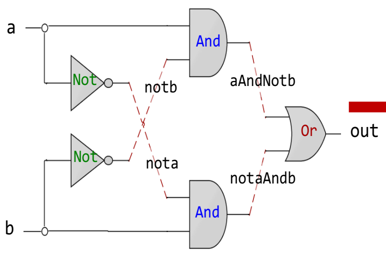
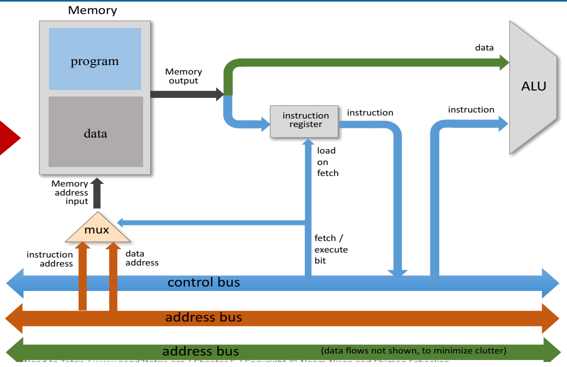
# Digital Logic Design



```
CHIP Xor {
  IN a, b;
  OUT out;
  PARTS:
  Not(in=a, out=nota);
  Not(in=b, out=notb);
  And(a=nota, b=b, out=w1);
  And(a=a, b=notb, out=w2);
  Or(a=w1, b=w2, out=out);
}
```



```
@R1
D=M
@temp
M=D
```

```
0000000000000001
1111110000010000
0000000000010000
1110001100001000
```

# Lecture # 17

# Design of Synchronous Sequential Circuits

```
#include<stdio.h>
#include<stdlib.h>
int main(){
  printf("Learning is fun with Arif\n");
  exit(0);
}
```

```
global main
SECTION .data
   msg: db "Learning is fun with Arif", 0Ah, 0h
   len_msg: equ $ - msg
SECTION .text
   main:
      mov rax,1
      mov rdi,1
      mov rsi,msg
      mov rdx,len_msg
      syscall
      mov rax,60
      mov rdi,0
      syscall
```

```
0:  b8 01 00 00 00
5:  bf 01 00 00 00
a:  48 be 00 00 00 00 00
11: 00 00 00
14: ba 1b 00 00 00
19: 0f 05
1b: b8 3c 00 00 00
20: bf 00 00 00 00
25: 0f 05
```

Slides of first half of the course are adapted from:
https://www.nand2tetris.org
Download s/w tools required for first half of the course from the following link:
https://drive.google.com/file/d/0B9c0BdDJz6XpZUh3X2dPR1o0MUE/view

Instructor: Muhammad Arif Butt, Ph.D.

# Today's Agenda

**Steps of Designing Synchronous Sequential Circuits**

1. Construct **State Diagram** from problem statement
2. Derive **State Table** from state diagram
3. Perform **State Reduction,** if possible (optional)
4. Do **State Assignment** and decide on **number** of Flip Flops to be used
5. Construct **Excitation Table**
6. Derive **Circuit output and Flip Flop input equations**
7. Draw the **Circuit Diagram**

## Example 1

**Problem Statement:** Design a sequence recognizer circuit that detects an input sequence of ‘**1011**’. The sequence recognizer outputs a ‘1’ on the detection of this input sequence

## Step 1: Construct State Diagram

- **Problem Statement:** Design a sequence recognizer circuit that detects an input sequence of ‘**1011**’. The sequence recognizer outputs a ‘1’ on the detection of this input sequence

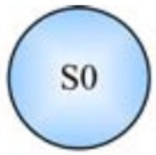
| **Input** | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Output** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

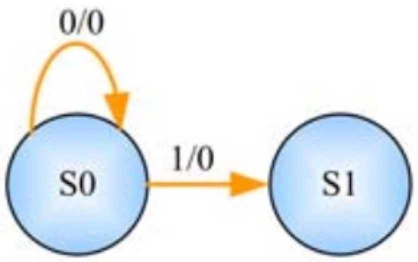
- **State Diagram:** A state diagram consists of circles (which represent the states) and directed arcs that connect the circles and represent the transitions between states. In a state diagram:
    - The number of circles is equal to the number of states. Every state is given a label (or a binary encoding) written inside the corresponding circle
    - The number of arcs leaving any circle is **$2^n$**, where **n** is the number of inputs of the sequential circuit
    - The label of each arc has the notation x/y, where x is the input vector that causes the state transition, and y is the value of the output during that present state
    - An arc may leave a state and end up in the same or any other state

