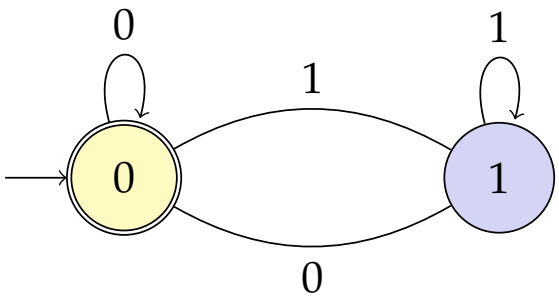
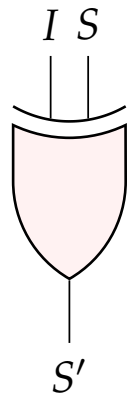
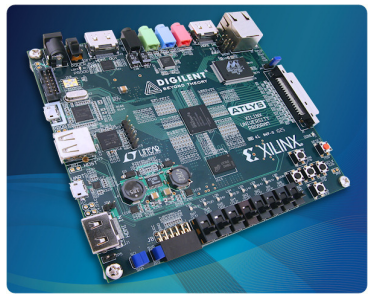
CS 226: Digital Logic Design

Lecture 7: Synchronous Sequential Logic

Ashutosh Trivedi







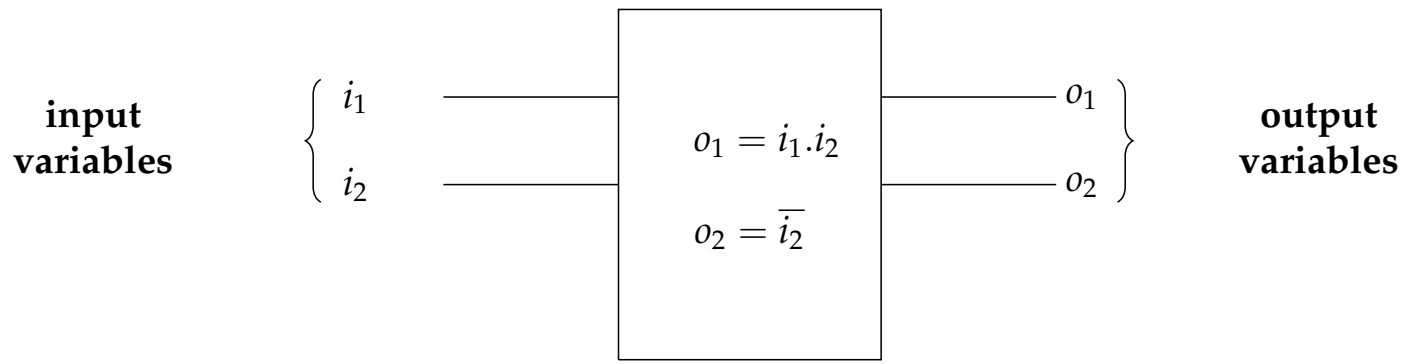
Department of Computer Science and **Engineering**,
Indian Institute of Technology Bombay.

Objectives

In this lecture we will introduce:

1. Synchronous Sequential circuits
2. Logic circuits that can store information (latches and flip-flops)
3. Analysis of sequential circuits
4. State reduction and Assignment
5. Design Procedure

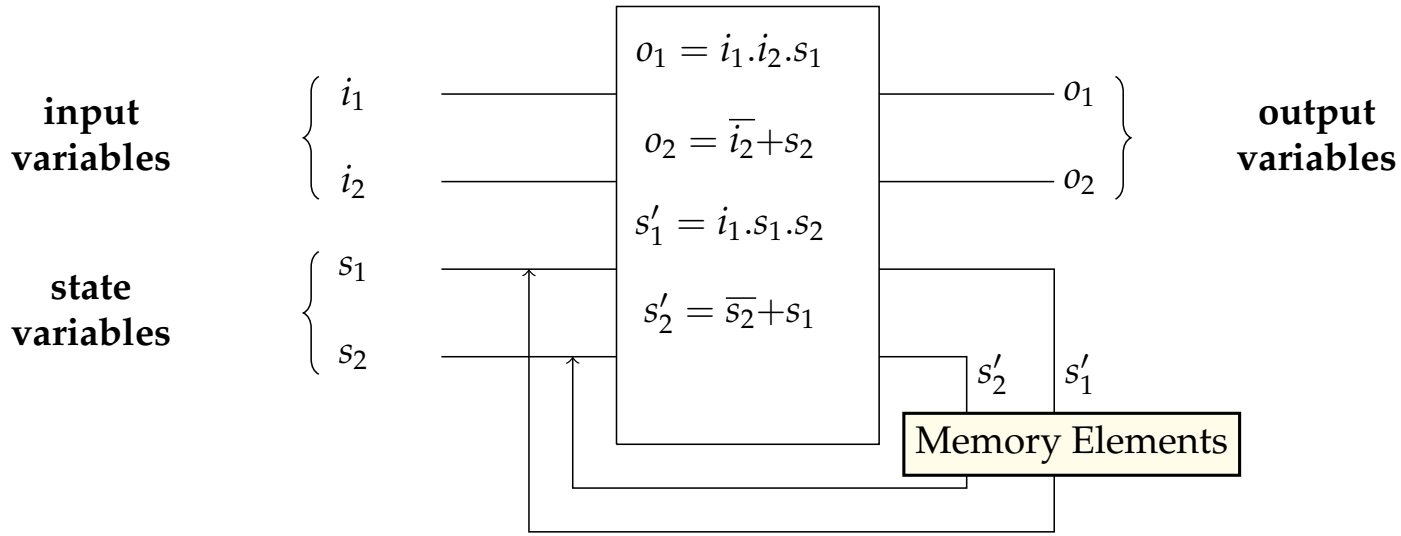
Combinational Circuits



Combinational circuits:

- Consists only of logic gates
- The output are determined by the present value of input
- Circuit behavior specified by a set of Boolean functions, Truth-tables, K-maps
- We have learned techniques to analyze and synthesize such circuits
- Examples: Adder, Multiplexers, Encoders, Decoders, etc.

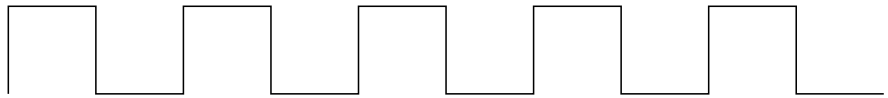
Sequential Circuits



Sequential Circuits:

- Consists of logic gates and storage elements
- The output are determined by the present value of the input and the state of the storage elements
- In effect, the output may depend on past values of the input via the state of the storage elements
- Circuit behavior specified by timed sequence of input and internal states via state machines

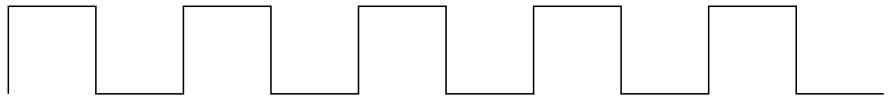
Classification of Sequential Circuits



Timing Diagram of a clock pulse

1. Synchronous sequential circuits:
   - Behavior defined from knowledge of signals at discrete instants of time.
   - Synchronization is achieved by a timing device called clock generator which provides a clock signal having periodic train of clock pulses.
   - Storage elements change state only at the arrival of the pulse.
   - Easy to design, however the performance (speed) depends on frequency of clock signal.

Classification of Sequential Circuits



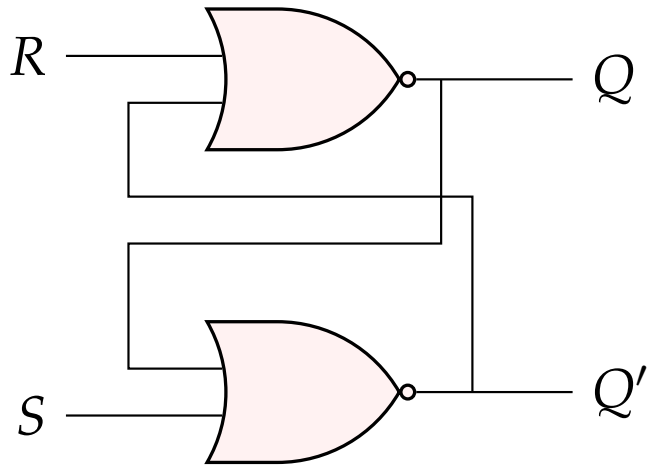
Timing Diagram of a clock pulse

1. Synchronous sequential circuits:
   - Behavior defined from knowledge of signals at discrete instants of time.
   - Synchronization is achieved by a timing device called clock generator which provides a clock signal having periodic train of clock pulses.
   - Storage elements change state only at the arrival of the pulse.
   - Easy to design, however the performance (speed) depends on frequency of clock signal.
2. Asynchronous sequential circuits:
   - The behavior depends upon the input signal at any instant of time and order in which input changes.
   - Asynchronous circuits do not use clock pulses, and the storage elements change states as soon as the input signal changes
   - Have better performance, but extremely difficult to design correctly.

