Sequential Circuit Design

COE 202

Digital Logic Design

Dr. Muhamed Mudawar

King Fahd University of Petroleum and Minerals

Presentation Outline

- The Design Procedure
- Moore Sequence Detector
- Mealy Sequence Detector
- Tracing State Diagrams and Verifying Correctness
- Sequential Comparator
- Binary Counter
- Up/Down Counter with Enable

The Design Procedure

Given a Description (or Specification) of the Problem

1. Obtain a state diagram for the sequential circuit
2. Assign binary codes to the states and fill the state table
3. Select the type of Flip-Flops and derive the FF input equations
4. Derive the output equations
5. Draw the circuit diagram
6. Verify the correctness of the final design (verification)

The State Diagram

- A **state** is an abstraction of memory
- A state **remembers** a history of inputs applied to the circuit
- Examples:
  - State **S0** represents the fact that the last input is a **0**
  - State **S1** represents the fact that the last input is a **1**
  - State **S2** represents the fact that the last two-input sequence is **"11"**
- Obtaining the state diagram is the most important step
- Requires experience and good understanding of the problem

Example: Sequence Detector

- A sequence detector is a sequential circuit
- Detects a specific sequence of bits in the input
- The input is a serial **bit stream**:

  One input bit x is fed to the sequence detector each cycle
- The output is also a **bit stream**: One output bit z each cycle

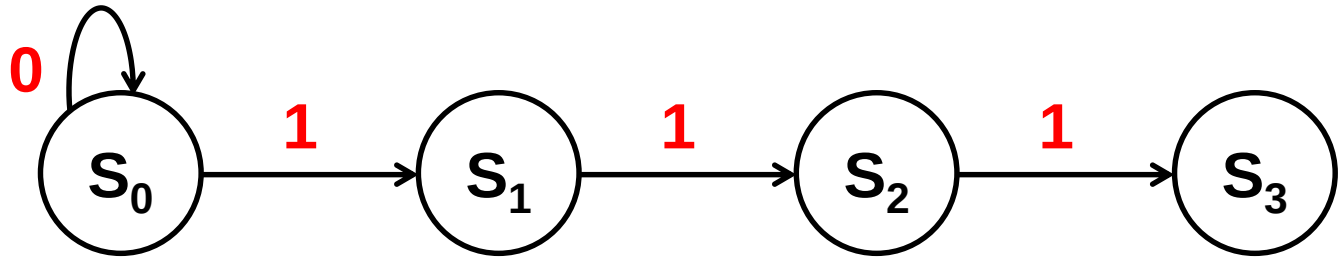
  Indicates whether a given sequence is detected or not

State Diagram for a Sequence Detector

- Example: Design a circuit that detects the input sequence **"111"**
- Begin in an initial state: call it $\mathbf{S_0}$

  $\mathbf{S_0}$ indicates that a **1** is NOT detected yet

  As long as the input x is **0**, remain in the initial state $\mathbf{S_0}$
- Add a state (call it $\mathbf{S_1}$) that detects the first **1** in the input
- Add a state (call it $\mathbf{S_2}$) that detects the input sequence **"11"**
- Add a state (call it $\mathbf{S_3}$) that detects the input sequence **"111"**



Complete the State Diagram

Moore Design: Assign Output to States

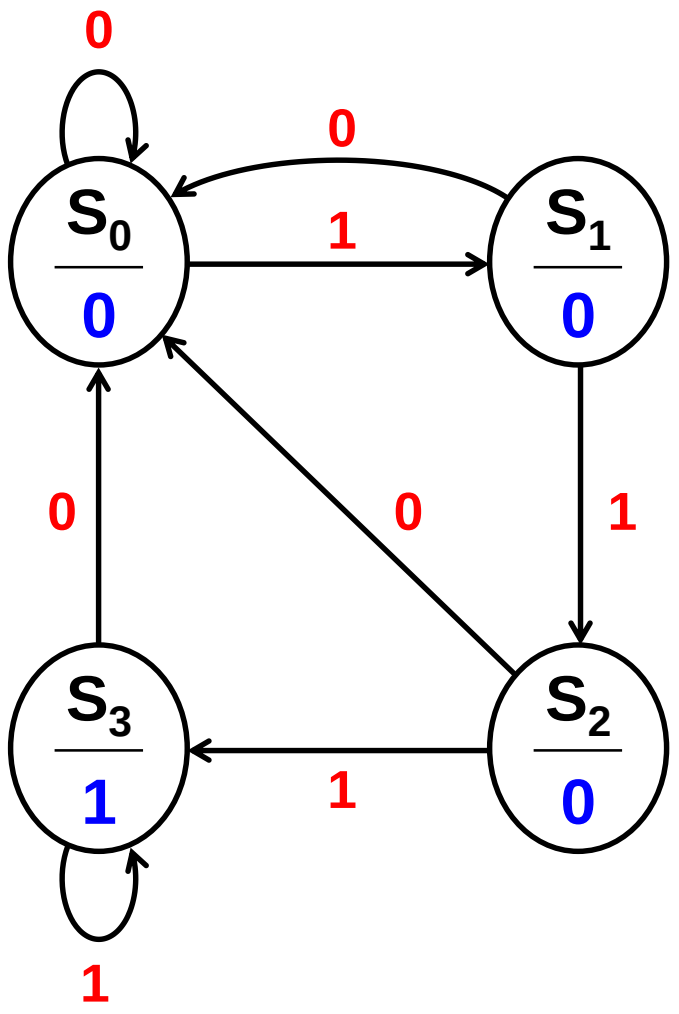
The output in S_0, S_1, and S_2 should be **0**

The output in S_3 should be **1**

Now complete the state diagram:

Add transitions from S_1, S_2, S_3 back to S_0 if the input is **0**

Add transition from S_3 to itself if the input is **1** to detect sequences longer than three **1's**



State Assignment

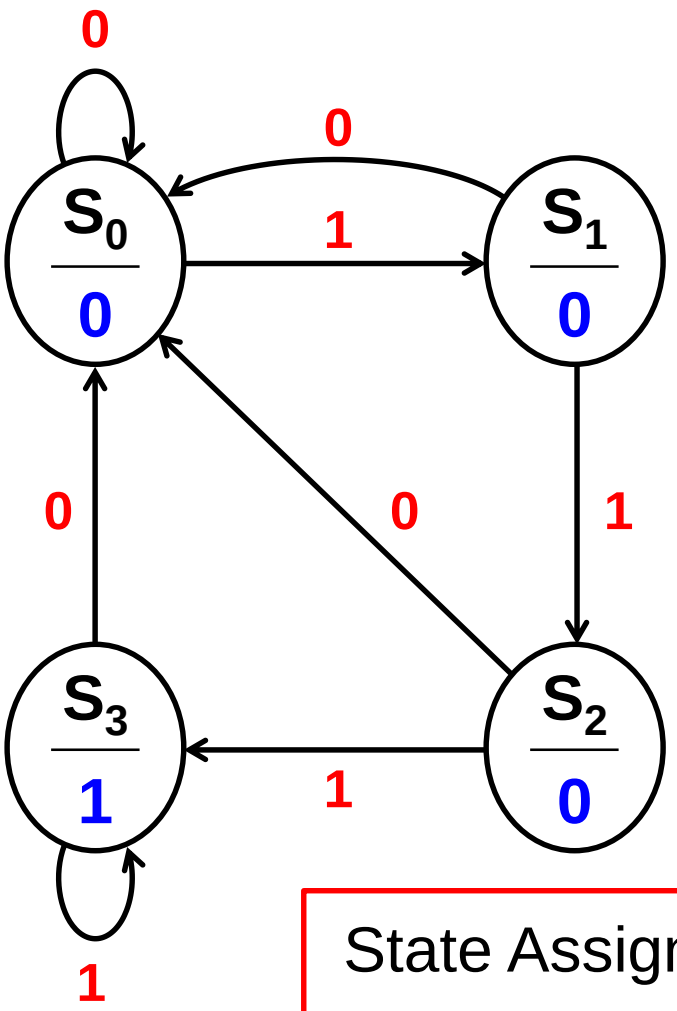
- Each state must be assigned a unique binary code
- If there are m states then

  The minimum number of state bits: $n = \lceil log_2 m \rceil$

  $\lceil x \rceil$ is the smallest integer ≥ x (ceiling function)
- In our example, there are four states: **S_0**, **S_1**, **S_2**, and **S_3**

  Therefore, the minimum number of state bits (Flip-Flops) = 2
- State assignment: **S_0 = 00, S_1 = 01, S_2 = 10** and **S_3 = 11**
- If n bits are used, the number of unused states = $2^n - m$
- In our example, there are NO unused states

From State Diagram to State Table



State Assignment

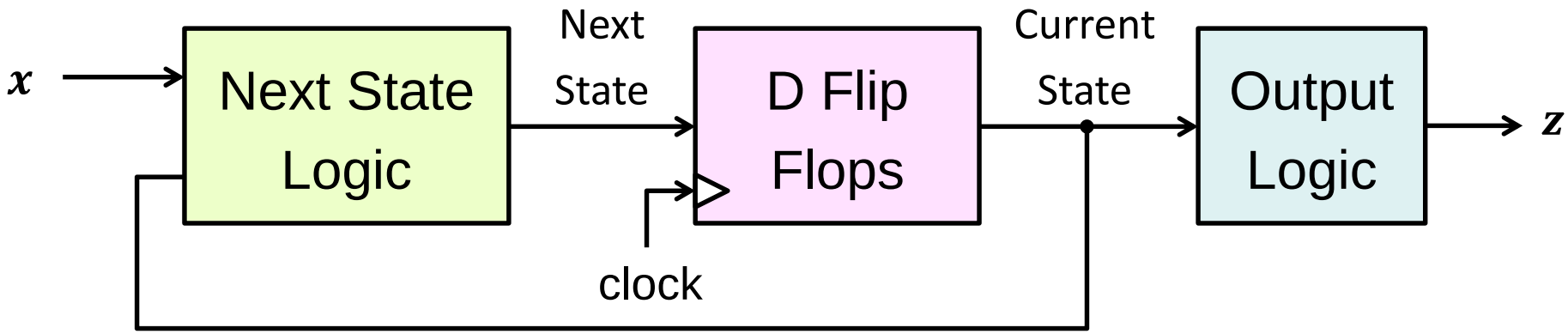
$S_0 = 00$, $S_1 = 01$

$S_2 = 10$, $S_3 = 11$

Present State	Next State		Output
	$x = 0$	$x = 1$	z
S_0	S_0	S_1	0
S_1	S_0	S_2	0
S_2	S_0	S_3	0
S_3	S_0	S_3	1