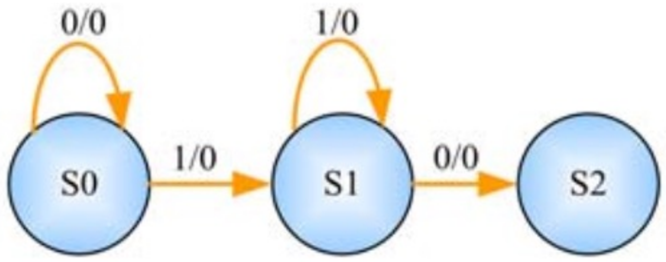
# Step 1: Construct State Diagram (cont…)

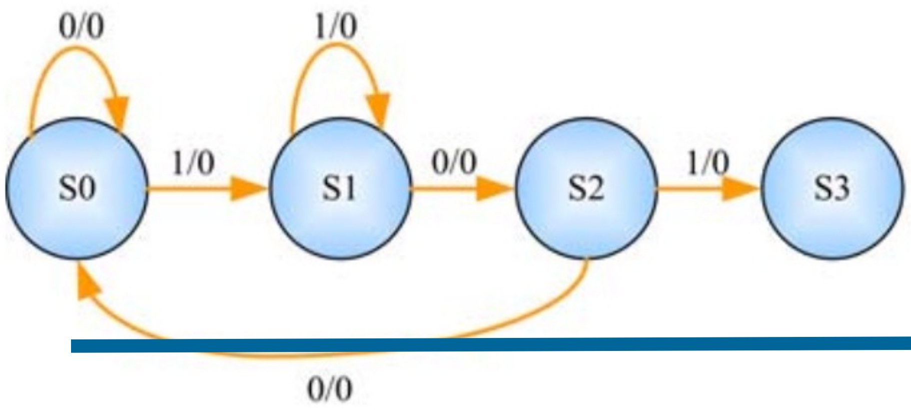
| | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Input** | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |
| **Output** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

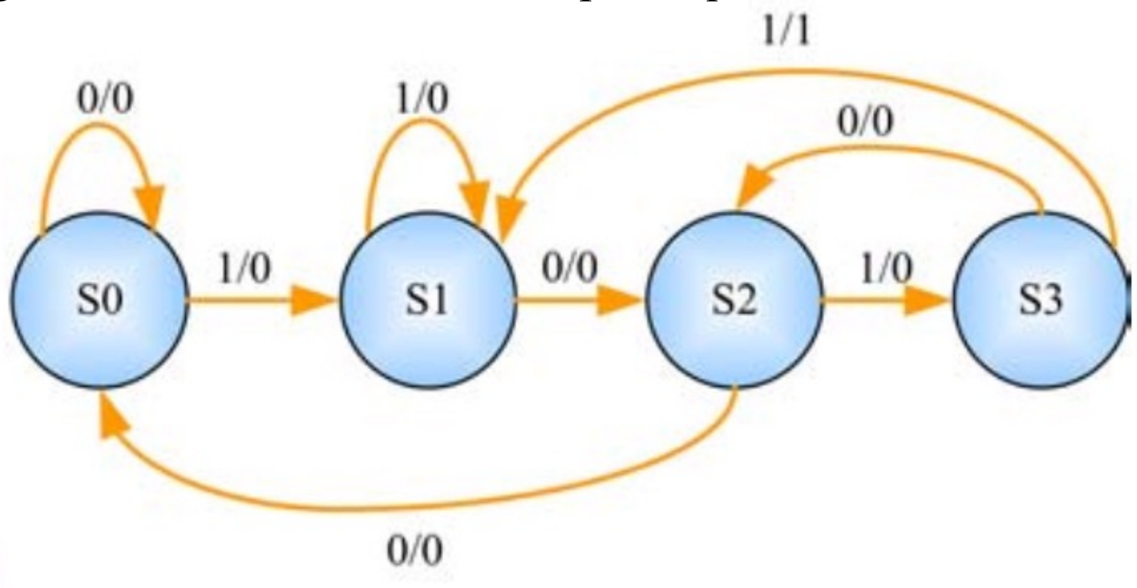
Design a sequence recognizer circuit that detects an input sequence of '**1011**'