Memory Elements

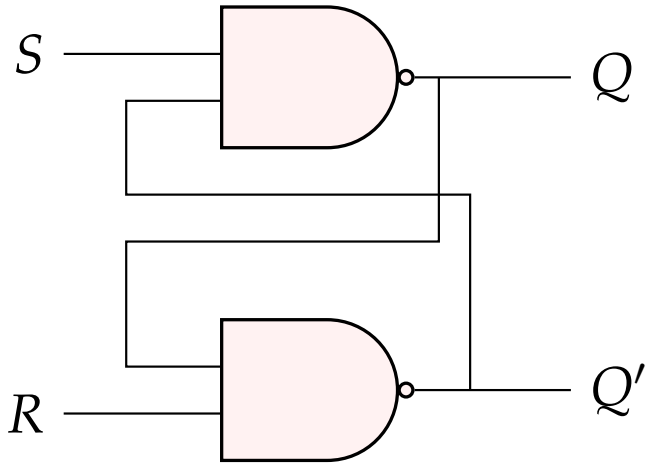
- Can maintain a binary state indefinitely (as long as circuit is powered)
- Memory elements can be considered as the most basic form of sequential circuits.
- Major difference between various memory elements are in the number of inputs and how the input affects the binary state.
- We will next introduce the following memory elements:
  - Latches (operate with signal levels, aka level-sensitive devices)
    - SR (set-reset) latches
    - D (data) latches
  - Flip-flops (operate by clock transitions, aka edge-sensitive devices)
    - Master-Slave flip-flops
    - Edge-triggered flip-flops
    - JK flip-flop
    - T (toggle) flip-flop
  - RAM and ROM — bulk memory elements (will discuss in another lecture)

Basic Latches with NOR Gates – SR Latches



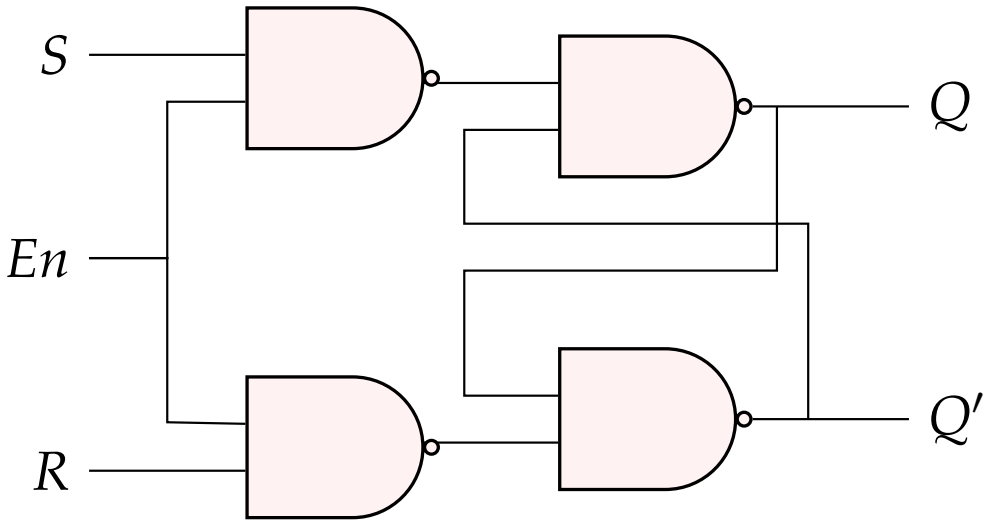
S	R	Q	Q'	
1	0	1	0	
0	0	1	0	(after $S = 1$ and $R = 0$)
0	1	0	1	
0	0	0	1	(after $S = 0$ and $R = 1$)
1	1	0	0	(forbidden)

Basic Latches with NAND Gates – $\overline{S}.\overline{R}$ Latches



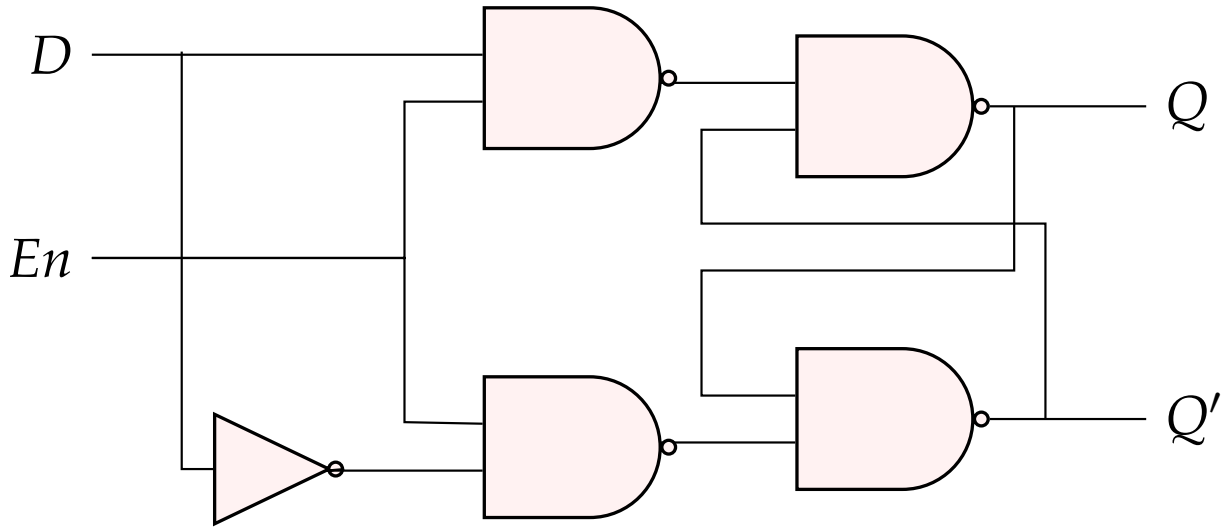
S	R	Q	Q'	
1	0	0	1	
1	1	0	1	(after $S = 1$ and $R = 0$)
0	1	1	0	
1	1	1	0	(after $S = 0$ and $R = 1$)
0	0	1	1	(forbidden)

SR Latches with Control Input



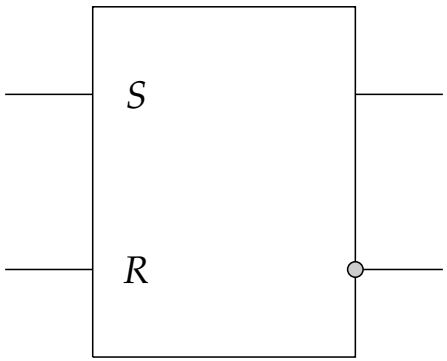
En	S	R	Next State of Q
0	X	X	No Change
1	0	0	No Change
1	0	1	$Q = 0$; Reset State
1	1	0	$Q = 1$; Set State
1	1	1	Indeterminate

D Latches

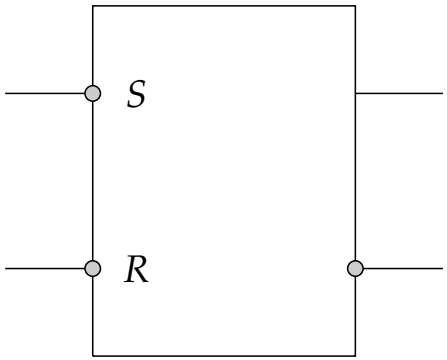


En	D	Next State of Q
0	X	No Change
1	0	$Q = 0$; Reset State
1	1	$Q = 1$; Set State

