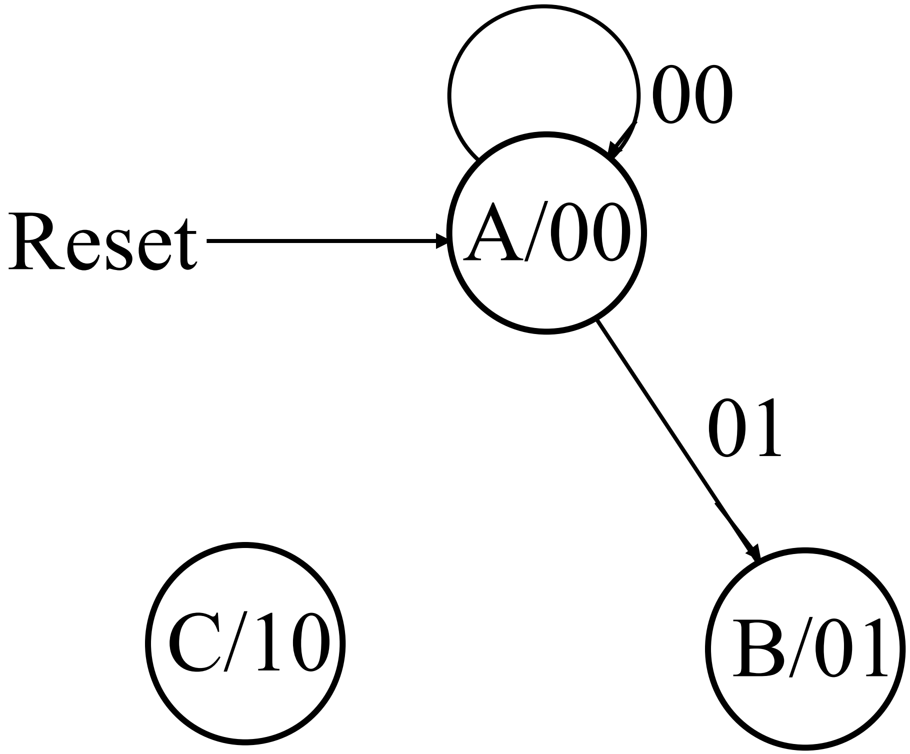
- **For example, originally, Y1Y0=00 (assume this state is called A/00) and the first input is X1X0=10, then ( 10 + 00 ) mod 3 = 10. Therefore, state A/00 will transfer to state C/10. If the first input is X1X0=01, state A/00 will transfer to B/01 because the remainder of (stored sum + input) mod 3 = 01.**
- **Why we don’t need input X1X0=11 ? Because whenever the input is 11, no state change!! ( something + 11) mod 3 = something itself!**

# Analysis

- **Assume that X1X0 is input (a two-bit binary number), Z1Z0 is output, Y1Y0 is stored sum and it is used to store the result of [(X1X0 + Y1Y0) mod 3].**
- **Originally, Y1Y0=00. Also, Z1Z0=Y1Y0 (now you understand that the output Z1Z0 only depends on the state Y1Y0 and therefore this is a Moore model sequential circuit).**
- **We're using D Flip-Flops for storing Y1Y0. For example, originally, Y1Y0=00 (assume this state is called A/00) and the first input is X1X0=01, then (01 + 00) mod 3 = 01. Therefore, state A/00 will transfer to state B/01. Since there are only 3 possibilities for state Y1Y0 (A/00, B/01, C/10), we only need 3 states to describe the behavior of this sequential circuit.**

# Example 2 (continued)

- **Complete the state diagram:**



# Example 2 (continued)

- **Complete the state table**

| $X_1X_0$ / $Y_1Y_0$ | 00 | 01 | 11 | 10 | $Z_1Z_0$ |
|---|---|---|---|---|---|
| | $Y_1(t+1)$, $Y_0(t+1)$ | $Y_1(t+1)$, $Y_0(t+1)$ | $Y_1(t+1)$, $Y_0(t+1)$ | $Y_1(t+1)$, $Y_0(t+1)$ | |
| A (00) | 00 | 01 | X | 10 | 00 |
| B (01) | 01 | 10 | X | 00 | 01 |
| - (11) | X | X | X | X | 11 |
| C (10) | 10 | 00 | X | 01 | 10 |