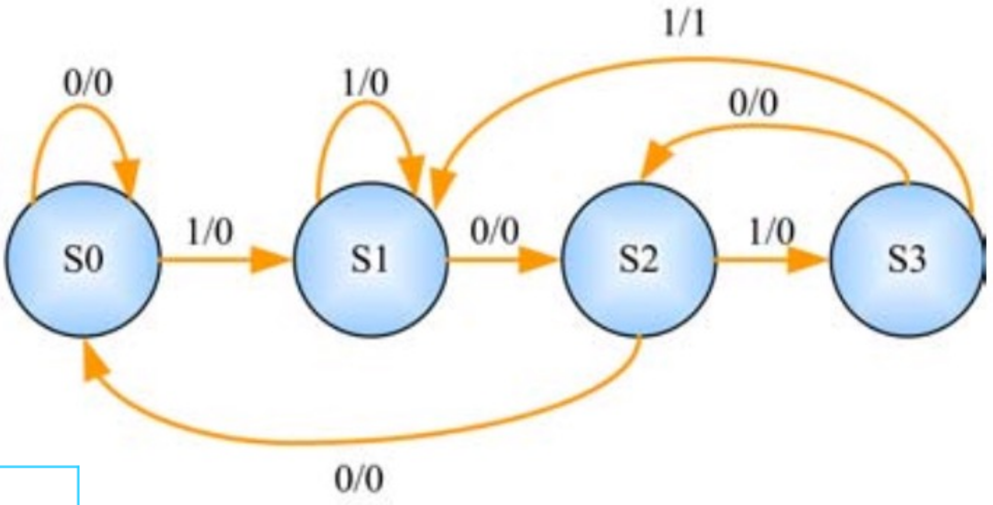






## Step 2: Derive State Table

- A **State Table** represents the time sequence of inputs, outputs and states in a tabular form. The state table of the given state diagram is given below:

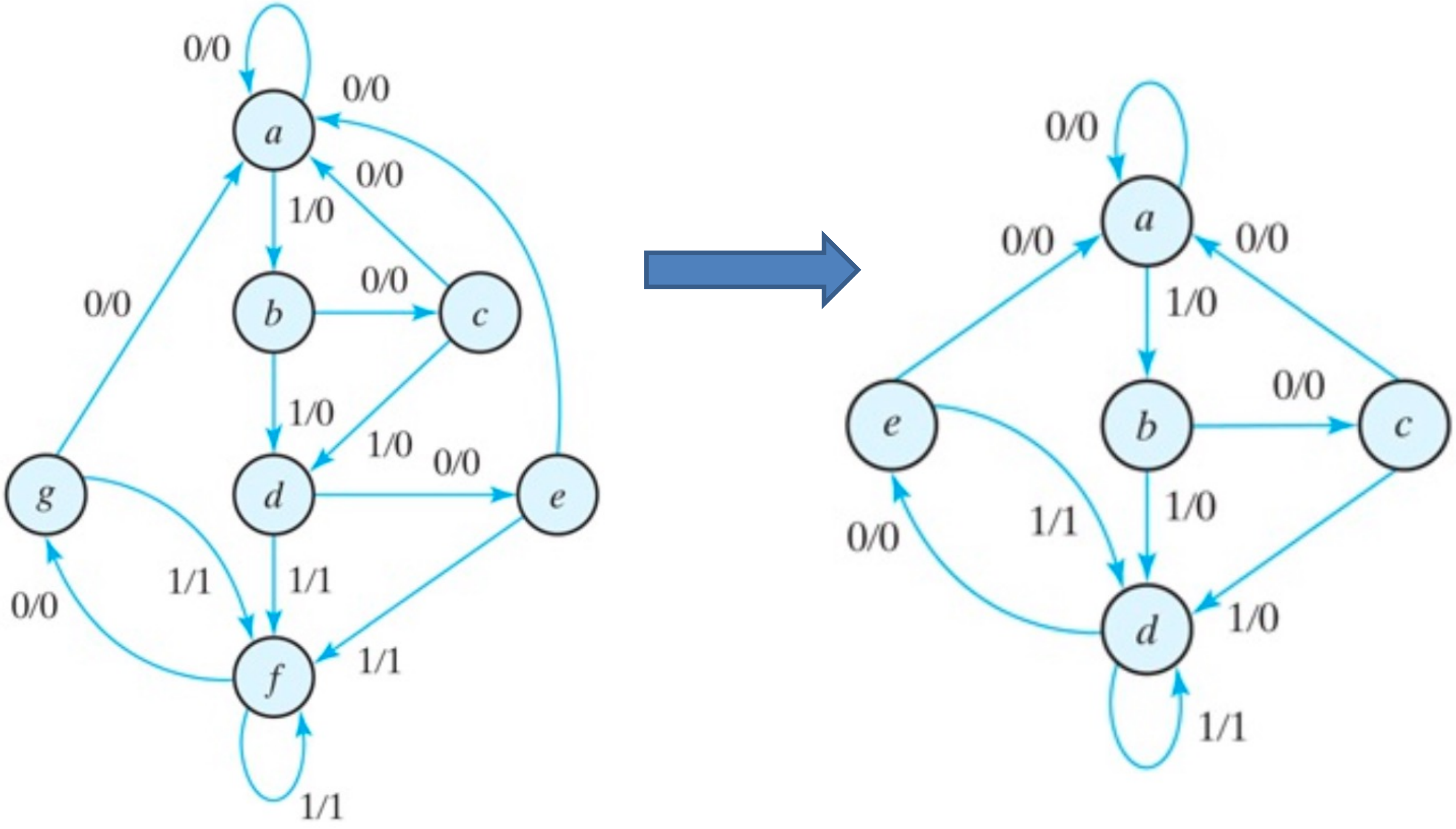


| Inputs of Combinational Circuit | | Next State | Output |
|---|---|---|---|
| Present State | Input | | |
| *S0* | 0 | *S0* | 0 |
| *S0* | 1 | *S1* | 0 |
| *S1* | 0 | *S2* | 0 |
| *S1* | 1 | *S1* | 0 |
| *S2* | 0 | *S0* | 0 |
| *S2* | 1 | *S3* | 0 |
| *S3* | 0 | *S2* | 0 |
| *S3* | 1 | *S1* | 1 |

| Present State | Next State | | Output | |
|---|---|---|---|---|
| | X=0 | X=1 | X=0 | X=1 |
| *S0* | *S0* | *S1* | 0 | 0 |
| *S1* | *S2* | *S1* | 0 | 0 |
| *S2* | *S0* | *S3* | 0 | 0 |
| *S3* | *S2* | *S1* | 0 | 1 |

## Step 3: State Reduction

- State reduction means reducing the states of the state diagram, while keeping the input and output requirements unchanged



## Step 4: State Assignment and Number of FF

State assignment is the process of assigning binary codes to states. In the state diagram since we have four states, so two bit code is used. If there are unused states, then they are treated as don't care conditions during design.

| *State* | *Assignment* |
|---|---|
| *S0* | 00 |
| *S1* | 01 |
| *S2* | 10 |
| *S3* | 11 |

**State Table with Symbolic States**

| Inputs of Combinational Circuit | | Next State | Output |
|---|---|---|---|
| **Present State** | **Input** | | |
| *S0* | 0 | *S0* | 0 |
| *S0* | 1 | *S1* | 0 |
| *S1* | 0 | *S2* | 0 |
| *S1* | 1 | *S1* | 0 |
| *S2* | 0 | *S0* | 0 |
| *S2* | 1 | *S3* | 0 |
| *S3* | 0 | *S2* | 0 |
| *S3* | 1 | *S1* | 1 |

**State Table with Binary States**

| Inputs of Combinational Circuit | | Next State | Output |
|---|---|---|---|
| **Present State** | **Input** | | |
| ***A B*** | ***X*** | ***A B*** | ***Y*** |
| 0 0 | 0 | 0 0 | 0 |
| 0 0 | 1 | 0 1 | 0 |
| 0 1 | 0 | 1 0 | 0 |
| 0 1 | 1 | 0 1 | 0 |
| 1 0 | 0 | 0 0 | 0 |
| 1 0 | 1 | 1 1 | 0 |
| 1 1 | 0 | 1 0 | 0 |
| 1 1 | 1 | 0 1 | 1 |

## Step 5: Construct Excitation Table

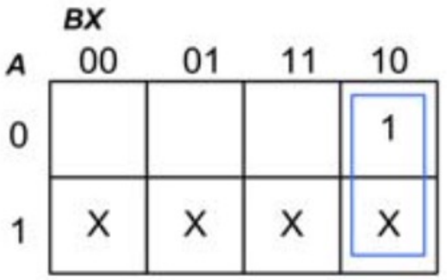
- Suppose we decide to use JK and D Flip Flops.