Present State	Next State		Output
	$x = 0$	$x = 1$	z
0 0	0 0	0 1	0
0 1	0 0	1 0	0
1 0	0 0	1 1	0
1 1	0 0	1 1	1

Structure of a Moore Sequence Detector

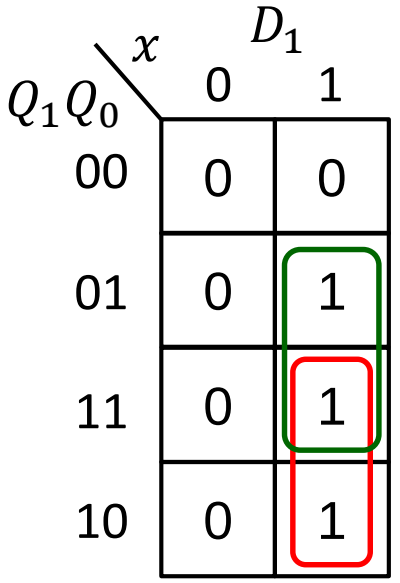


- In our design examples, only D-type Flip-Flops will be used
- They are the simplest to analyze and implement
- Next, we need minimal expressions for

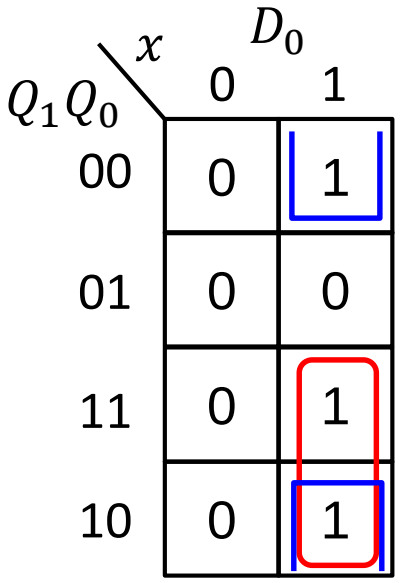
1. Next State Logic
2. Output Logic

Derive Next State an Output Equations

Present State	Next State		Output
	$x = 0$	$x = 1$	z
0 0	0 0	0 1	0
0 1	0 0	1 0	0
1 0	0 0	1 1	0
1 1	0 0	1 1	1



$Q_1\,x + Q_0\,x$



$Q_1\,x + Q_0'\,x$

Two D-type Flips-Flops

Present State = Flip-Flop Outputs Q_1 and Q_0

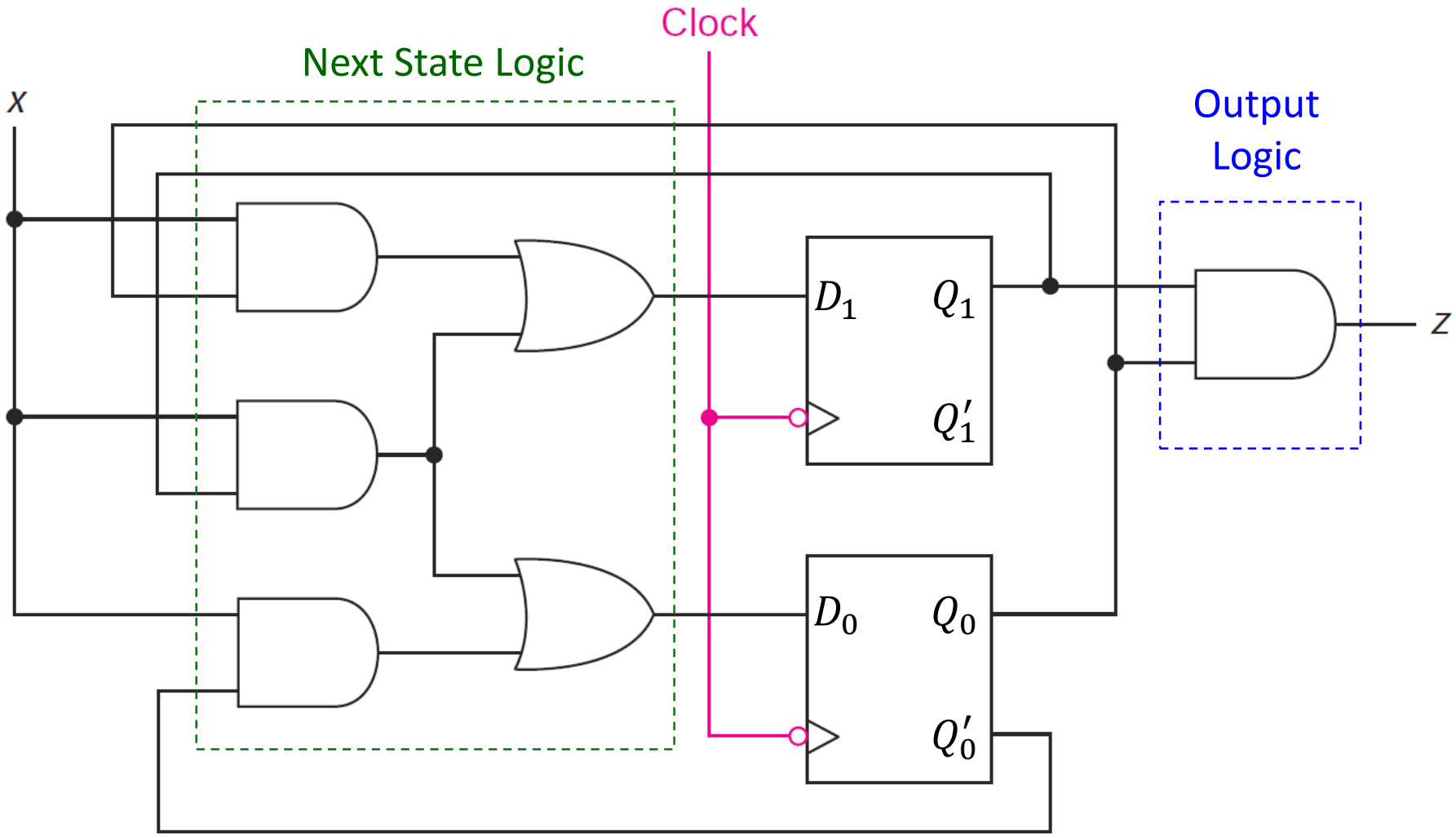
Next State = Flip-Flop Inputs D_1 and D_0

Next State equations: $D_1 = Q_1\,x + Q_0\,x$ and $D_0 = Q_1\,x + Q_0'\,x$

Output equation: $z = Q_1\,Q_0$ (from the state diagram)

Draw the Moore Sequence Detector Circuit

$$D_1 = Q_1\,x + Q_0\,x\ ,\ D_0 = Q_1\,x + Q_0'\,x \qquad z = Q_1\,Q_0$$

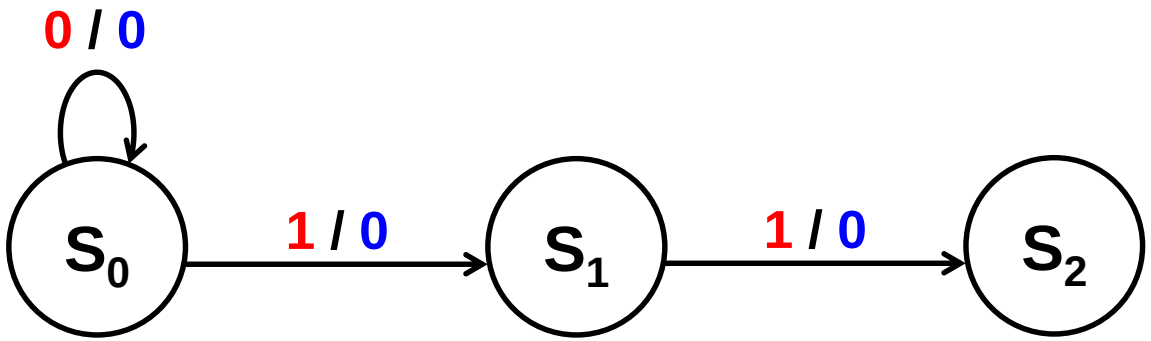


Mealy Type Sequence Detector

- Let us redesign a Mealy type **"111"** sequence detector
- The initial state S_0 indicates that a **1** is NOT detected yet

  As long as the input x is **0**, remain in the initial state S_0

  Notice that **input / output** is written on the arc (Mealy type)
- Add a state (call it S_1) that detects the first **1** in the input
- Add a state (call it S_2) that detects the input sequence **"11"**