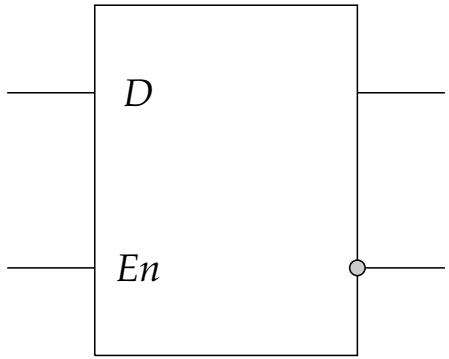
Graphical Symbols for Latches



SR

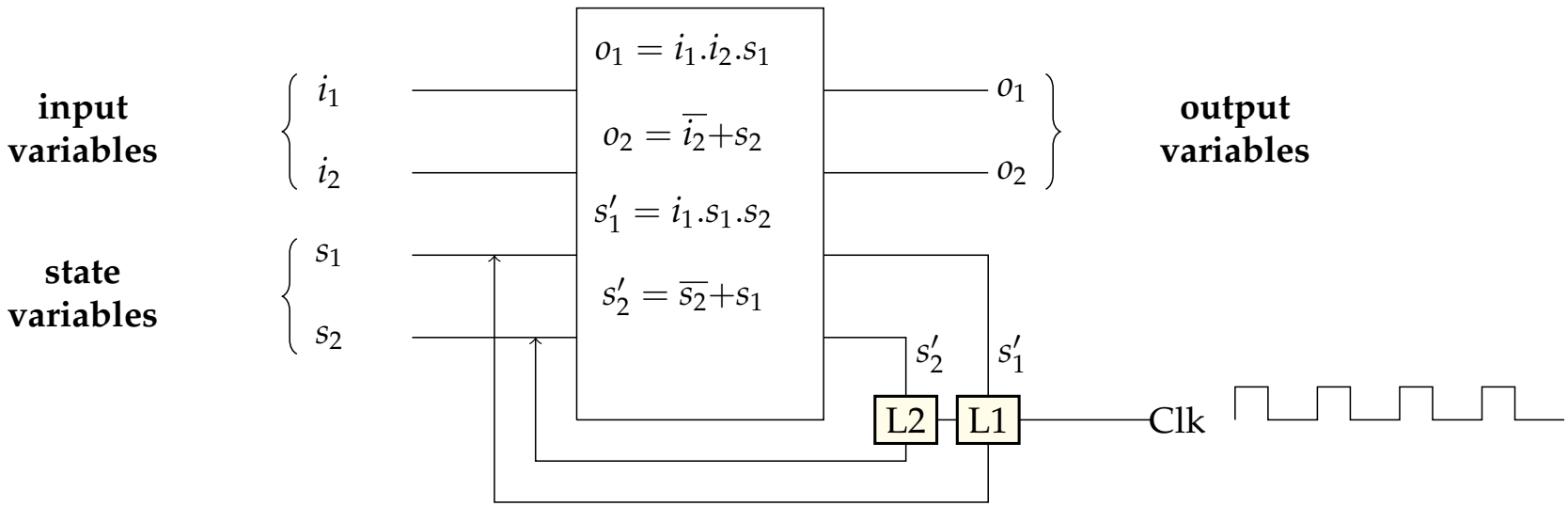


$\overline{S}.\overline{R}$



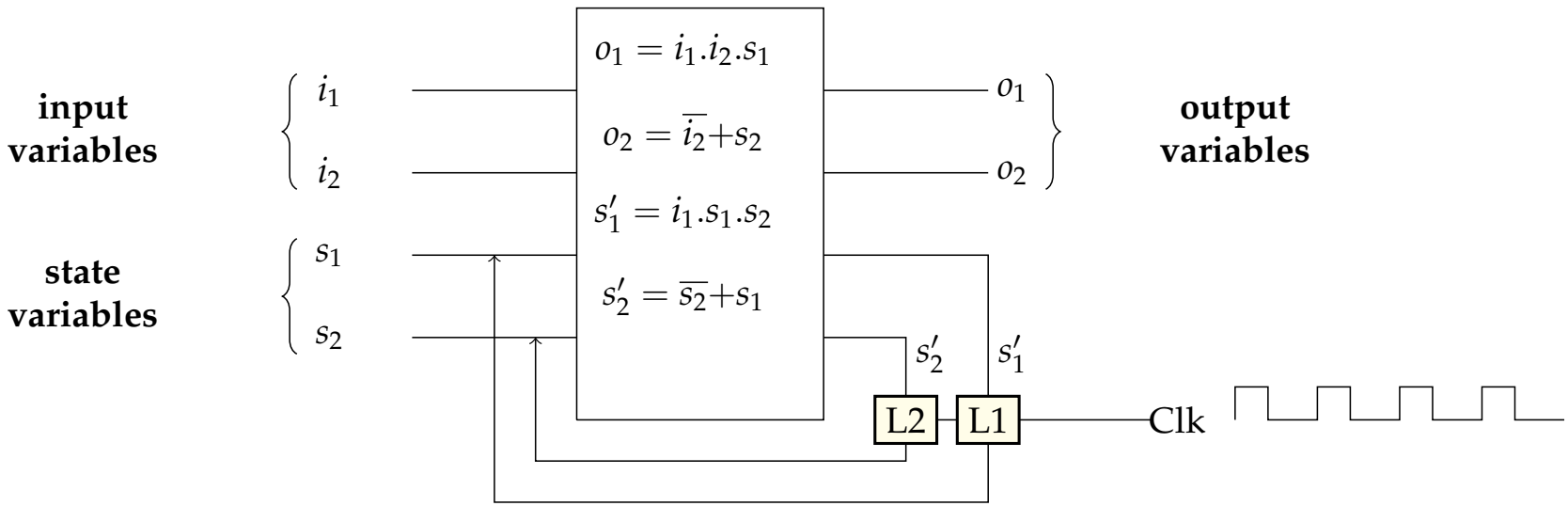
D

Problem with Latches



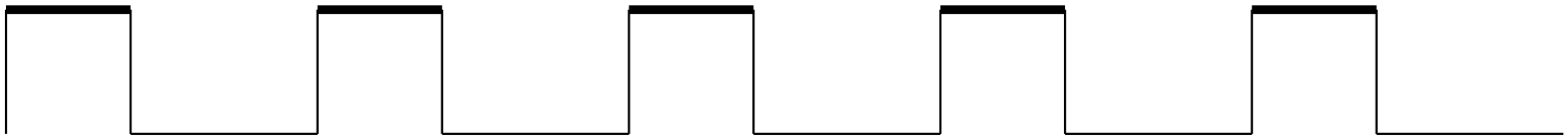
- Latches operate with signal level
- Feedback path in sequential circuits
- Unpredictable situation since the states may keep on changing as long as clock pulse is high! Why?

Problem with Latches

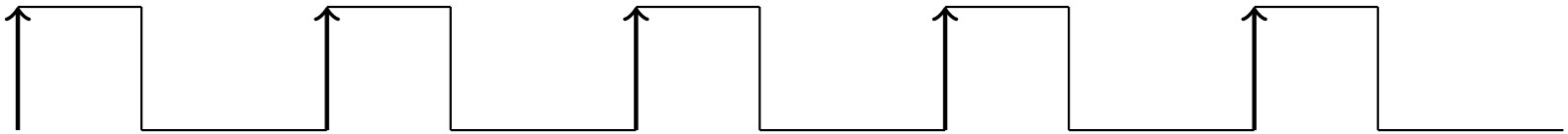


- Latches operate with signal level
- Feedback path in sequential circuits
- Unpredictable situation since the states may keep on changing as long as clock pulse is high! Why?
- **Solution**:
  Force latches to change their state only once during a clock cycle!