| $Q_t$ | $Q_{t+1}$ | J | K |
|---|---|---|---|
| 0 | 0 | 0 | X |
| 0 | 1 | 1 | X |
| 1 | 0 | X | 1 |
| 1 | 1 | X | 0 |

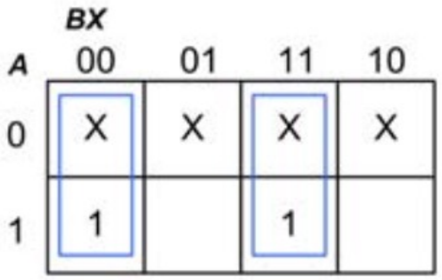
| $Q_t$ | $Q_{t+1}$ | D |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

| *Present State* | | *Input* | Next State | | o/p | *Flip-flops Inputs* | | |
|---|---|---|---|---|---|---|---|---|
| *A* | *B* | *X* | *A* | *B* | *Y* | $J_A$ | $K_A$ | $D_B$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | X | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | X | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 | X | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | X | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | X | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | X | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 0 | X | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 | X | 1 | 1 |

# Step 6: Derive Circuit o/p and FF i/p Equations

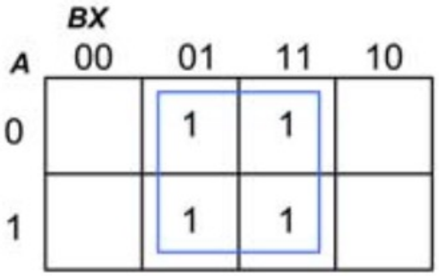


JA = BX′

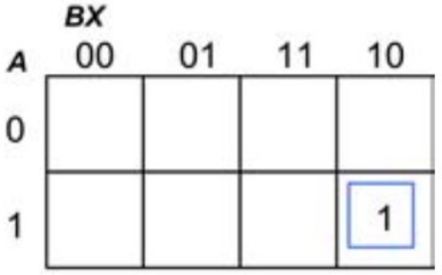


KA = BX + B′X′

| Present State | | Input | Next State | | o/p | Flip-flops Inputs | | |
|---|---|---|---|---|---|---|---|---|
| *A* | *B* | *X* | *A* | *B* | *Y* | $J_A$ | $K_A$ | $D_B$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | X | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | X | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 | X | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | X | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | X | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | X | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 0 | X | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 | X | 1 | 1 |