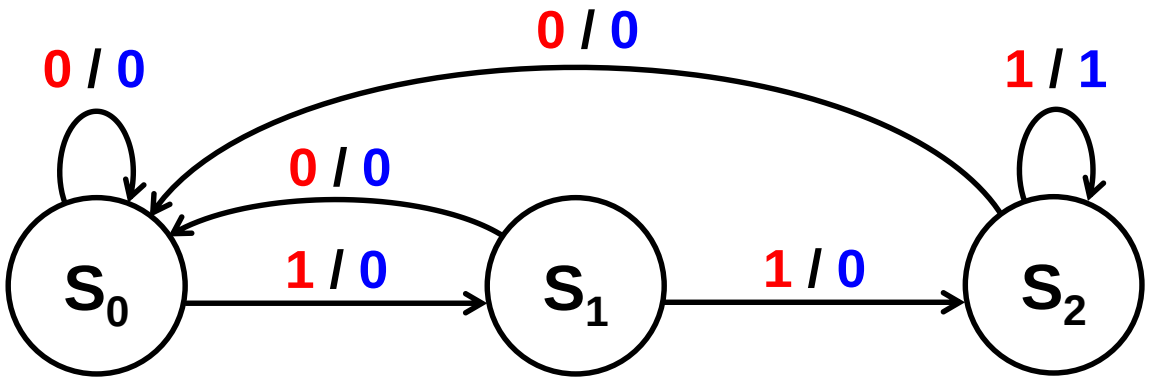
Complete the Mealy State Diagram

- State S_2 is reached after detecting the input sequence **"11"**
- At S_2, if the next input is **1** then the output should be **1**

  Make a transition from S_2 back to itself labeled **1 / 1**

  No need for state S_3, because output is on the arc
- Now complete the state diagram

  Add transitions from S_1 and S_2 back to S_0 when input is **0**

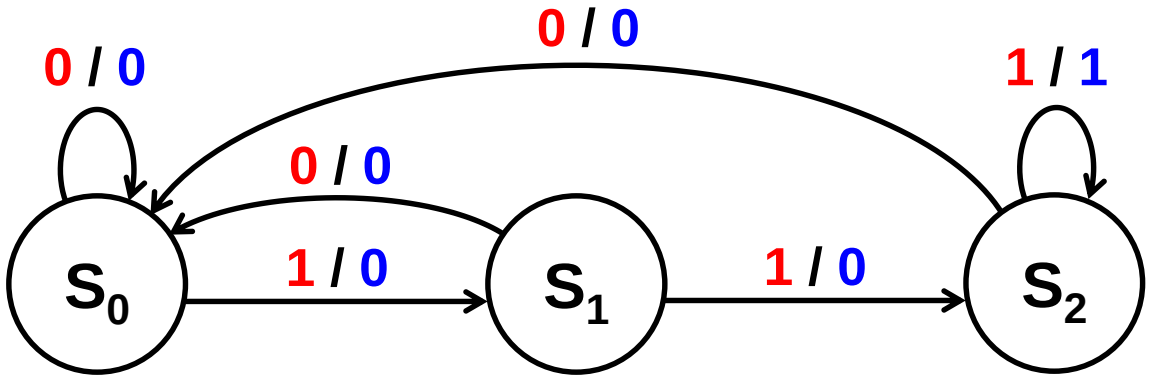


Mealy Machines typically use **less states** than Moore Machines

State Assignment and State Table

Three States ➔ Minimum number of state bits (Flip-Flops) = 2

Assign: $S_0 = 00$, $S_1 = 01$, and $S_2 = 10$ (State **11** is **Unused**)

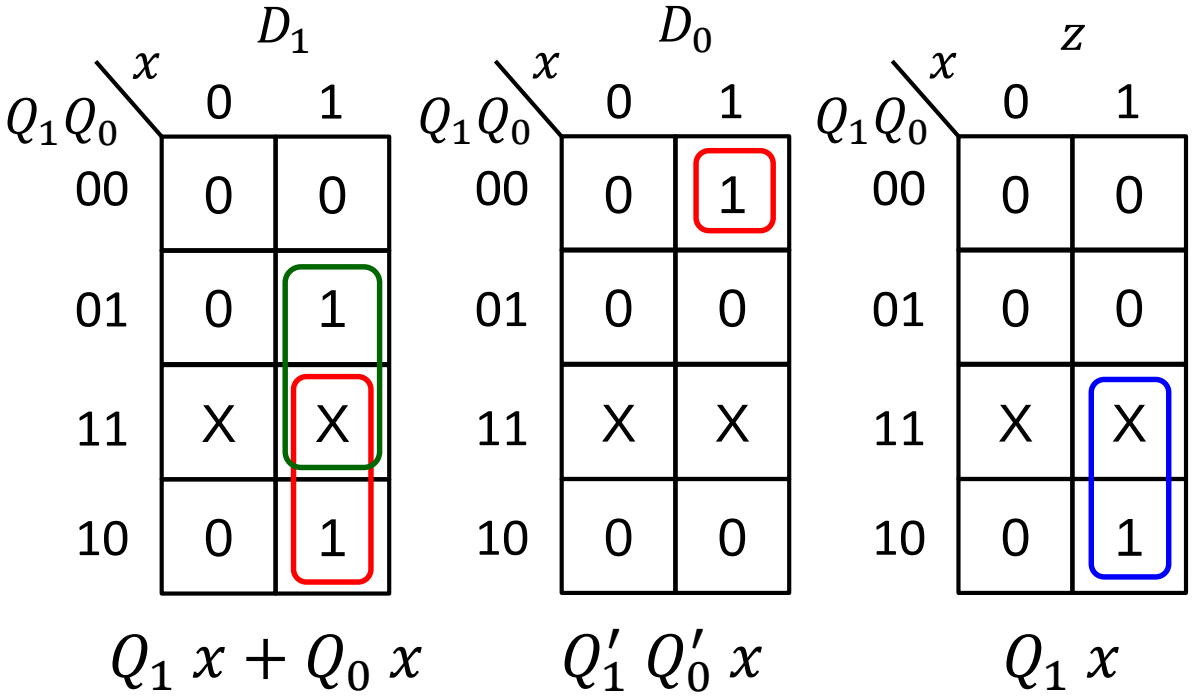


Present State	Next State		Output z	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
S_0	S_0	S_1	0	0
S_1	S_0	S_2	0	0
S_2	S_0	S_2	0	1

Present State	Next State		Output z	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
0 0	0 0	0 1	0	0
0 1	0 0	1 0	0	0
1 0	0 0	1 0	0	1

Derive Next State and Output Equations

Present State	Next State		Output z	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
0 0	0 0	0 1	0	0
0 1	0 0	1 0	0	0
1 0	0 0	1 0	0	1
1 1	X X	X X	X	X



Present State = Flip-Flop Outputs Q_1 and Q_0 (state 11 is unused)

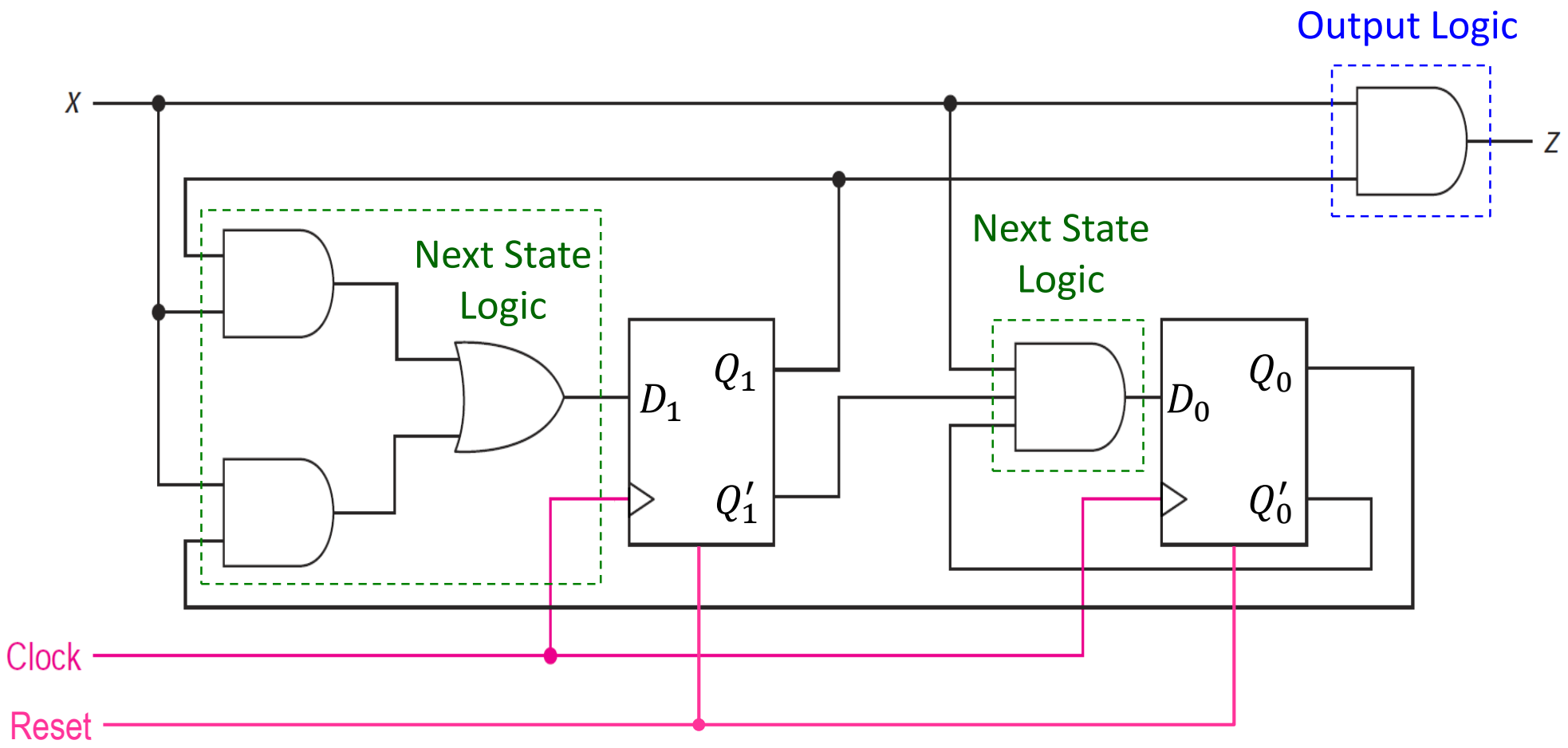
Next State = Flip-Flop Inputs D_1 and D_0