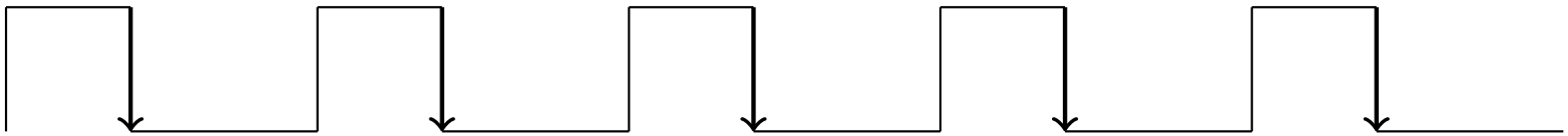
Clock response in storage elements



(a) Response to positive level

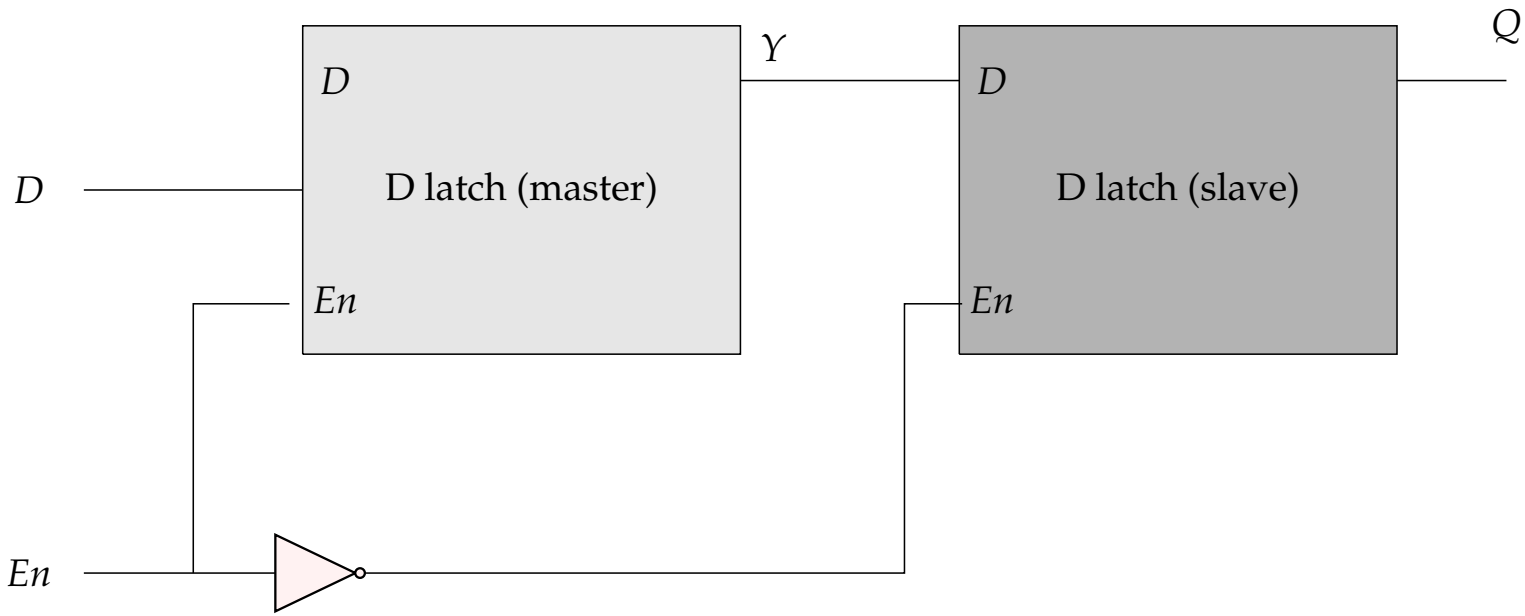


(b) Positive-Edge Response



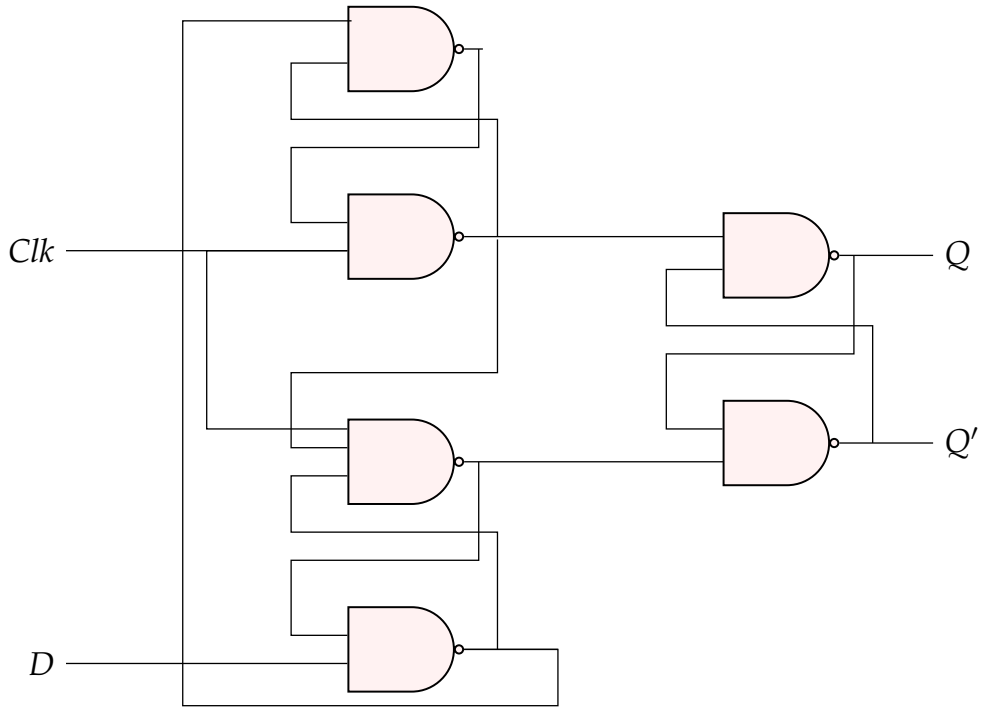
(a) Negative-Edge Response

Master-Slave D Flip-flop



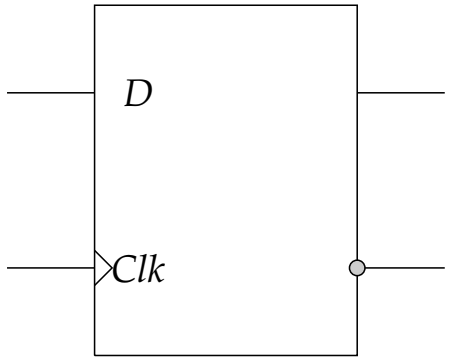
- Notice at what time input value is transferred
- Is it positive-edge triggered. or negative edge-triggered?

D-type Positive-Edge-Triggered Flip-Flop

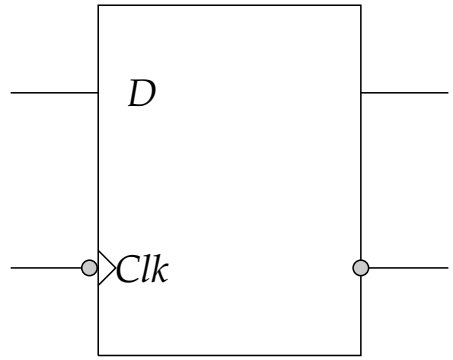


- Made with three SR latches
- Consider the following three cases:
  - When Clk is 0
  - When Clk goes to 1, while D is at 0 (What if D changes when clk is still high)
  - When En goes to 1, while D is at 1 (What if D changes when clk is still high)
- setup and hold times

Graphical Symbols for Flip-flops

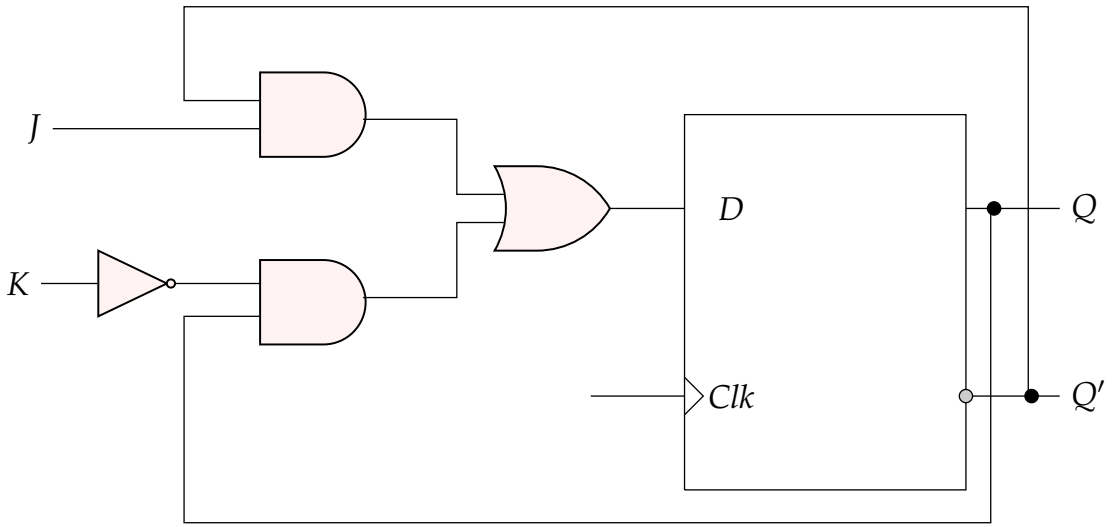


a) Positive-Edge Triggered

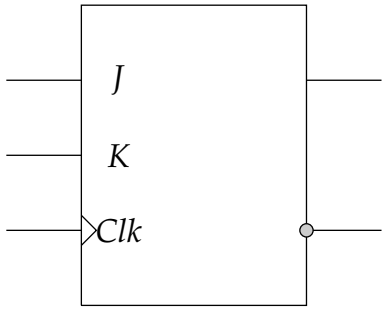


b) Negative-Edge Triggered

J-K Flip-Flop



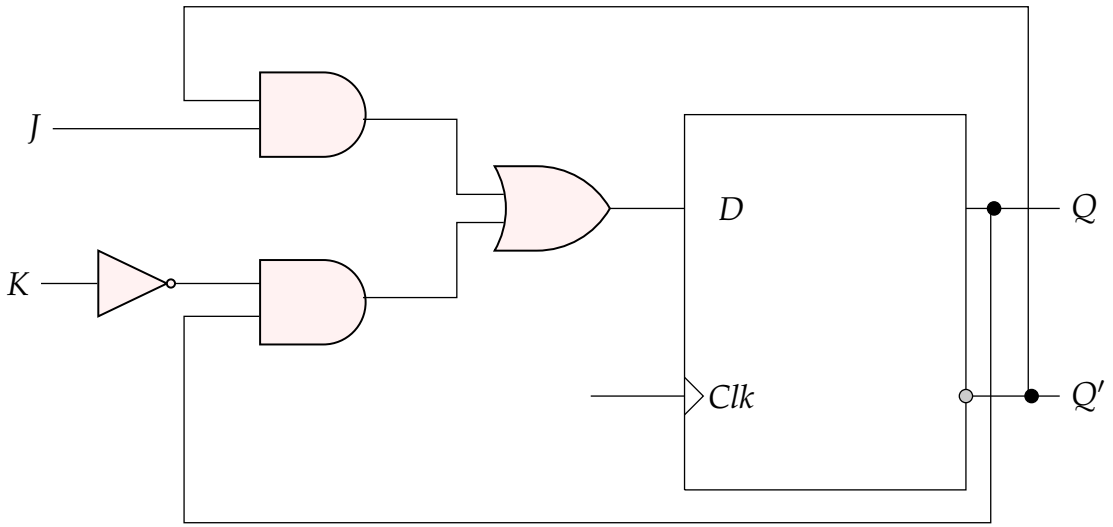
a) Circuit Diagram



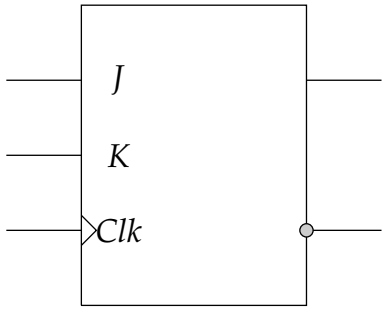
b) Graphical symbol

- Three functions: set, reset, and complement
- Construct form *D*-flip-flops
- Two inputs *J* and *K* such that *J* sets, *K* resets, both complement, while both low keep the old value. How?

J-K Flip-Flop



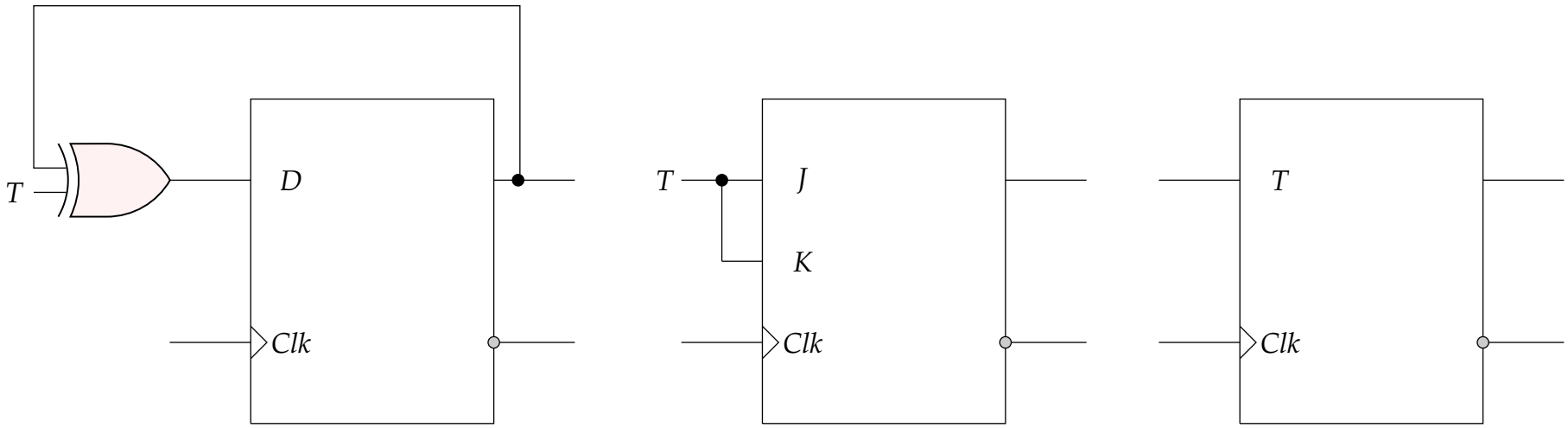
a) Circuit Diagram



b) Graphical symbol

- Three functions: set, reset, and complement
- Construct form D-flip-flops
- Two inputs J and K such that J sets, K resets, both complement, while both low keep the old value. How?
- $D = JQ' + K'Q$.