DB = X



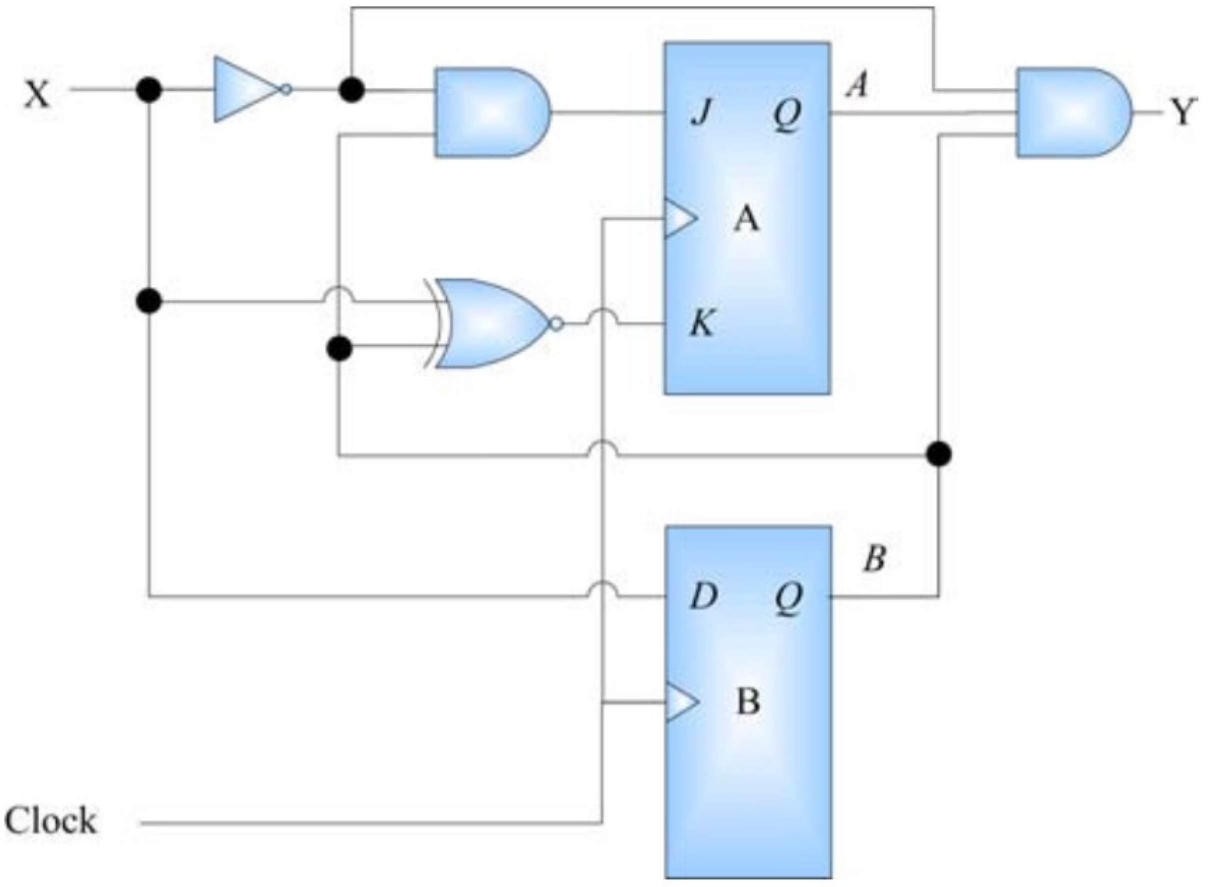
Y = ABX′

## Step 7: Circuit Diagram

JA = BX′ KA = BX + B′X′

DB = X

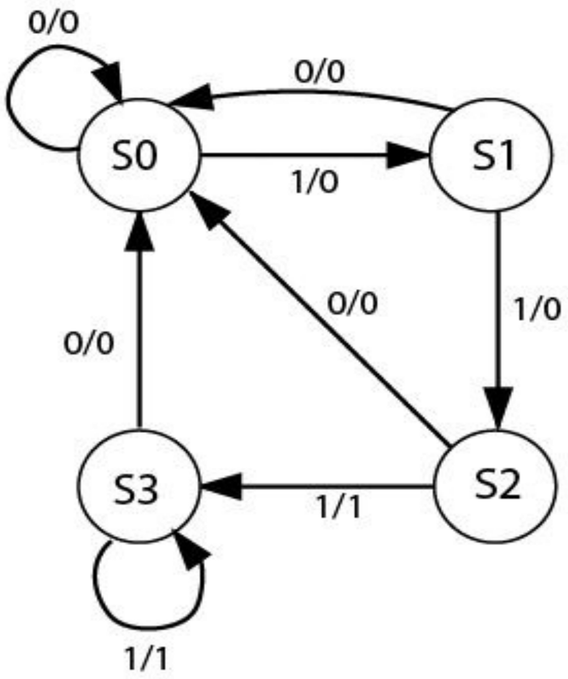
Y = ABX′



## Example 2

**Problem Statement:** Design a sequence recognizer circuit that detects three or more consecutive 1s in a string of bits coming through an input line. The output becomes 1 on detection of every third consecutive 1, otherwise it remains 0.

- Use D Flip Flops
- Use JK Flip Flops
- Use T Flip Flops



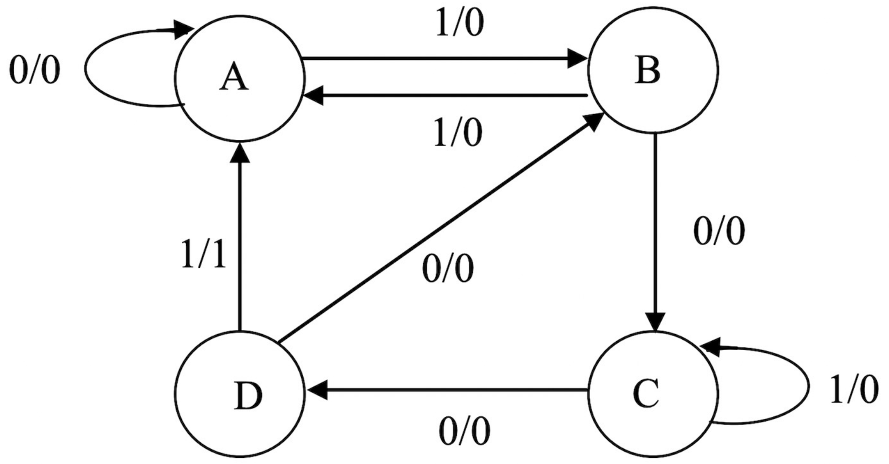
### Steps of Designing Synchronous Sequential Circuits