Flip-Flop Input equations: $D_1 = Q_1\,x + Q_0\,x$ and $D_0 = Q_1'\,Q_0'\,x$

Output equation: $z = Q_1\,x$

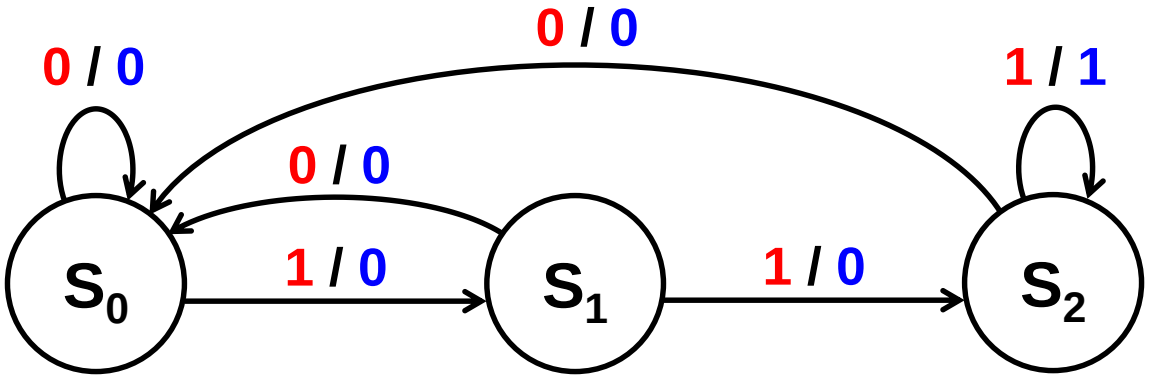
Draw the Mealy Sequence Detector Circuit

$$D_1 = Q_1\,x + Q_0\,x \qquad D_0 = Q_1'\,Q_0'\,x \qquad z = Q_1\,x$$

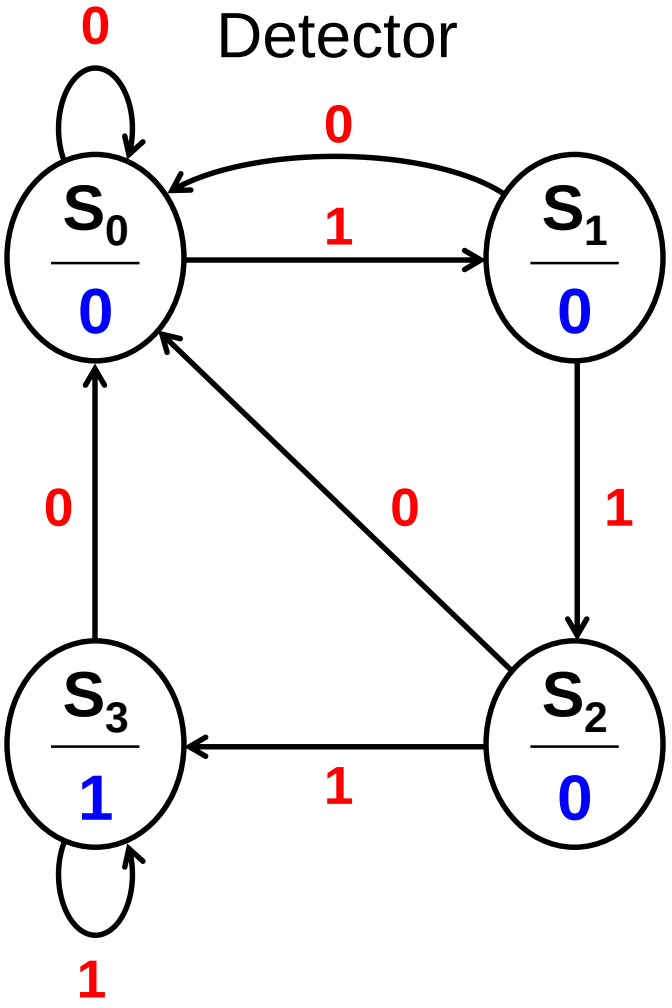


Mealy versus Moore Sequence Detector

Mealy Sequence Detector



Moore Sequence Detector



In general, Moore state diagrams have **more states** than corresponding Mealy.

The drawback of Mealy is that **glitches** can appear in the output if the input is not synchronized with the clock.

Verification

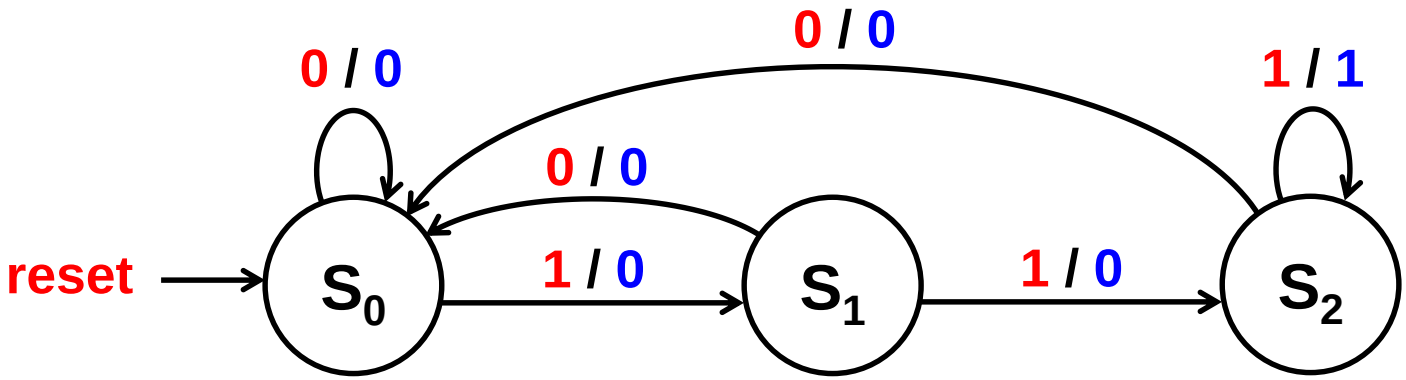
- Sequential circuits should be verified by showing that the circuit produces the original state diagram
- Verification can be done manually, or with the help of a simulation program
- All possible input combinations are applied at each state and the state variables and outputs are observed
- A **reset** input is used to reset the circuit to its initial state
- Apply a sequence of inputs to test all the state-input combinations, i.e., **all transitions** in the state diagram
- Observe the output and the next state that appears after each clock edge in the timing diagram

Input Test Sequence

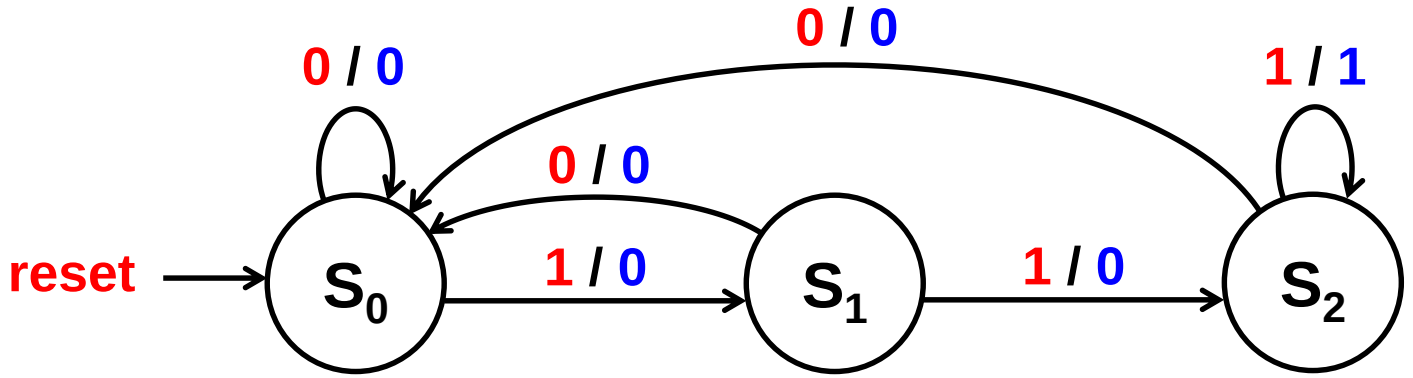
- Required to verify the correct operation of a sequential circuit
- It should test each state transition of the state diagram
- Test sequences can be produced from the state diagram
- Consider the Mealy sequence detector, starting at **S_0** (**reset**), we can use an input test sequence to verify all state transitions:

  Input test sequence: reset then x = 0, 1, 0, 1, 1, 0, 1, 1, 1, 1

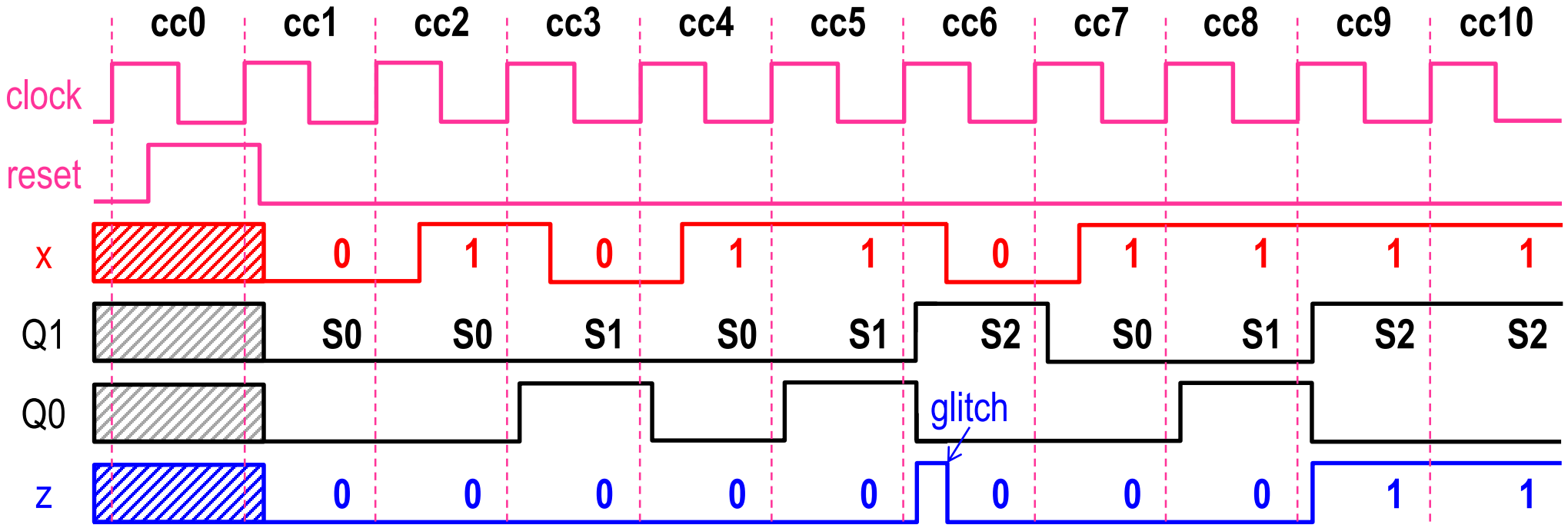
Reset input forces initial state to be **S_0**



Verifying the Mealy Sequence Detector



Input test sequence: reset then x = 0, 1, 0, 1, 1, 0, 1, 1, 1, 1



Sequential Comparator

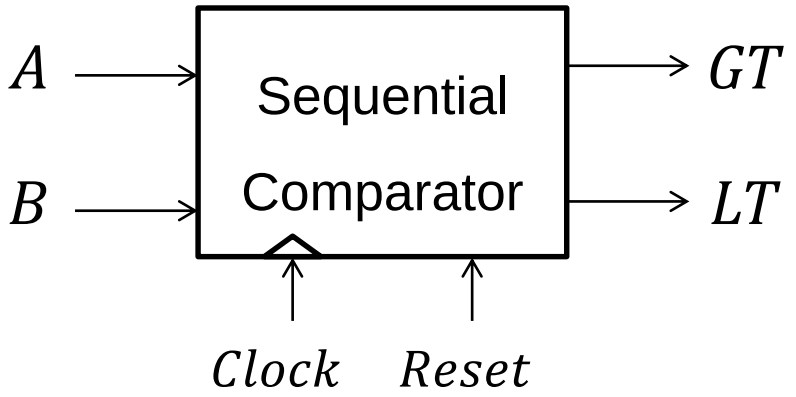
Problem Description:

- Design a sequential circuit that compares two numbers A and B
- Two Inputs: A and B

  A consists of *n* bits

  B consists of *n* bits

  Bit streaming starting at bit 0



  Compare A_0 with B_0, A_1 with B_1, etc. (Bit 0 is least significant bit)

- A reset signal resets the comparator to its initial state

  Reset is required before starting a new comparison

- Two outputs: GT (Greater Than) and LT (Less Than)

Designing the State Diagram

- Reset: start initially in state **EQ**

  EQ indicates Equality (output is **00**)

  Stay in **EQ** as long as A_iB_i = **00** or **11**

- Go to **LT** if $A_i < B_i$ (A_iB_i = **01**)

  LT indicates Less Than (output is **01**)

- Go to **GT** if $A_i > B_i$ (A_iB_i = **10**)

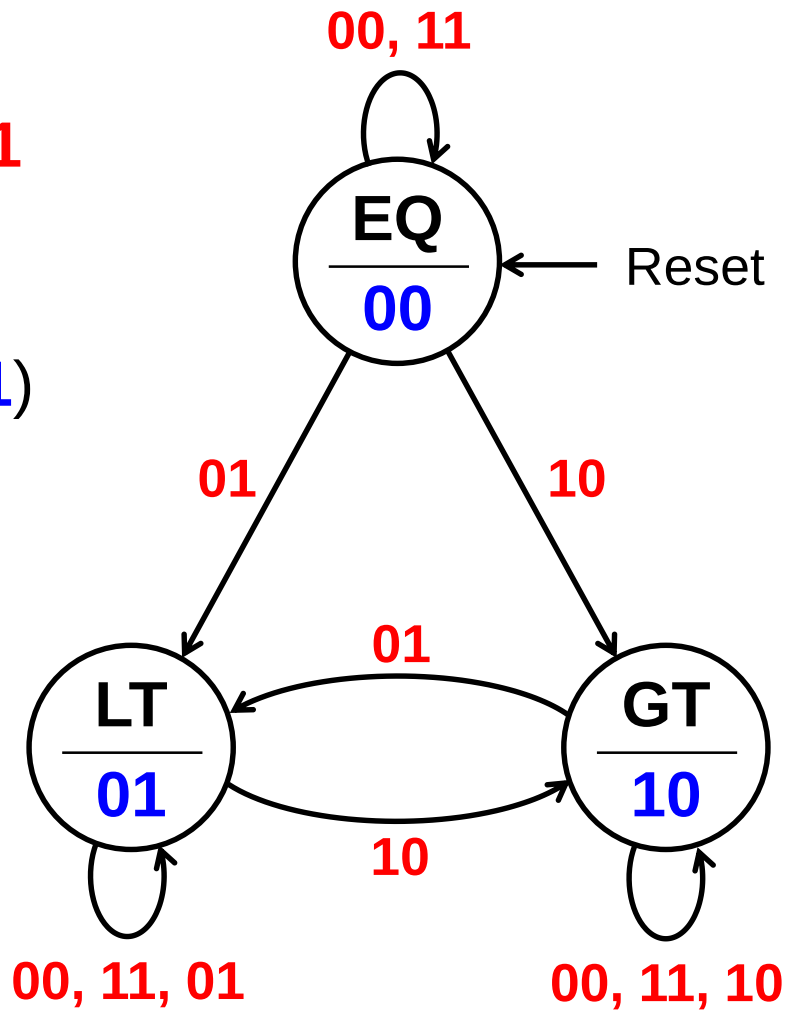
  GT is Greater Than (output is **10**)

- Complete the state diagram

  LT → **GT** and **GT** → **LT**

  LT → **LT** and **GT** → **GT**

Moore State Diagram



State Assignment and State Table

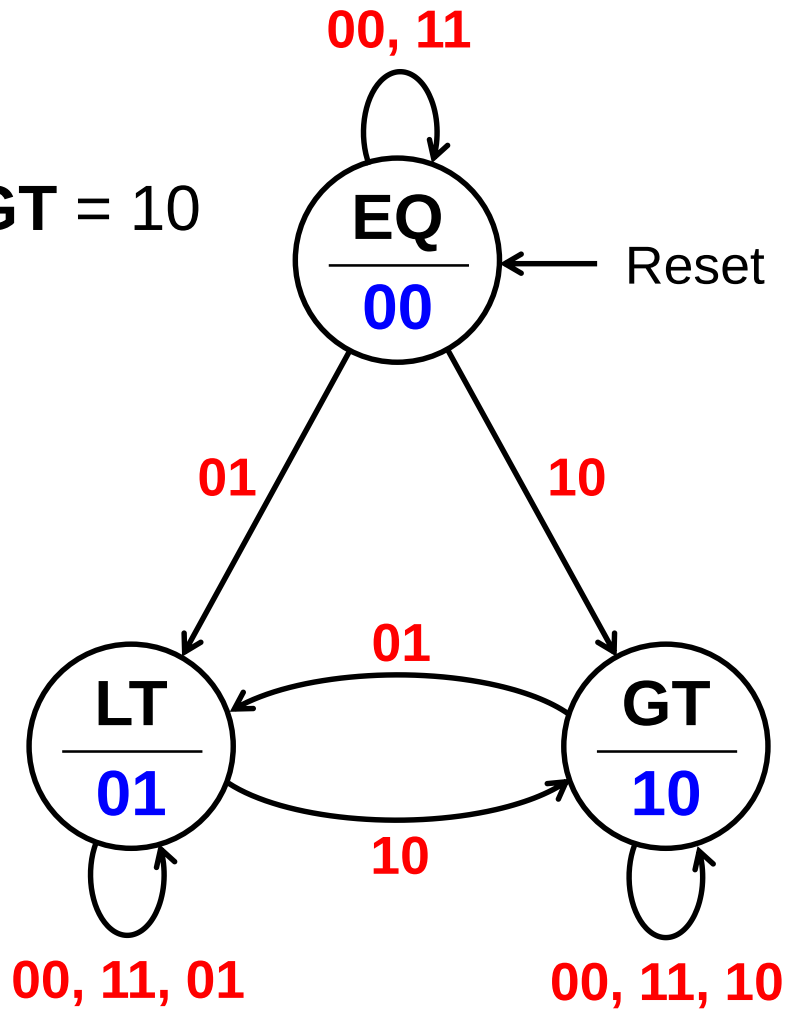
Three States ➔ Two Flip-Flops

D-type Flip-Flops will be used

State Assignment: **EQ** = 00, **LT** = 01, **GT** = 10

Output = Present State (**Q_1** and **Q_0**)

Present State Q_1 Q_0	Next State (D_1 D_0)			
	A B 0 0	A B 0 1	A B 1 0	A B 1 1
0 0	0 0	0 1	1 0	0 0
0 1	0 1	0 1	1 0	0 1
1 0	1 0	0 1	1 0	1 0
1 1	X X	X X	X X	X X