T Flip-Flop



a) From D-Flip-flop b) From J-K flip-flop c) Graphical symbol

- Single input T that toggles (complements) the state
- Can be constructed from both D flip-flop or J-K flip-flop
- $D = T \oplus Q$
- $J = K = T$.

Characteristic Table

- A characteristic table defines the logical properties of a flip-flop by describing its operation in tabular form.
- J-K Flip-flop

J	K	$Q(t+1)$	
0	0	$Q(t)$	No Change
0	1	0	Reset
1	0	1	Set
1	1	$\overline{Q(t)}$	Complement

- D Flip-flop

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

- T Flip-flop

T	$Q(t+1)$	
0	$Q(t)$	Reset
1	$\overline{Q(t)}$	Set

Characteristic Equations

- The logical properties of a filp-flop can be described algebraically with a characteristic equation.
- The characteristic equation for J-K Flip-flop is

$$Q(t+1) = J.\overline{Q} + \overline{K}.Q$$

- The characteristic equation for D Flip-flop is

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

- The characteristic equation for T Flip-flop is

$$Q(t+1) = Q \cdot T + \overline{Q} \cdot \overline{T} = Q \oplus T$$

Objectives

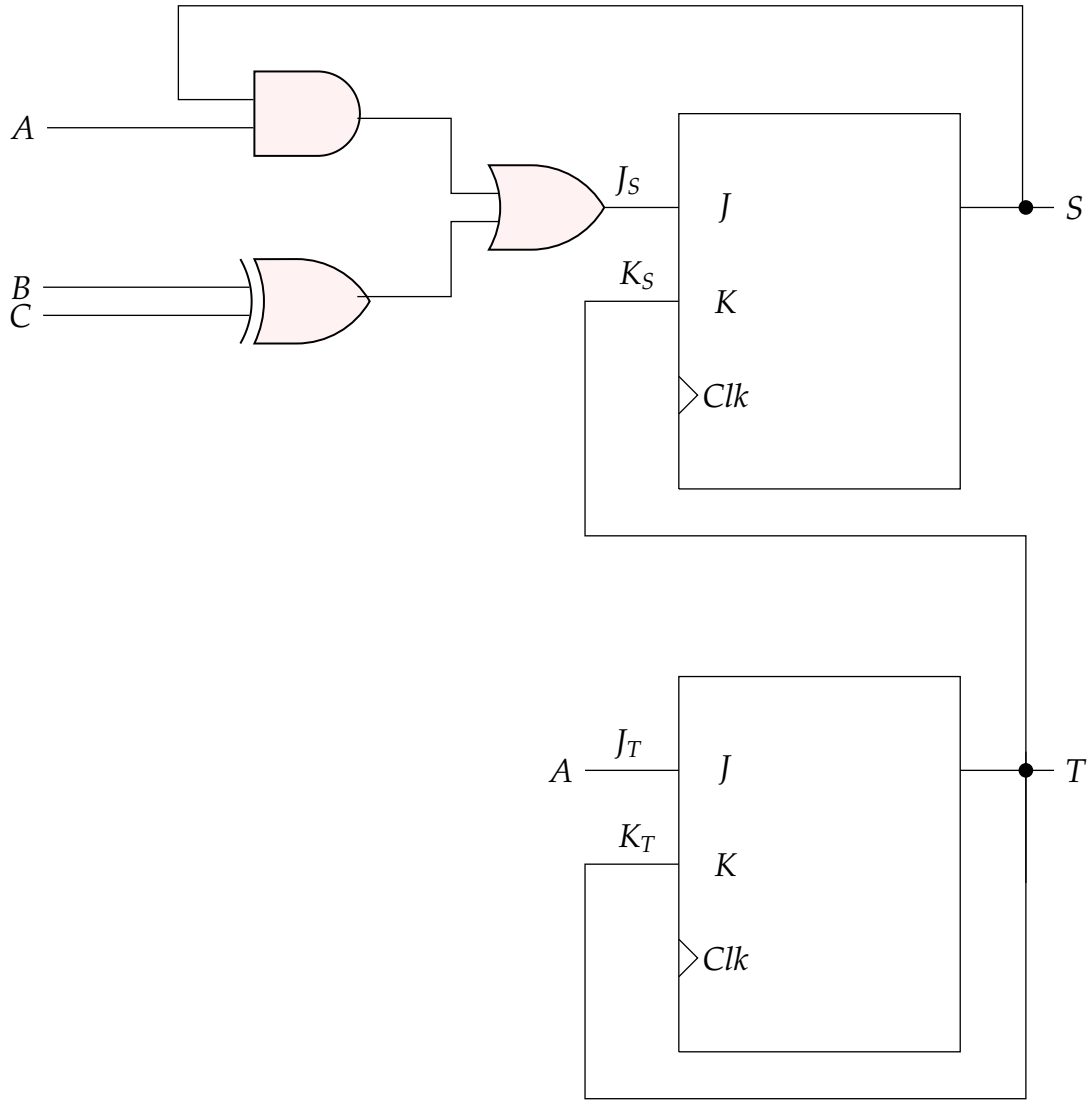
Synchronous Sequential Circuits

Storage Elements

Analysis of Clocked Sequential Circuits

Synthesis of Clocked Sequential Circuits

Analysis of Sequential Circuits



Representation of Sequential Circuits

How do we specify a combinational circuits?

Representation of Sequential Circuits

How do we specify a combinational circuits?

Output variables as a function of input variables, using

1. natural language,
2. logic gates,
3. Boolean expressions, and
4. Truth-tables or K-maps.

Representation of Sequential Circuits

How do we specify a combinational circuits?

Output variables as a function of input variables, using

1. natural language,
2. logic gates,
3. Boolean expressions, and
4. Truth-tables or K-maps.

How do we specify a clocked sequential circuits?

Representation of Sequential Circuits

How do we specify a combinational circuits?

Output variables as a function of input variables, using

1. natural language,
2. logic gates,
3. Boolean expressions, and
4. Truth-tables or K-maps.

How do we specify a clocked sequential circuits?

Output variables as a function of input variables, using

1. natural language Ok.

Representation of Sequential Circuits

How do we specify a combinational circuits?

Output variables as a function of input variables, using

1. natural language,
2. logic gates,
3. Boolean expressions, and
4. Truth-tables or K-maps.

How do we specify a clocked sequential circuits?

Output variables as a function of input variables, using

1. natural language — Ok.
2. logic gates — Ok.

Representation of Sequential Circuits

How do we specify a combinational circuits?

Output variables as a function of input variables, using

1. natural language,
2. logic gates,
3. Boolean expressions, and
4. Truth-tables or K-maps.

How do we specify a clocked sequential circuits?

Output variables as a function of input variables, using

1. natural language — Ok.
2. logic gates — Ok.
3. Boolean expressions, — Wait a minute!

Representation of Sequential Circuits

How do we specify a combinational circuits?

Output variables as a function of input variables, using

1. natural language,
2. logic gates,
3. Boolean expressions, and
4. Truth-tables or K-maps.

How do we specify a clocked sequential circuits?

Output variables as a function of input variables, using

1. natural language — Ok.
2. logic gates — Ok.
3. Boolean expressions, — Wait a minute!
4. Truth-tables or K-maps.

Clocked Sequential Circuits

How to express clocked sequential circuits using Boolean expressions.

- How clocked sequential circuits are different?

Clocked Sequential Circuits

How to express clocked sequential circuits using Boolean expressions.

- How clocked sequential circuits are different?
  - due to storage elements that introduce state variables

Clocked Sequential Circuits

How to express clocked sequential circuits using Boolean expressions.

- How clocked sequential circuits are different?
  - due to storage elements that introduce state variables
  - state variables act both as input and output variables

Clocked Sequential Circuits

How to express clocked sequential circuits using Boolean expressions.

- How clocked sequential circuits are different?
  - due to storage elements that introduce state variables
  - state variables act both as input and output variables
  - output variables may depend on both the input variables and current value of state variables

Clocked Sequential Circuits

How to express clocked sequential circuits using Boolean expressions.

- How clocked sequential circuits are different?
  - due to storage elements that introduce state variables
  - state variables act both as input and output variables
  - output variables may depend on both the input variables and current value of state variables
  - The next value of state variables may also depend on input variable and the current value of the state variable

Clocked Sequential Circuits

How to express clocked sequential circuits using Boolean expressions.

- How clocked sequential circuits are different?
  - due to storage elements that introduce state variables
  - state variables act both as input and output variables
  - output variables may depend on both the input variables and current value of state variables
  - The next value of state variables may also depend on input variable and the current value of the state variable
- Solution:
  - We need two copies of a state variable A— present-state variable $A(t)$ representing the present value of A and next-state variable $A(t+1)$ representing the value of A one clock-edge later
  - Specify output variables and next-state variables as function of input variables and present-state variables.