1. Construct **State Diagram** from problem statement
2. Derive **State Table** from state diagram
3. Perform **State Reduction,** if possible (optional)
4. Do **State Assignment** and decide on **number** of Flip Flops to be used
5. Construct **Excitation Table**
6. Derive **Circuit output and Flip Flop input equations**
7. Draw the **Circuit Diagram**

# Example 3

**Problem Statement:** Design the sequential circuit of following state diagram using:

- Use D Flip Flops
- Use JK Flip Flops
- Use T Flip Flops



**Steps of Designing Synchronous Sequential Circuits**

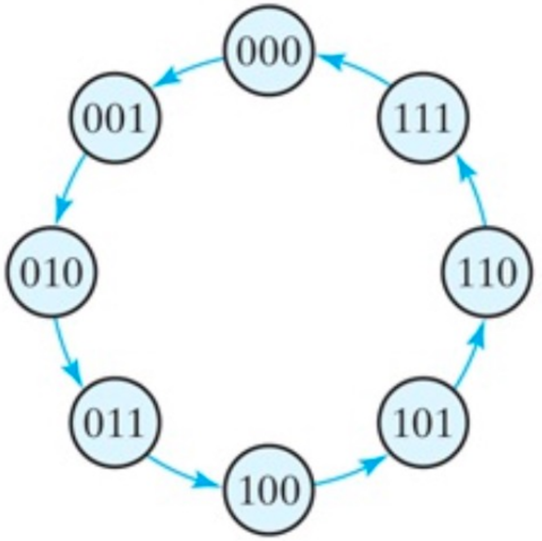
1. Construct **State Diagram** from problem statement
2. Derive **State Table** from state diagram
3. Perform **State Reduction,** if possible (optional)
4. Do **State Assignment** and decide on **number** of Flip Flops to be used
5. Construct **Excitation Table**
6. Derive **Circuit output and Flip Flop input equations**
7. Draw the **Circuit Diagram**

## Counters

- A counter is a special type of register that goes through a pre-determined sequence of states upon the application of input pulses
- **Counters are used for:**
  - Counting the number of occurrences of an event
  - Keeping time or calculating amount of time between events
  - Baud rate generation
- **A w-bit counter consists of two main elements:**
  - A w-bit register to store a w-bit value
  - A combinational logic to
    - Compute the next value (according to a specific counting function)
    - Load a new value of user/programmer choice
    - Reset the counter to a default value
- **Examples:**
  - Simple Up/Down Binary Counters
  - BCD Counter(s)
  - Gray Code Counter
  - Ring Counter
  - Johnson Counter

## Example 4

**Problem Statement:** Design a 3 bit binary up counter using D Flip Flops

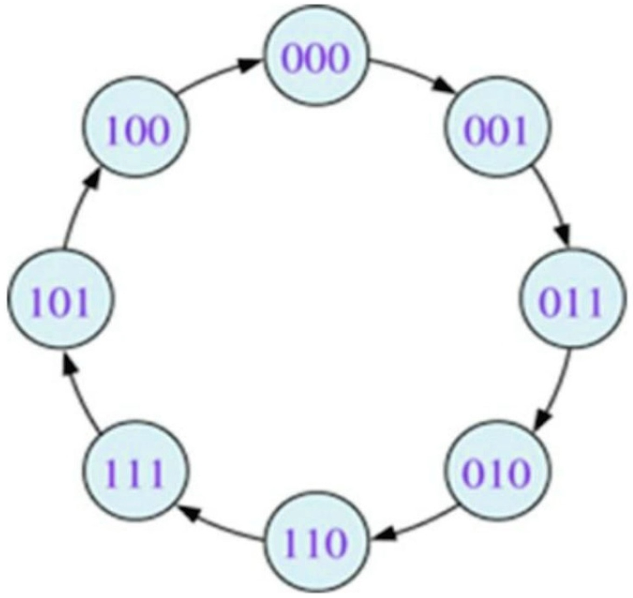


**Steps of Designing Synchronous Sequential Circuits**

1. Construct **State Diagram** from problem statement
2. Derive **State Table** from state diagram
3. Perform **State Reduction,** if possible (optional)
4. Do **State Assignment** and decide on **number** of Flip Flops to be used
5. Construct **Excitation Table**
6. Derive **Circuit output and Flip Flop input equations**
7. Draw the **Circuit Diagram**