- **State Assignment: $(Y_1,Y_0) = (Z_1,Z_0)$**
- **Codes are in gray code order to ease use of K-maps in the next step**

# Example 2 (continued)

- **Find optimized flip-flop input equations for D flip-flops**

**$D_1$** (columns: $X_1$ spans columns 3–4, $X_0$ spans columns 2–3; rows: $Y_0$ spans rows 2–3, $Y_1$ spans rows 3–4)

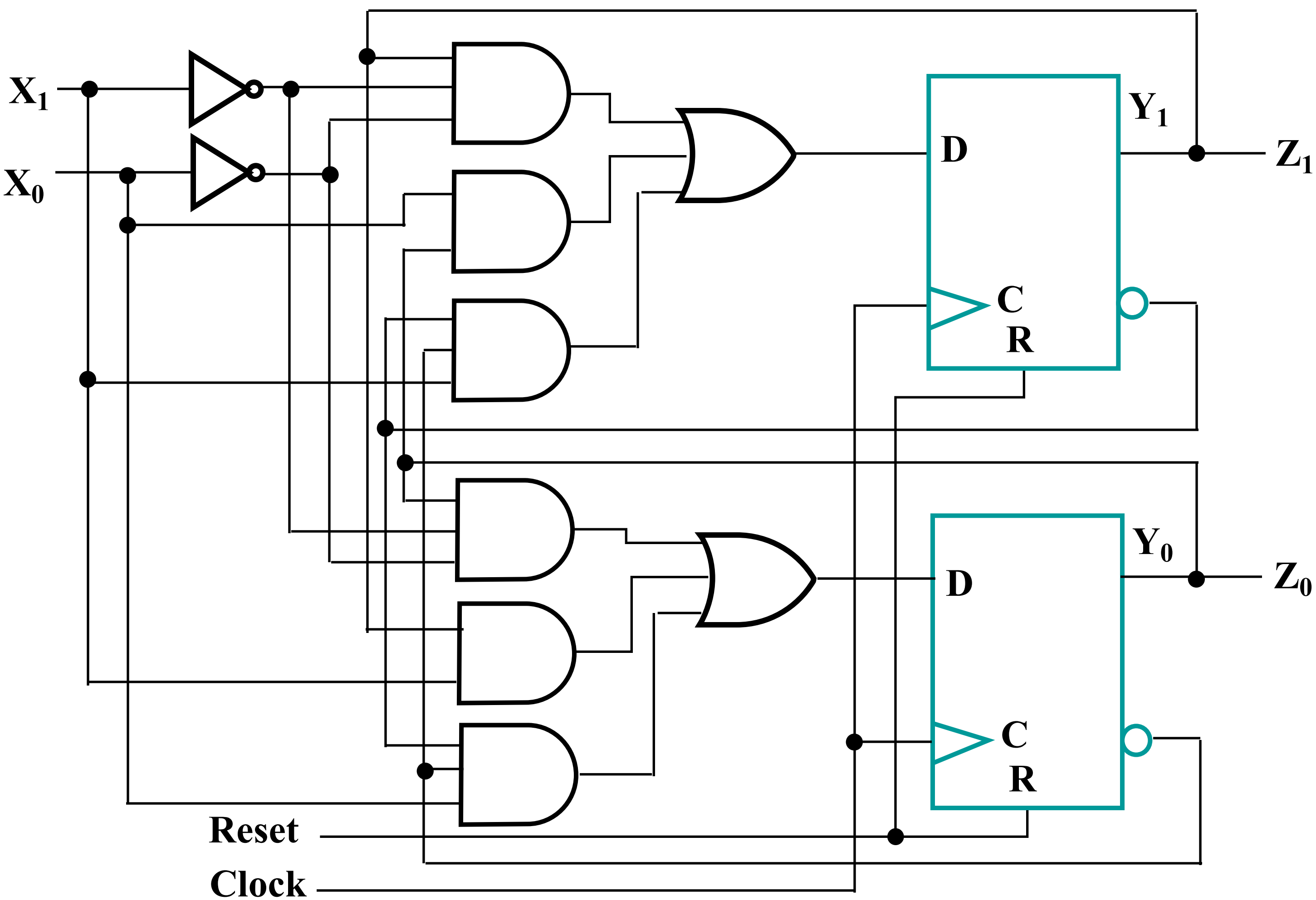
| | | | |
|---|---|---|---|
| | | X | 1 |
| | 1 | X | |
| X | X | X | X |
| 1 | | X | |