Clocked Sequential Circuits: State Equations

- State Equations or transition equations

$$\begin{aligned} A(t+1) &= A(t) \cdot x(t) + B(t) \cdot x(t) \\ B(t+1) &= \overline{A(t)} \cdot x(t) \\ y(t) &= (A(t) + B(t)) \cdot \overline{x(t)} \end{aligned}$$

- To simplify, we omit (t) from the right hand side, and write:

$$\begin{aligned} A(t+1) &= A \cdot x + B \cdot x \\ B(t+1) &= \overline{A} \cdot x \\ y(t) &= (A + B) \cdot \overline{x} \end{aligned}$$

Clocked Sequential Circuits: State Tables

Present State		*Input*	*Next State*		*Output*
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

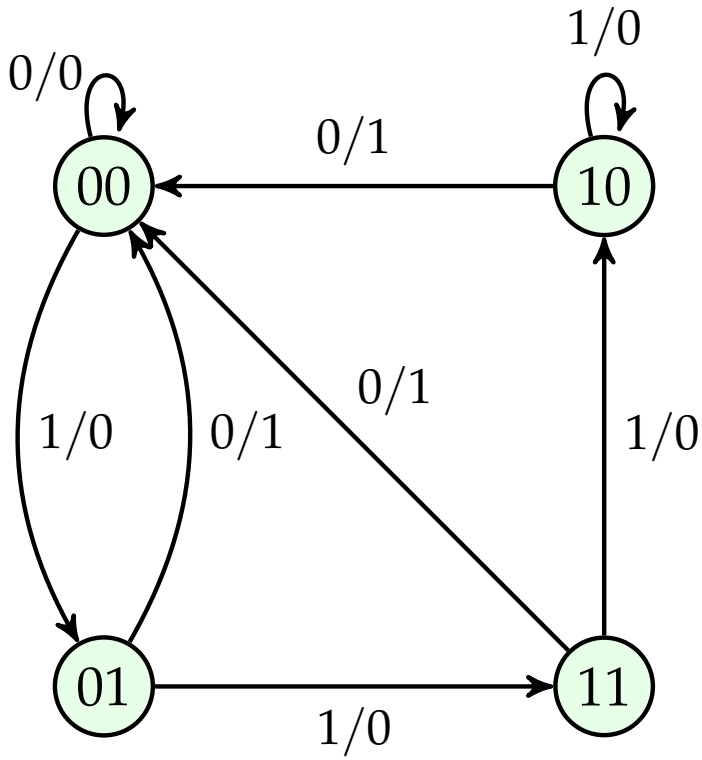
State table for $A(t+1) = Ax + Bx$, $B(t+1) = \overline{A}x$ and $y = A\overline{x} + B\overline{x}$.

Clocked Sequential Circuits: State Tables

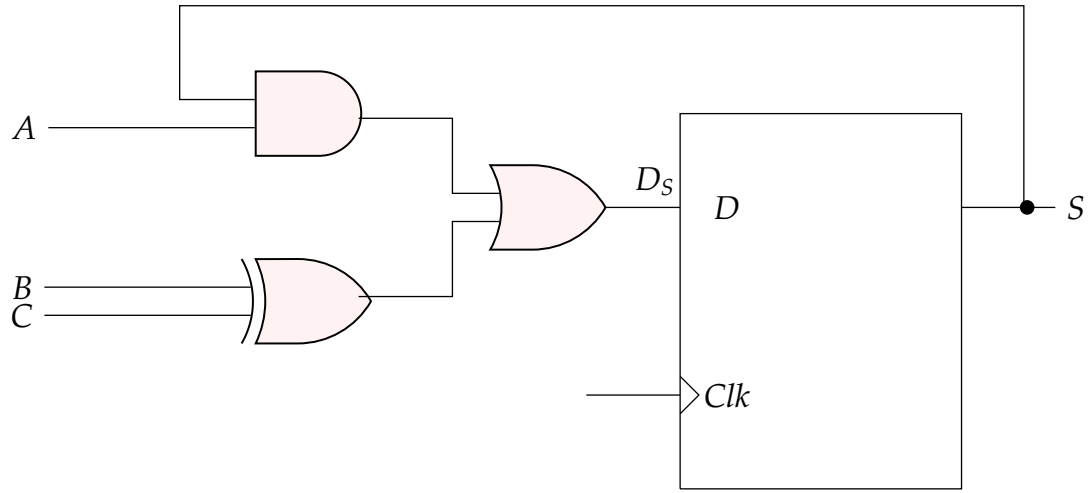
Present State		*NextState*				*Output*	
		$x = 0$		$x = 1$		$x = 0$	$x = 1$
A	B	A	B	A	B	y	y
0	0	0	0	0	1	0	0
0	1	0	0	1	1	1	0
1	0	0	0	1	0	1	0
1	1	0	0	1	0	1	0

State table for $A(t+1) = Ax + Bx$, $B(t+1) = \overline{A}x$ and $y = A\overline{x} + B\overline{x}$.

Clocked Sequential Circuits: State Diagram



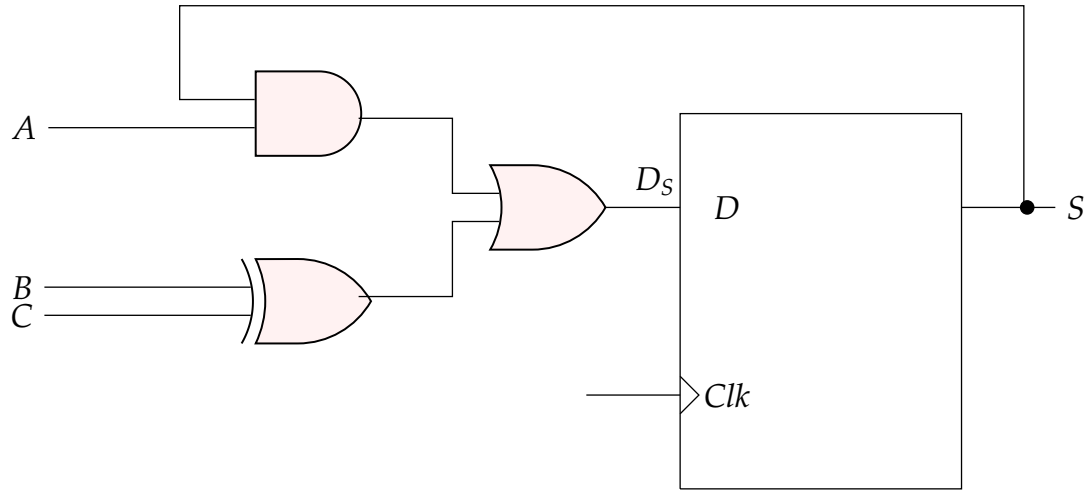
Clocked Sequential Circuits: Input Equations



- We call D_Q the D-input of a flip-flop whose output is labeled with symbol Q.
- The input equation for D_S is

$$D_S = A \cdot Q + (B \oplus C)$$

Clocked Sequential Circuits: Input Equations



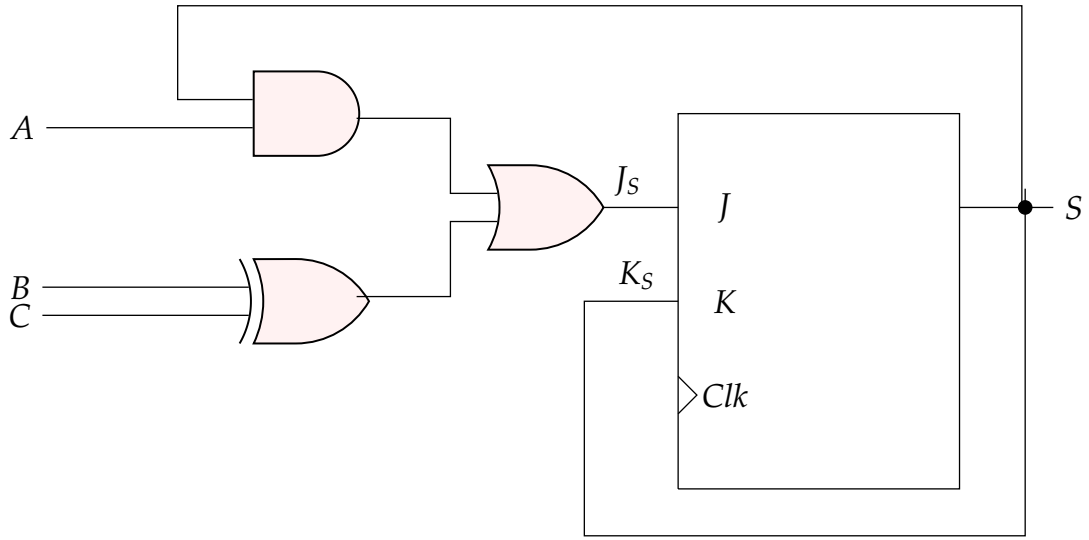
- We call D_Q the D-input of a flip-flop whose output is labeled with symbol Q.
- The input equation for D_S is

$$D_S = A \cdot Q + (B \oplus C)$$

- Combining the input equation with the characteristic equation of the flip-flop will give the next-state equation. Consider

$$S(t+1) = D_S = A \cdot S + (B \oplus C).$$

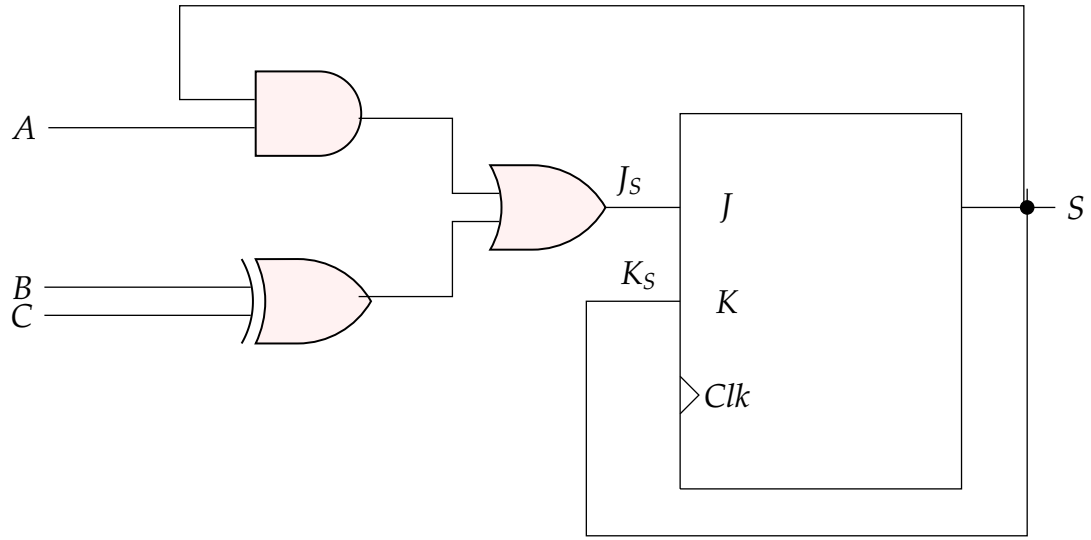
Clocked Sequential Circuits: Input Equations