## Example 5

**Problem Statement:** Design a 3 bit Grey Code counter using D Flip Flops



**Steps of Designing Synchronous Sequential Circuits**

1. Construct **State Diagram** from problem statement
2. Derive **State Table** from state diagram
3. Perform **State Reduction,** if possible (optional)
4. Do **State Assignment** and decide on **number** of Flip Flops to be used
5. Construct **Excitation Table**
6. Derive **Circuit output and Flip Flop input equations**
7. Draw the **Circuit Diagram**

## Mealy vs Moore Machines

- The output of a Mealy Machine is determined by both its current state and current inputs
- The output of a Moore machine is determined by its current state only

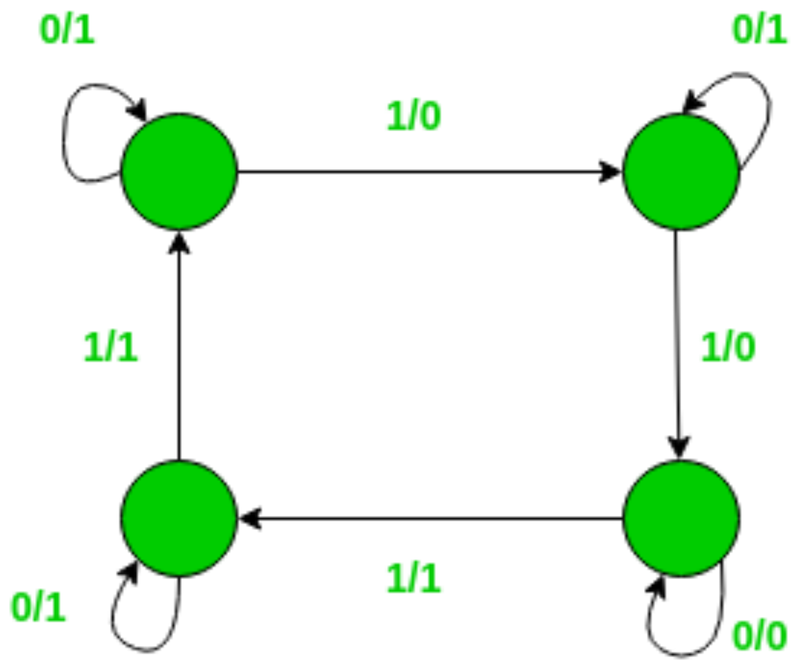


**Figure** - Mealy machine

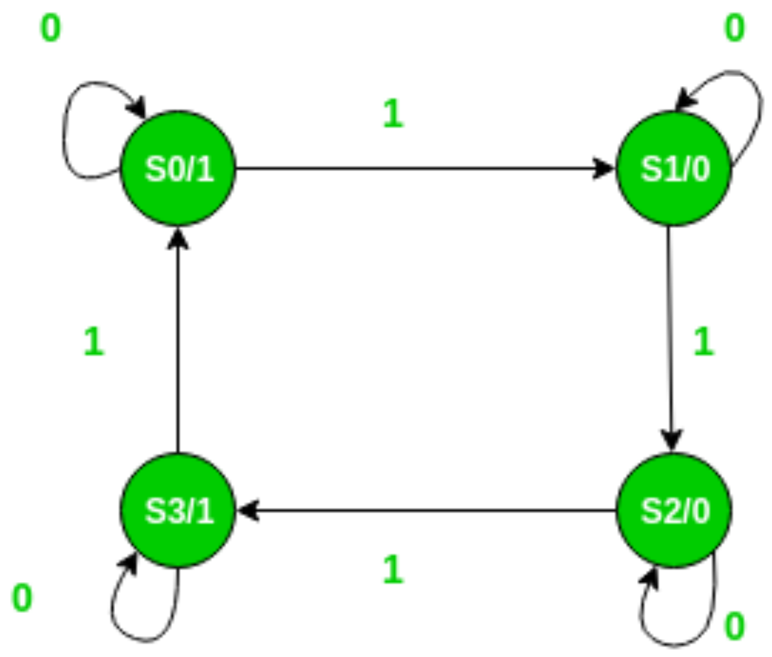


**Figure** - Moore machine

## Things To Do

- Practice
- Practice
- Practice
- Practice
- Practice
- Practice
- Practice
- Practice

**Coming to office hours does NOT mean you are academically weak!**