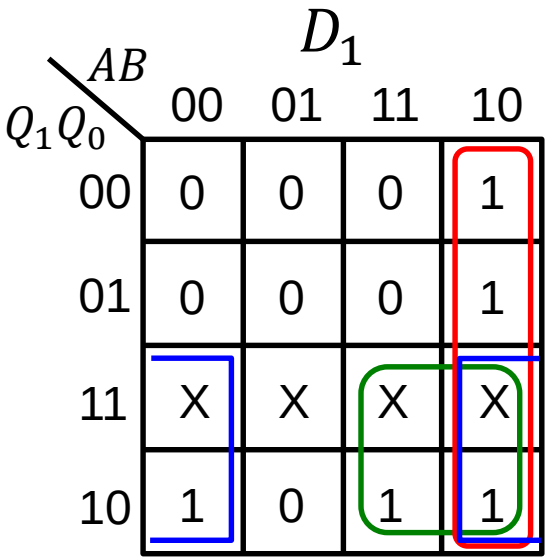
Deriving the Next State Equations

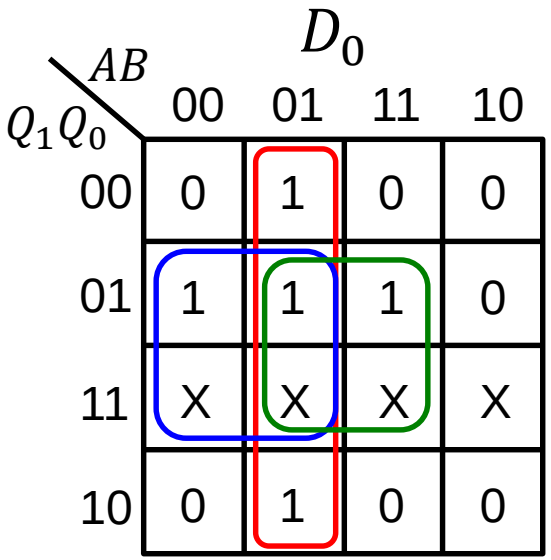
Present State Q_1 Q_0	Next State (D_1 D_0)			
	A B 0 0	A B 0 1	A B 1 0	A B 1 1
0 0	0 0	0 1	1 0	0 0
0 1	0 1	0 1	1 0	0 1
1 0	1 0	0 1	1 0	1 0
1 1	X X	X X	X X	X X

State **11** is **unused**

When present state is 11, **don't cares** are used to fill the next state.

Output = Present state





$$D_1 = Q_1A + Q_1B' + AB'$$

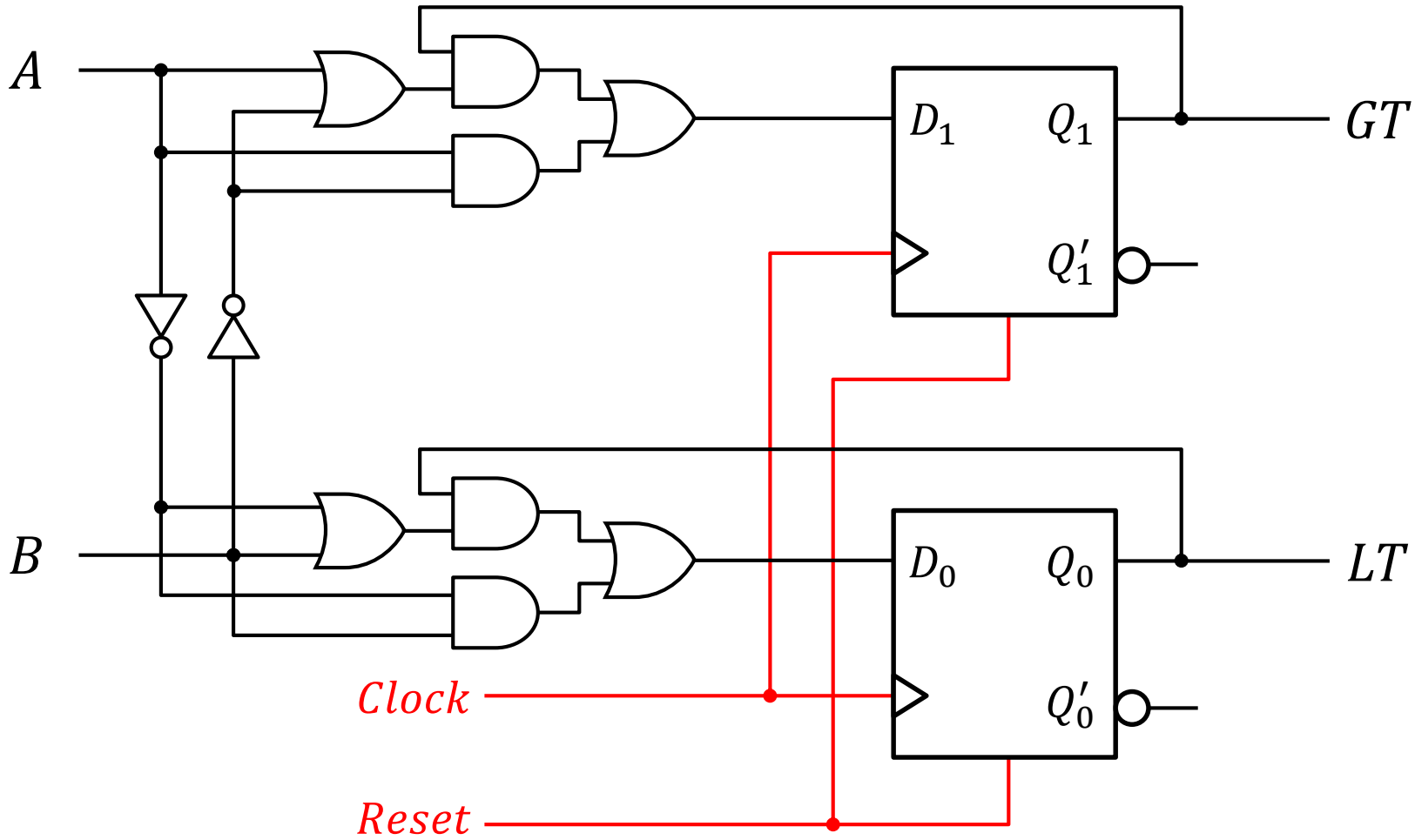
$$D_1 = Q_1(A + B') + AB'$$

$$D_0 = Q_0A' + Q_0B + A'B$$

$$D_0 = Q_0(A' + B) + A'B$$

Sequential Comparator Circuit Diagram

$$D_1 = Q_1(A + B') + AB' \qquad D_0 = Q_0(A' + B) + A'B$$



Designing with Unused States

- A circuit with n flip-flops has 2^n binary states
- However, the state diagram may have m states
- Therefore, the number of unused states = $2^n - m$
- Unused states do not appear in the state diagram
  - Sometimes treated as don't cares when simplifying expressions in K-maps
- However, it is possible to enter an unused state!
  - Caused by outside interference or a circuit malfunction
- Must specify the output and next state values for unused states
- A special (invalid) output can be used to indicate an unused state
- A return to a valid state must be possible without reset
  - Next state for unused states should be specified (NOT don't cares!)

Design of a Binary Counter

Problem Specification:

- Design a circuit that counts up from 0 to 7 then back to 0

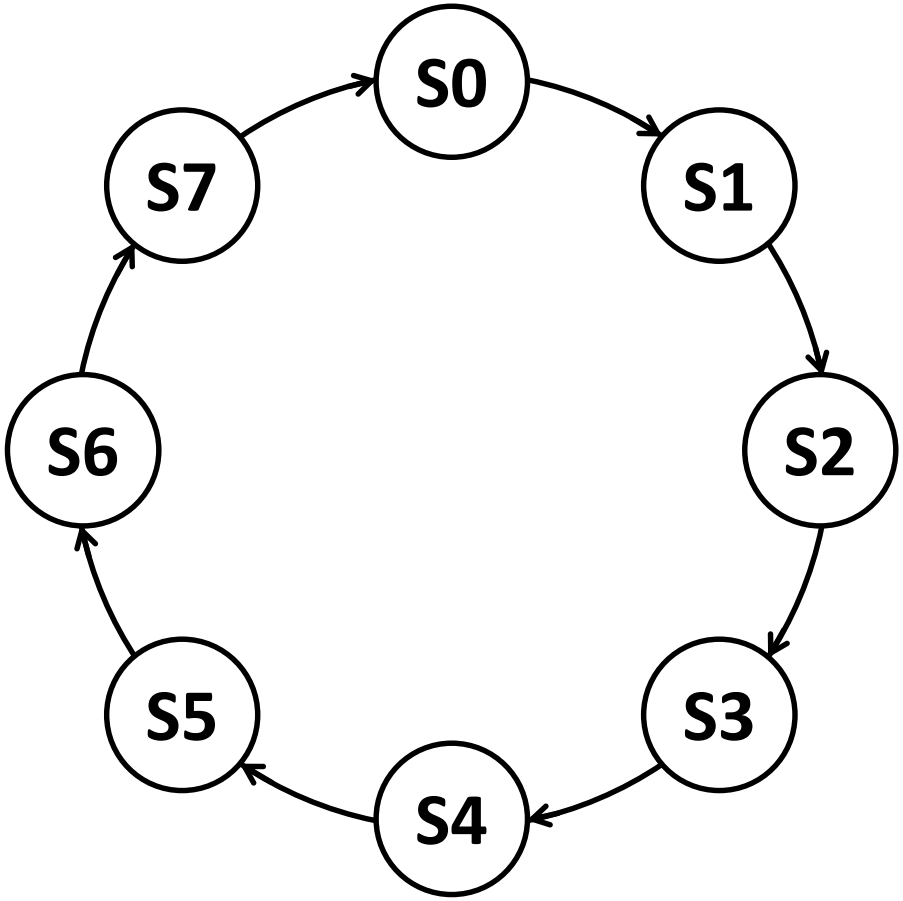
  000 → 001 → 010 → 011 → 100 → 101 → 110 → 111 → 000