- Here the inputs are
- The input equation for J_S and K_S are:

$$\begin{aligned} J_S &= A \cdot S + (B \oplus C) \\ K_S &= S. \end{aligned}$$

Clocked Sequential Circuits: Input Equations



- Here the inputs are
- The input equation for J_S and K_S are:

$$\begin{aligned} J_S &= A \cdot S + (B \oplus C) \\ K_S &= S. \end{aligned}$$

- Combining the input equation with the characteristic equation of the flip-flop will give the next-state equation. Consider

$$\begin{aligned} S(t+1) &= J_S.\overline{S} + \overline{K_S}.S \\ &= (A \cdot S + (B \oplus C)) \cdot \overline{S} + \overline{S} \cdot S. \end{aligned}$$

Exercise

Given a gate description of a clocked synchronous circuit, construct the following:

- State equations
- State table
- State diagram

Synthesis of Sequential Circuits

The procedure for designing synchronous sequential circuits:

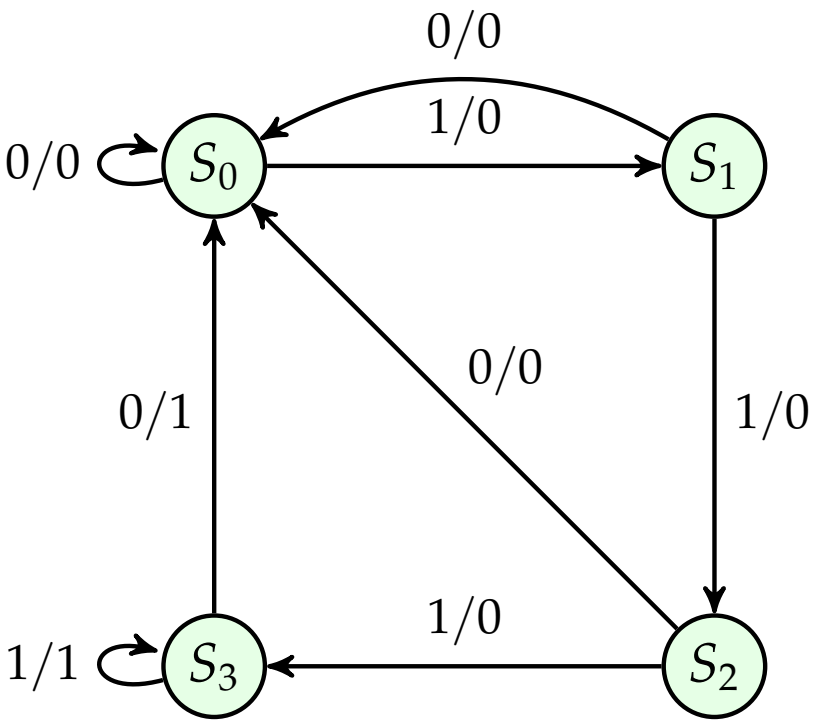
1. From the word description and specification of the desired operation, derive a state diagram for the circuit.
2. Reduce the number of states if necessary.
3. Assign binary values to the states.
4. Obtain the binary-coded state table.
5. Choose the type of flip-flops to be used.
6. Derive the simplified flip-flop input equations and output equations.
7. Draw the logic diagram.

Step 1: Deriving State Diagram

Derive state diagrams for the following problems:

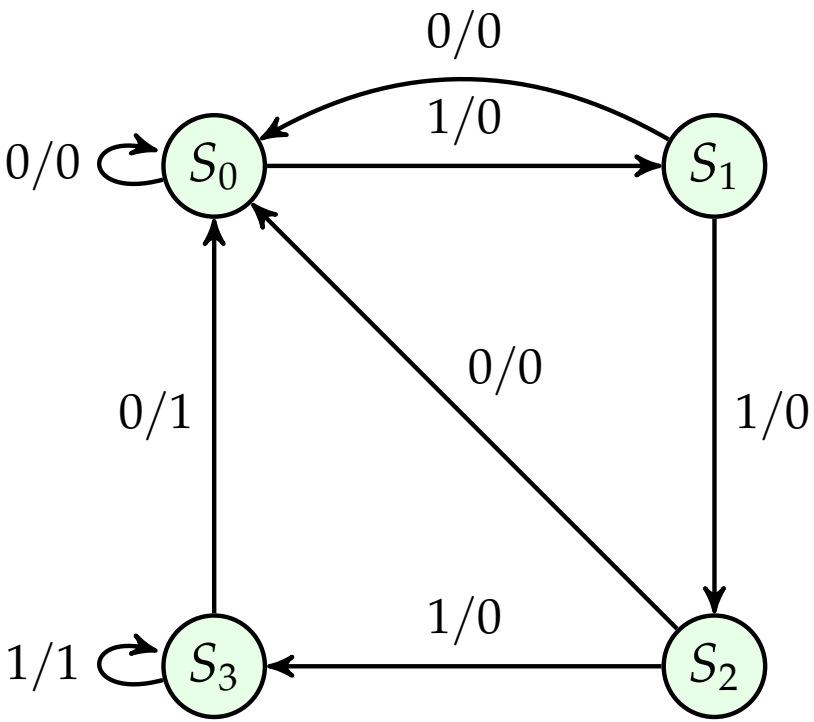
1. **Sequence Detector**: Design a circuit that detects a sequence of three of more consecutive 1's.
2. **Binary Counter**: Design a circuit for three-bit binary counter.
3. **Binary Counter**: Design a circuit for three-bit binary counter with one input w such that if $w = 0$ the count remains, and if $w = 1$ the count is incremented.
4. **Up-and-Down Binary Counter**: Design a circuit for three-bit binary counter with one input w such that if $w = 0$ the count is decremented, and if $w = 1$ the count is incremented.
5. **Even-Parity Checker**: Design a circuit for even-parity checker with two inputs w and *reset*, and one output z such that $z = 1$ if the number of times sigma w has been 1's since the previous reset is even.
6. Outputs 1 whenever the input bit sequence has exactly two 0s in the last three input bits.

Example 1: Sequence Detector

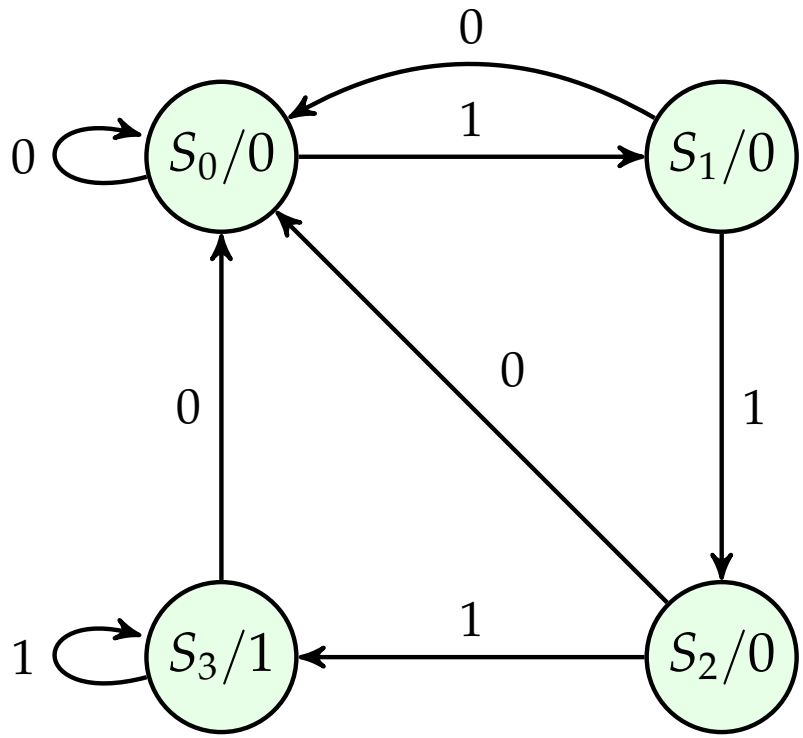


a) Mealy machine

Example 1: Sequence Detector



a) Mealy machine



b) Moore machine

Step 2: State Reduction

- State reduction is desirable since it may reduces the number of flip-flop required to implement the circuit.
- Two states are said to be equivalent if, for each member of the set of input, they give exactly same output and send the circuit either to same state or to an equivalent state.
- When two states are equivalent, one of them can be removed without changing the function of the circuit.
- It is difficult to tell whether two states are equivalent, but slightly easy to tell when they are not equivalent.
- Idea: Implication table. (Section 9.5)
- Examples.