**$D_0$** (columns: $X_1$ spans columns 3–4, $X_0$ spans columns 2–3; rows: $Y_0$ spans rows 2–3, $Y_1$ spans rows 3–4)

| | | | |
|---|---|---|---|
| | 1 | X | |
| 1 | | X | |
| X | X | X | X |
| | | X | 1 |

- **$D_1$** $= Y_1X_1'X_0' + Y_0X_0 + Y_1'Y_0'X_1$
- **$D_0$** $= Y_0X_1'X_0' + Y_1X_1 + Y_1'Y_0'X_0$

# Circuit - Final Result with AND, OR, NOT



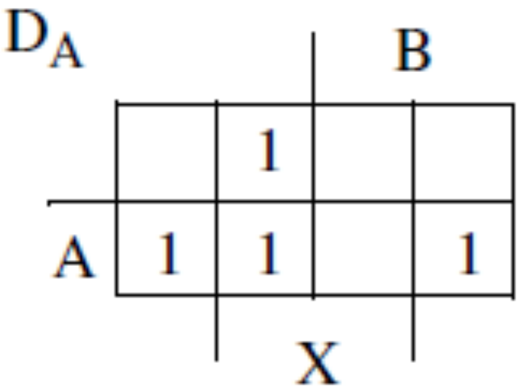
# Sequential Design: Example 3 (Q4-13)

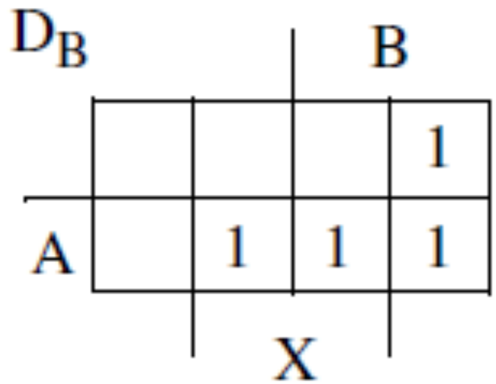
- **Design a sequential circuit with two D flip-flops A and B and one input X. When X=0, the state of the circuit remains the same. When X=1, the circuit goes through the state transitions from 00 to 10 to 11 to 01, back to 00, and then repeats.**

# Draw the state table

| Present state | | Input | Next state | |
|---|---|---|---|---|
| *A* | *B* | *X* | *A* | *B* |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |

# Derive equations using K-map



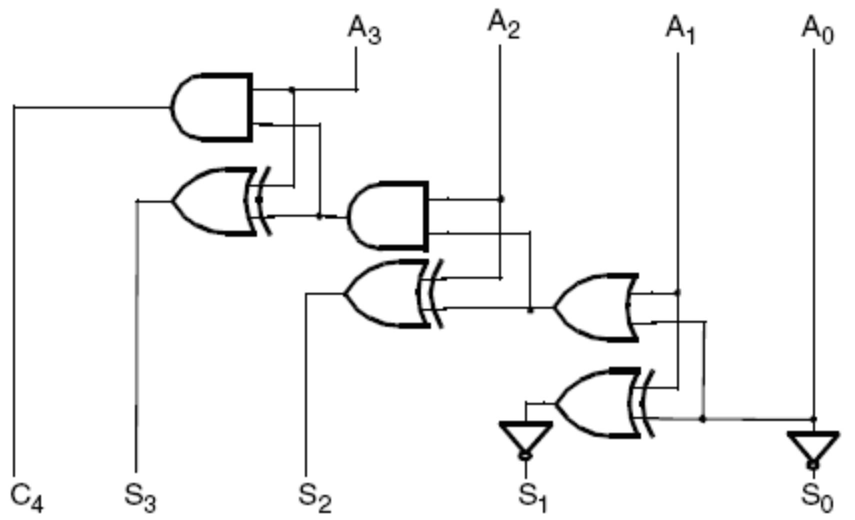




$D_A = A\overline{X} + \overline{B}X$

$D_B = AX + B\overline{X}$

# Hints Q 3-57

## Look at the slides 29~30 in Lecture 12



# Hints of Q 4-17

- **Moore model and you will have 5 states in order to remember the four specified inputs (your secret code for the luggage).**