  When reaching 7, the counter goes back to 0 then goes up again

- There is no input to the circuit
- The counter is incremented each cycle
- The output of the circuit is the present state (count value)
- The circuit should be designed using D-type Flip-Flops

Designing the State Diagram

- Eight states are needed to store the count values 0 to 7
- No input, state transition happens at the edge of each cycle



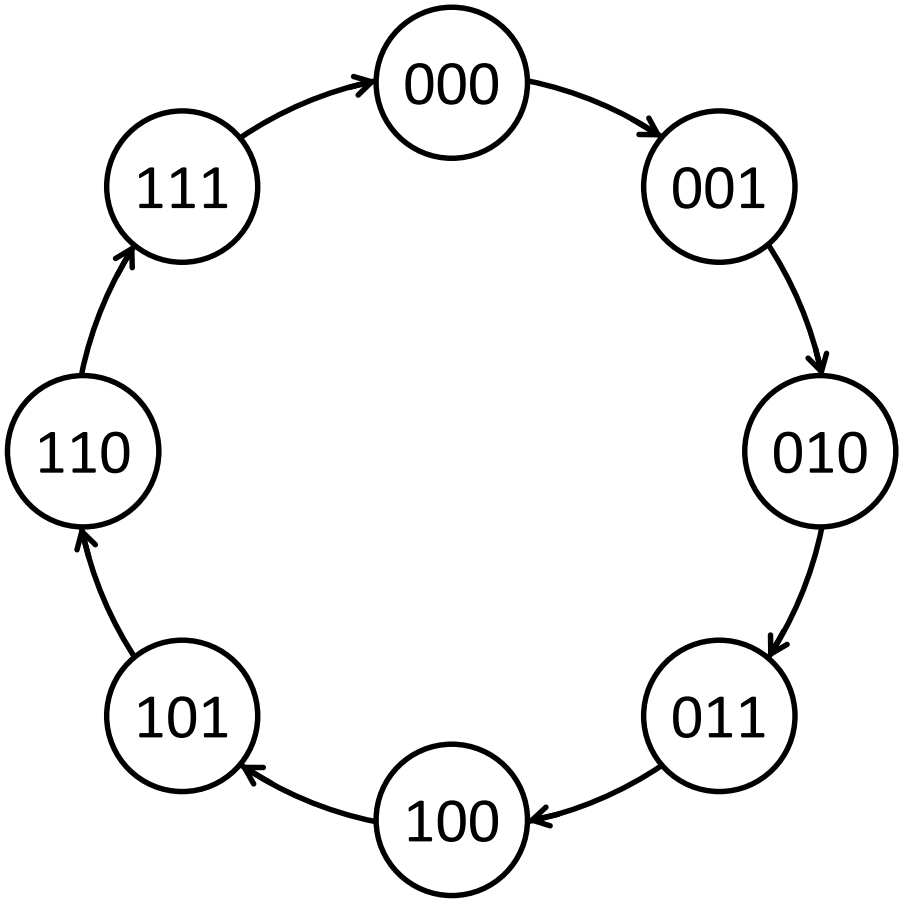
Three Flip-Flops are required for the eight states

Each state is assigned a unique binary count value

State Table

Only two columns: Present State and Next State

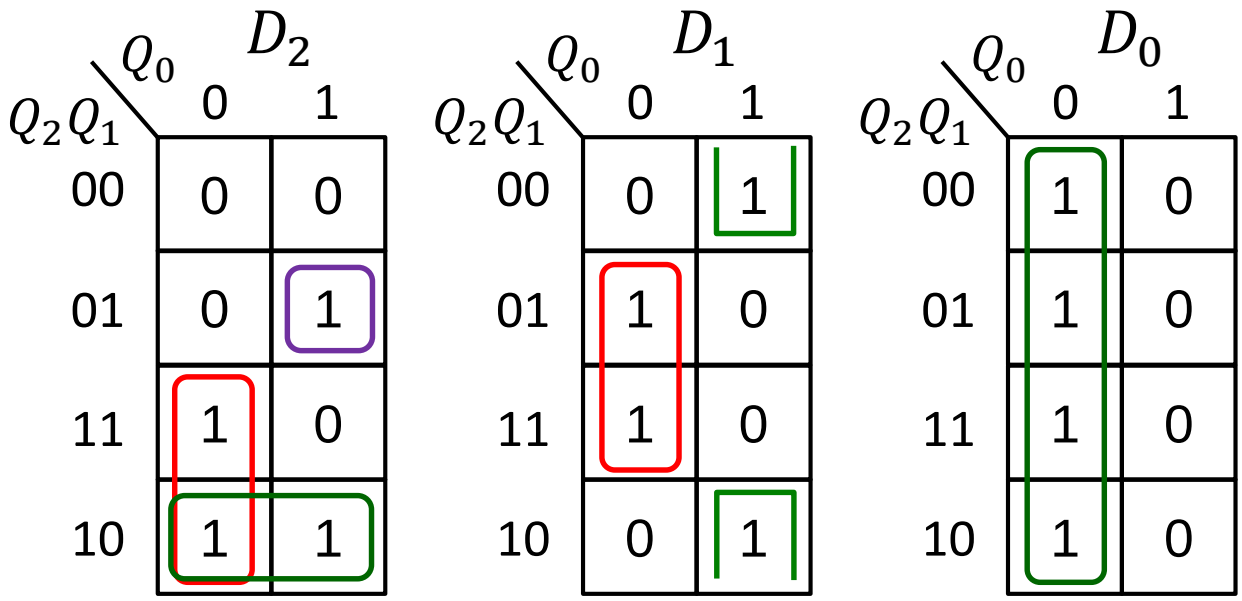
State changes each cycle



Present State Q_2 Q_1 Q_0	Next State D_2 D_1 D_0
0 0 0	0 0 1
0 0 1	0 1 0
0 1 0	0 1 1
0 1 1	1 0 0
1 0 0	1 0 1
1 0 1	1 1 0
1 1 0	1 1 1
1 1 1	0 0 0

Deriving the Next State Equations

Present State $Q_2\ Q_1\ Q_0$	Next State $D_2\ D_1\ D_0$
0 0 0	0 0 1
0 0 1	0 1 0
0 1 0	0 1 1
0 1 1	1 0 0
1 0 0	1 0 1
1 0 1	1 1 0
1 1 0	1 1 1
1 1 1	0 0 0



$$D_2 = Q_2Q_1' + Q_2Q_0' + Q_2'Q_1Q_0$$

$$D_2 = Q_2(Q_1' + Q_0') + Q_2'Q_1Q_0$$

$$D_2 = Q_2(Q_1Q_0)' + Q_2'(Q_1Q_0) = Q_2 \oplus (Q_1Q_0)$$

$$D_1 = Q_1Q_0' + Q_1'Q_0 = Q_1 \oplus Q_0$$

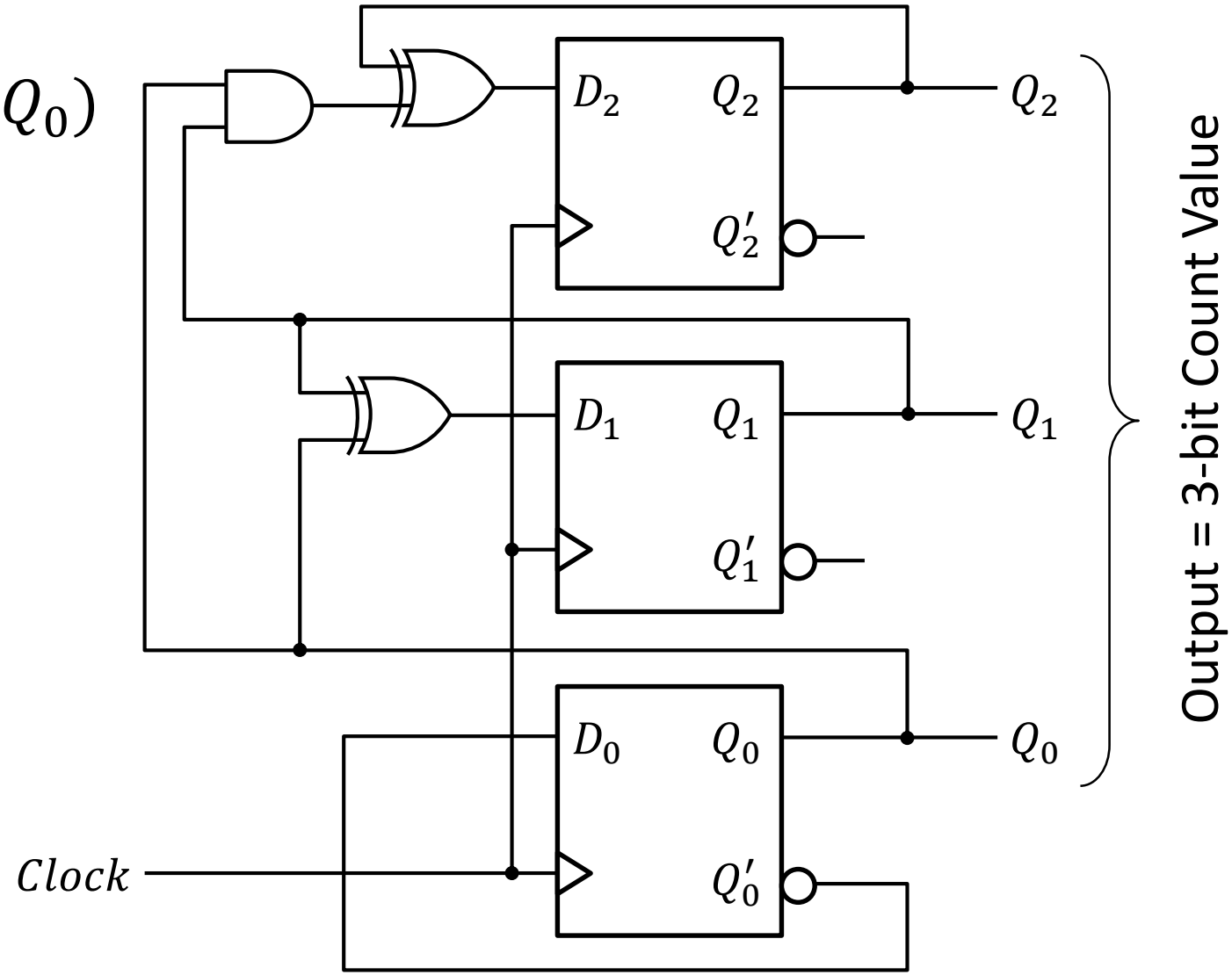
$$D_0 = Q_0'$$

3-Bit Counter Circuit Diagram

$$D_2 = Q_2 \oplus (Q_1 Q_0)$$

$$D_1 = Q_1 \oplus Q_0$$

$$D_0 = Q_0'$$



Up/Down Counter with Enable

Problem Specification:

- Design a 2-bit Up / Down counter
- Two inputs: E (Enable) and U (Up)

  If E = 0 then counter remains in the same state (regardless of U)

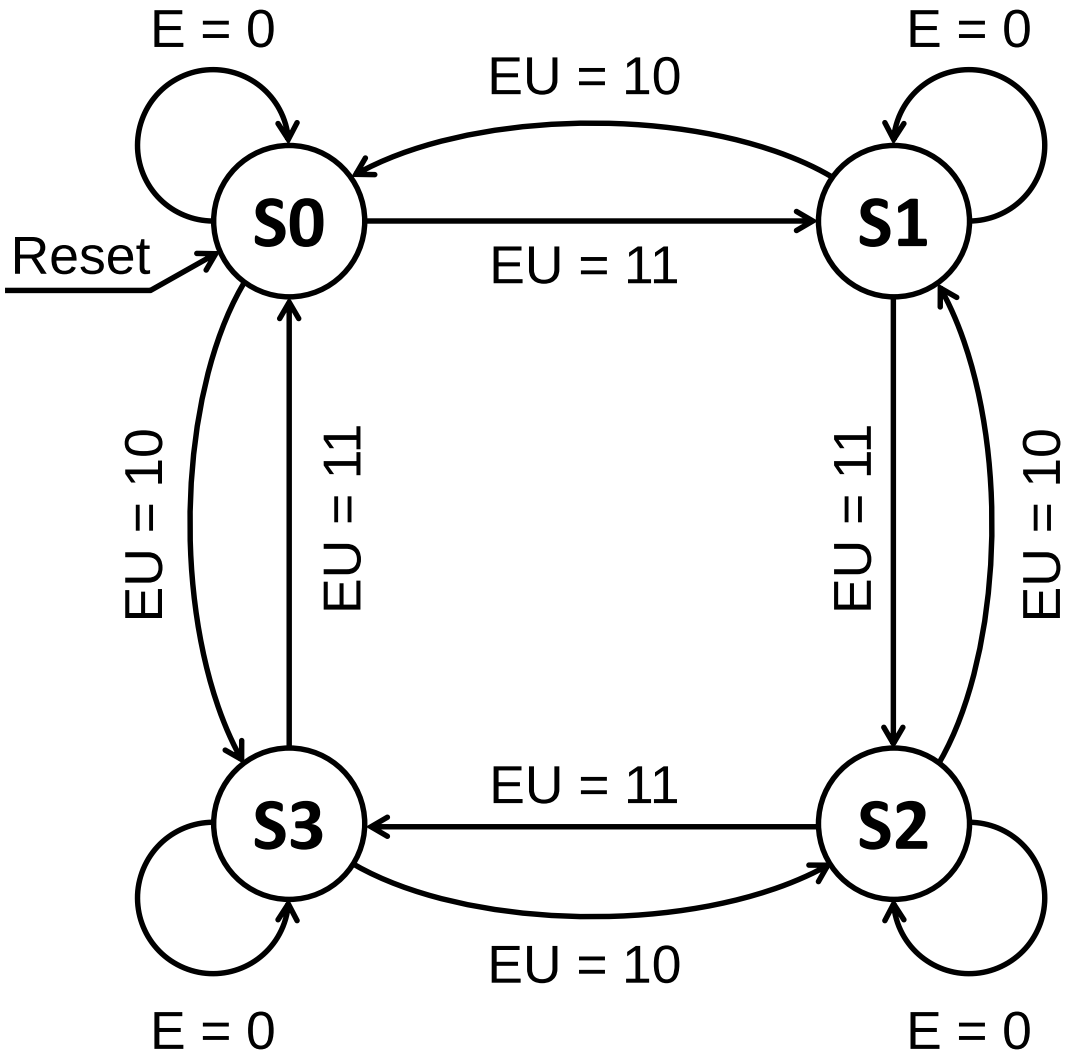
  If EU = 11 then count up from 0 to 3, then back to 0

  If EU = 10 then count down from 3 down to 0, then back to 3
- The output of the counter is the present state (count value)
- The circuit should be designed using T Flip-Flops
- A reset signal resets the counter to its initial state

Designing the State Diagram

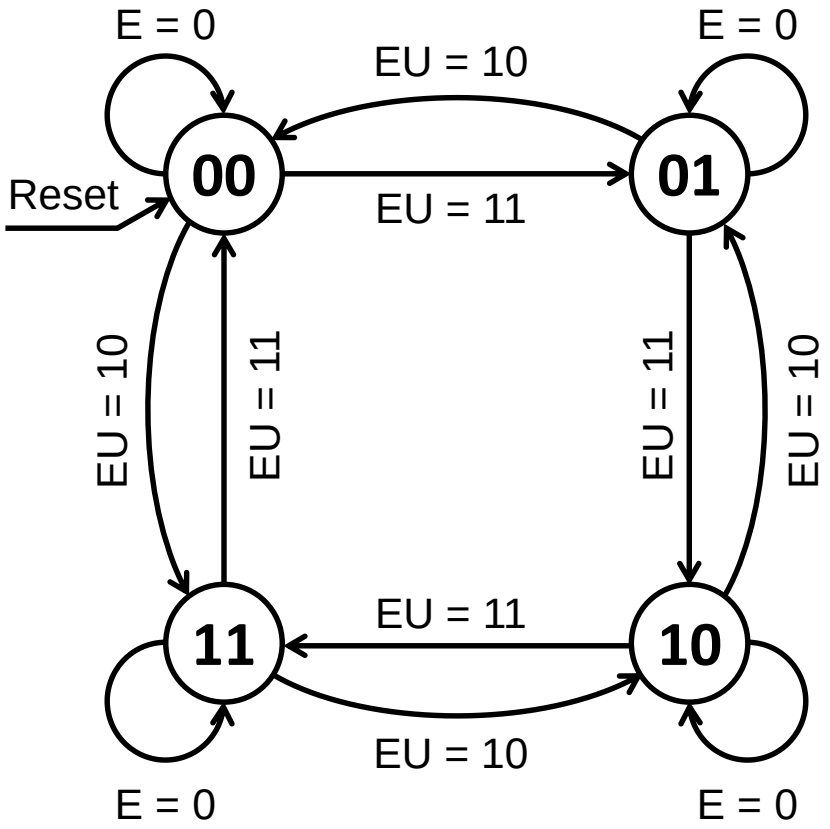
- Four states are required to store the count 0 to 3
- Count up if EU = 11
- Count down if EU = 10
- Disable counter if E = 0

  Transition to same state if EU = 00 or 01
- Asynchronous reset to start initially at state S0



State Assignment and State Table

- Four States ➔ Two State variables (2 Flip-Flops)
- State Assignment: S0 = 00, S1 = 01, S2 = 10, and S3 = 11



Present State Q_1 Q_0	Next State E U = 0 0	E U = 0 1	E U = 1 0	E U = 1 1
0 0	0 0	0 0	1 1	0 1
0 1	0 1	0 1	0 0	1 0
1 0	1 0	1 0	0 1	1 1
1 1	1 1	1 1	1 0	0 0

Excitation Table for Flip-Flops

- Excitation Table:

  Lists the required input for next state transition

- For D Flip-Flop: Input D = Next State $Q(t+1)$
- For T Flip-Flop: Input $T = Q(t) \oplus Q(t+1)$
- Excitation tables are used to find the Flip-Flop input equations

Excitation Tables for the D and T Flip-Flops

$Q(t)$	$Q(t+1)$	D
0	0	0
0	1	1
1	0	0
1	1	1

$Q(t)$	$Q(t+1)$	T
0	0	0
0	1	1
1	0	1
1	1	0

Deriving the T-Flip-Flop Input Equations

Present State Q_1 Q_0	Next State at (t+1) E U = 0 0	E U = 0 1	E U = 1 0	E U = 1 1
0 0	0 0	0 0	1 1	0 1
0 1	0 1	0 1	0 0	1 0
1 0	1 0	1 0	0 1	1 1
1 1	1 1	1 1	1 0	0 0

Present State Q_1 Q_0	Flip-Flop Inputs T_1 T_0 E U = 0 0	E U = 0 1	E U = 1 0	E U = 1 1
0 0	0 0	0 0	1 1	0 1
0 1	0 0	0 0	0 1	1 1
1 0	0 0	0 0	1 1	0 1
1 1	0 0	0 0	0 1	1 1

T_1

Q_1Q_0 \ EU	00	01	11	10
00	0	0	0	1
01	0	0	1	0
11	0	0	1	0
10	0	0	0	1

T_0

Q_1Q_0 \ EU	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	0	0	1	1
10	0	0	1	1

$$T_1 = Q_0EU + Q_0'EU'$$

$$T_1 = E(Q_0 \oplus U)'$$

$$T_0 = E$$

2-bit Up/Down Counter Circuit Diagram

$$T_1 = E(Q_0 \oplus U)'$$

$$T_0 = E$$