Step 3: State Assignment

- Assign unique binary values to states

Step 3: State Assignment

- Assign unique binary values to states
- For n states we need at least $\lceil log_2(n) \rceil$ variables to encode the states.
- Three possible binary state assignments:

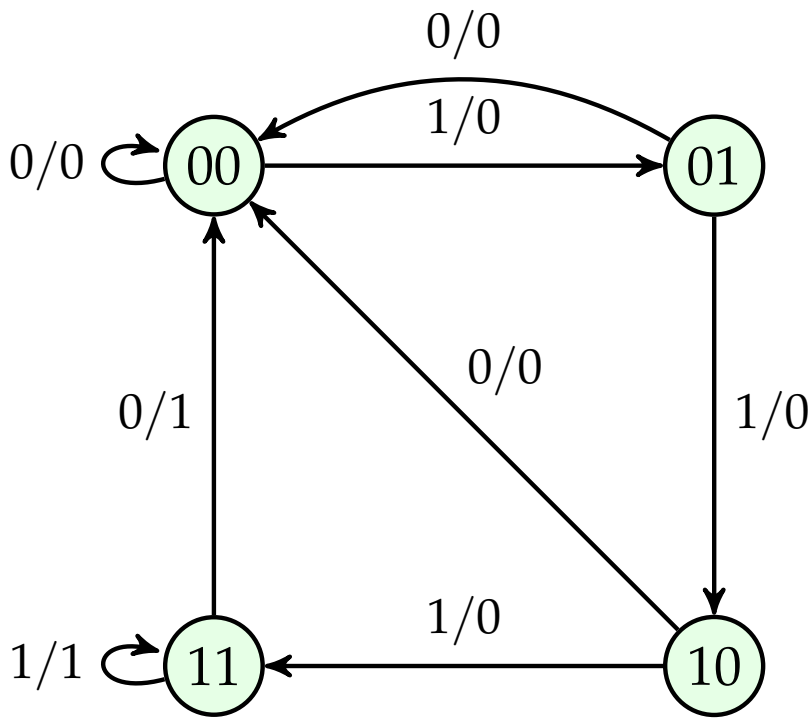
Step 3: State Assignment

- Assign unique binary values to states
- For n states we need at least $\lceil log_2(n) \rceil$ variables to encode the states.
- Three possible binary state assignments:

State	Binary	Gray Code	One-Hot
a	000	000	00001
b	001	001	00010
c	010	011	00100
d	011	010	01000
e	100	110	10000

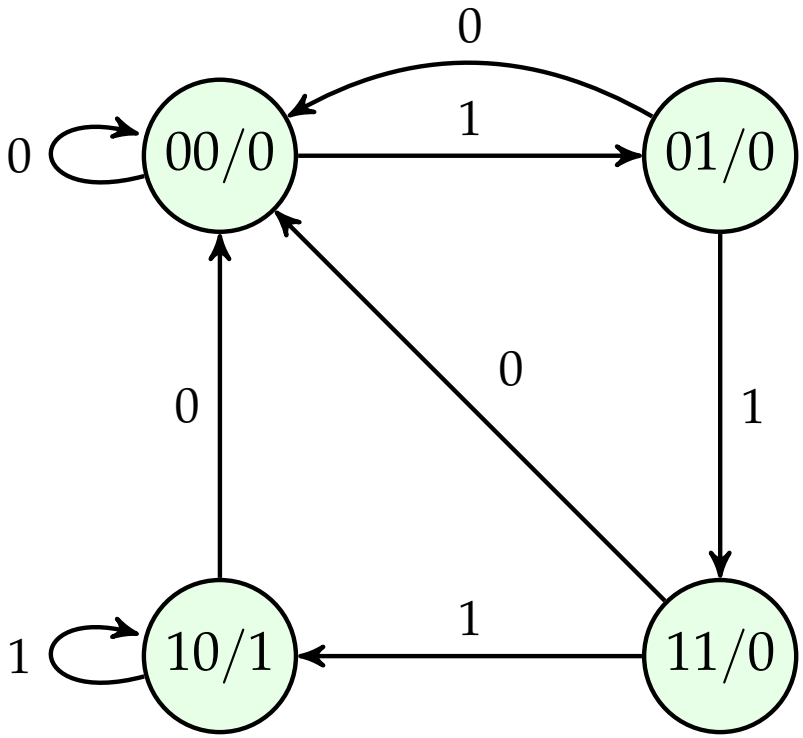
- In which situations you would prefer binary, gray, or one-hot encoding?

Example 1: Sequence Detector



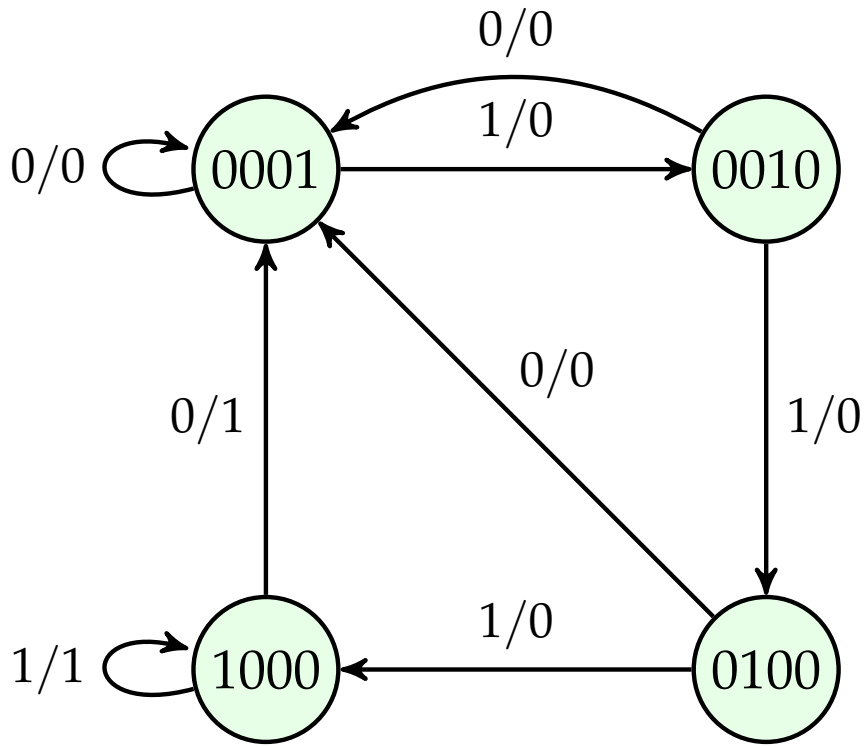
a) Mealy machine with Binary Assignment

Example 1: Sequence Detector



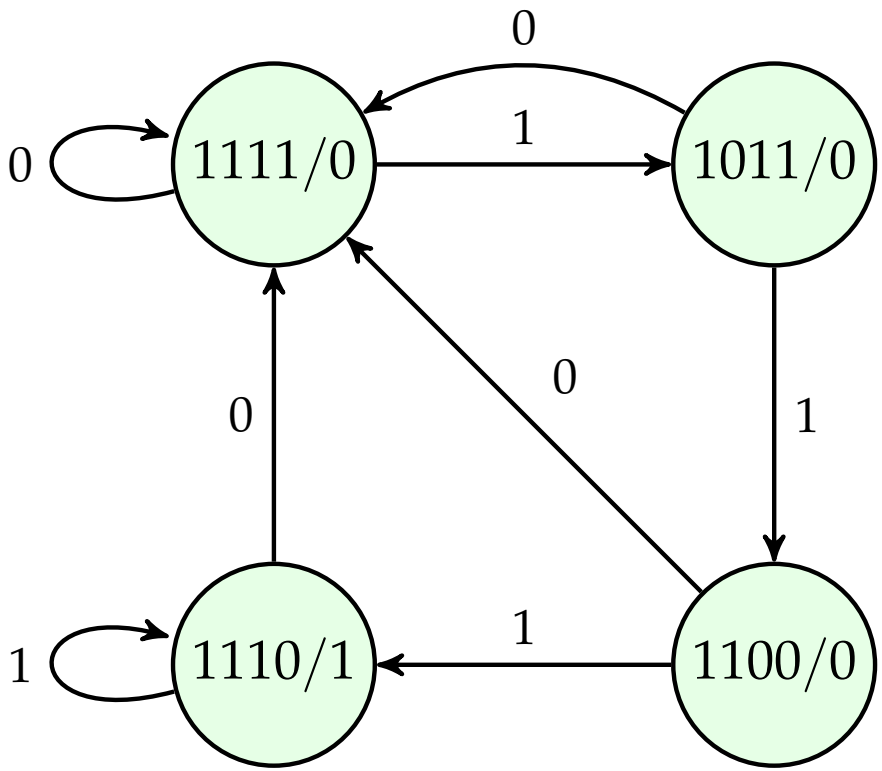
b) Moore machine with Gray Code Assignment

Example 1: Sequence Detector



c) Mealy machine with One-Hot Assignment

Example 1: Sequence Detector



d) Moore machine with Arbitrary Assignment

Step 4: Obtain binary-coded State Table

Present State		*Input*	*Next State*		*Output*
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

Step 5: Choose Flip-flops

Present State		*Input*	*Next State*		*Output*
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

- We need to store each state as a flip-flop such as *D*, J-K or *T* flip flops.
- We need to compute input logic for our flip-flops in order to properly change the state.
- The most common used flip-flop is of-course a D-flip-flop (Why?)

Excitation Table

- A excitation table defines the logical properties of the input of a flip-flop in order to make a desired change in the input.
- J-K Flip-flop

$Q(t)$	$Q(t+1)$	J	K
0	0	0	$\times$
0	1	1	$\times$
1	0	$\times$	1
1	1	$\times$	0

- T Flip-flop

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

- D Flip-flop

Step 6 and Step 7

- Using excitation table and state table, derive state table for corresponding flip-flop inputs
- Using K-maps simplify the flip-flop input equations and the output equations
- Draw the logic diagram.