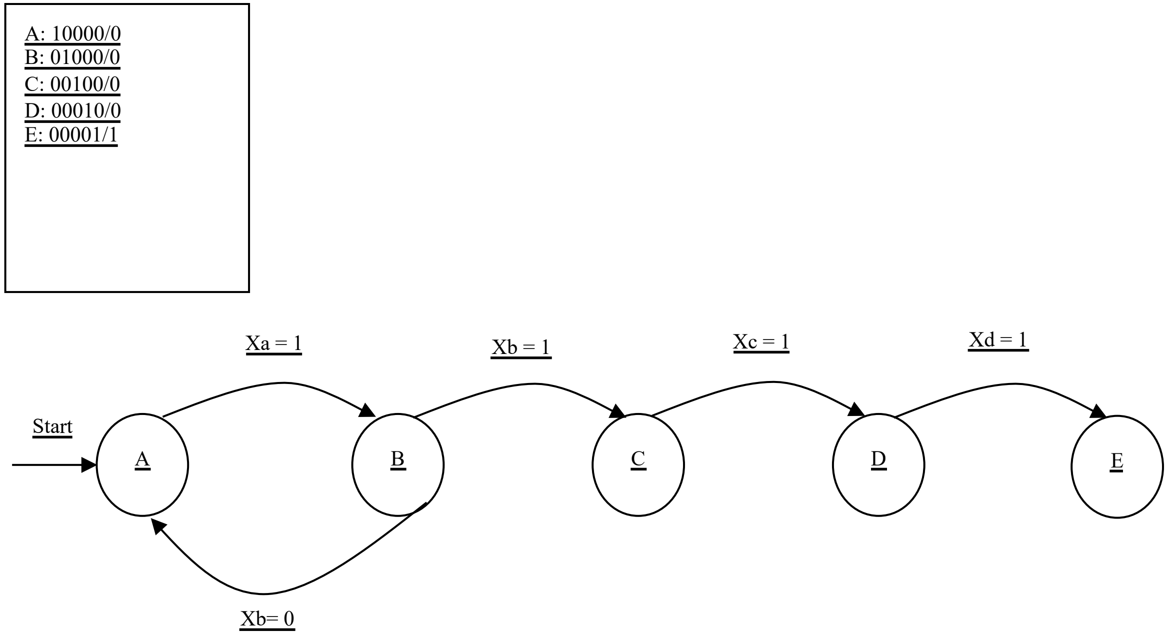
- **1-hot code refers to a group of bits among which the legal combinations of values are only those with a single high (1) bit and all the others low (0). For example, the output of a decoder is usually a one-hot code, and sometimes the state of a state machine is represented by a one-hot code.**

- **Since you're required to use a 1-hot code for the state assignment, you need 5 bits for each of the five states. For example, you will use "10000" for state A and "01000" for state B and so on.**

- **Your first state (let's say A) will be the starting state. If your first button push is correct (let's say Xa=1), then your state machine goes to next state (i.e., B) which has state "01000/0" (/0 means that the output is zero, which means the lock is not unlocked yet). If your first button push is wrong (i.e., Xa=0), then your state machine goes no where (in other words, the next state is the current state). Similarly, your state machine (or say state diagram) reaches its final state (00001/1), which represents opening the lock successfully after it saw "Xa, Xb, Xc, and Xd" (your secret code).**

- **You cannot arbitrarily use four digits to be your secret code. Instead, you need to use a general one like "Xa Xb Xc Xd". Your circuit should be general in the sense that your secret code can be any of 4 digits in the range of 0 to 9.**

# State Diagram of Q4-17



# 7. Question 4-17

| Present state | Inputs | | | | Next state | Output |
| --- | --- | --- | --- | --- | --- | --- |
| *ABCDE* | *Xa* | *Xb* | *Xc* | *Xd* | *ABCDE* | *U* |
| 10000 | 0 | x | x | x | 10000 | 0 |
| 10000 | 1 | x | x | x | 01000 | 0 |
| 01000 | x | 0 | x | x | 10000 | 0 |
| 01000 | x | 1 | x | x | 00100 | 0 |
| 00100 | x | x | 0 | x | 10000 | 0 |
| 00100 | x | x | 1 | x | 00010 | 0 |
| 00010 | x | x | x | 0 | 10000 | 0 |
| 00010 | x | x | x | 1 | 00001 | 0 |
| 00001 | x | x | x | x | 00001 | 1 |

# Terms of Use


- Return to Title Page