EEECS1010 Logic Design
Lecture 5 Sequential Circuits

Jenny Yi-Chun Liu
jennyliu@gapp.nthu.edu.tw

Outline

- Digital systems and information
- Boolean algebra and logic gates
- Gate-level minimization
- Combinational logic
- Sequential circuits
- Registers and counters
- Memory

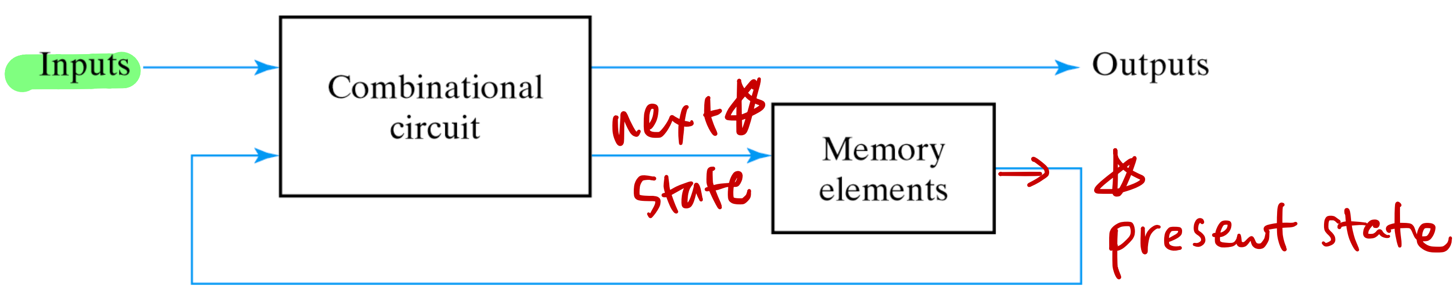
Overview

- Introduction
- Storage Element: latches
- Storage Element: flip-flops
- Analysis of clocked sequential circuits
- Design of clocked sequential circuits

Introduction

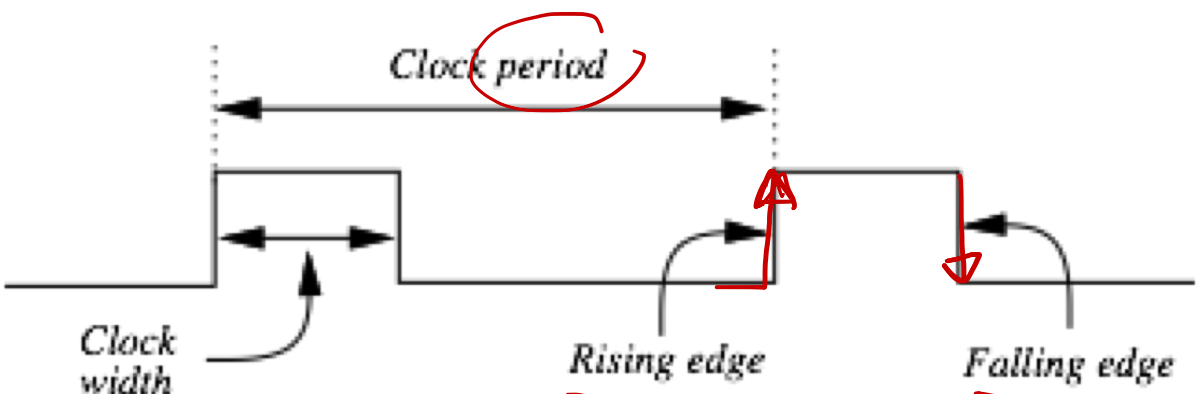
Introduction to Sequential Circuits

- A sequential circuit contains:
  - Combinational logic
  - Storage element (flip-flops, latches)
- Inputs and present state determine the outputs and next state.
  - Binary information stored in the memory elements defines the state of the sequential circuit.
- Block diagram of a sequential circuit:



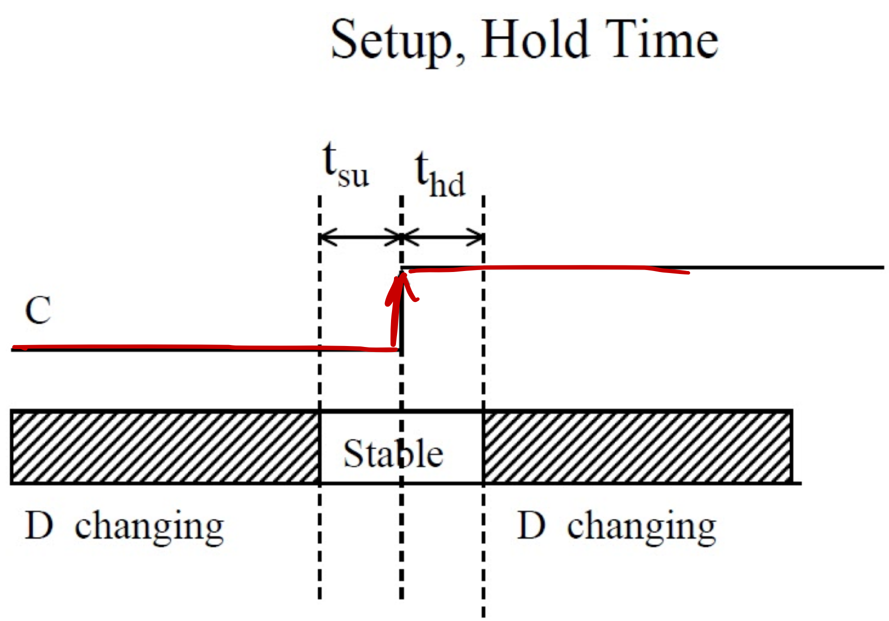
Synchronous vs. Asynchronous Sequential Circuits

- Timing of the respective signals differ.
- Synchronous sequential circuit: inputs and state are only defined at discrete time.
- Asynchronous sequential circuit: inputs and state can change at any time.
- Clock: a periodic train of clock pulses generated by timing device and distributed throughout the system.



Setup Time/Hold Time

- Setup time: Input must be maintained at a minimum amount of time prior to the clock edge.
- Hold time: Input must be maintained at a minimum amount of time after the clock edge.



Storage Elements: Latches

Latches

- Storage element: maintain a binary state indefinitely until directed by an input signal to switch state.

- Most basic storage element: lathes no clock
  - Latches are asynchronous sequential circuit. Its state changes whenever inputs change.
  - Latches have 2 states
  - Common latches: SR latch, D latch

SR Latch

- SR latch can be formed by two cross-coupled NORs or NANDs.
- Two inputs: set (S), reset (R)
- Two states: $Q = 1$ set state
  $Q = 0$ reset state
- Under normal conditions, both inputs (R, S) = 0 unless the state is to be changed.

inputs

Function table

S	R	Q^+ (next state)	
0	0	Q	no change
0	1	0	reset
1	0	1	set
1	1	X	undefined

S	R	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	X
1	1	1	X

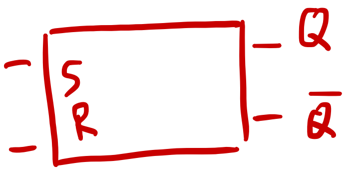
$Q^+ = Q$

reset

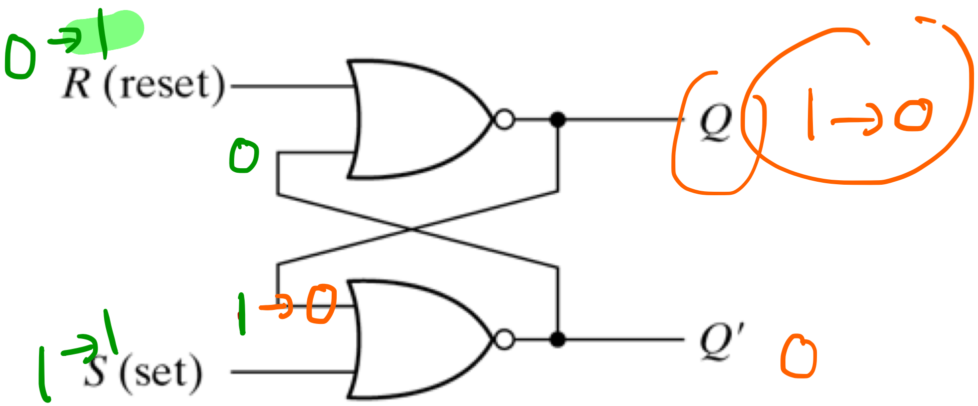
set

S \ RQ	00	01	11	10
0	0	1	0	0
1	1	1	X	X

$Q^+ = \bar{R}Q + S$



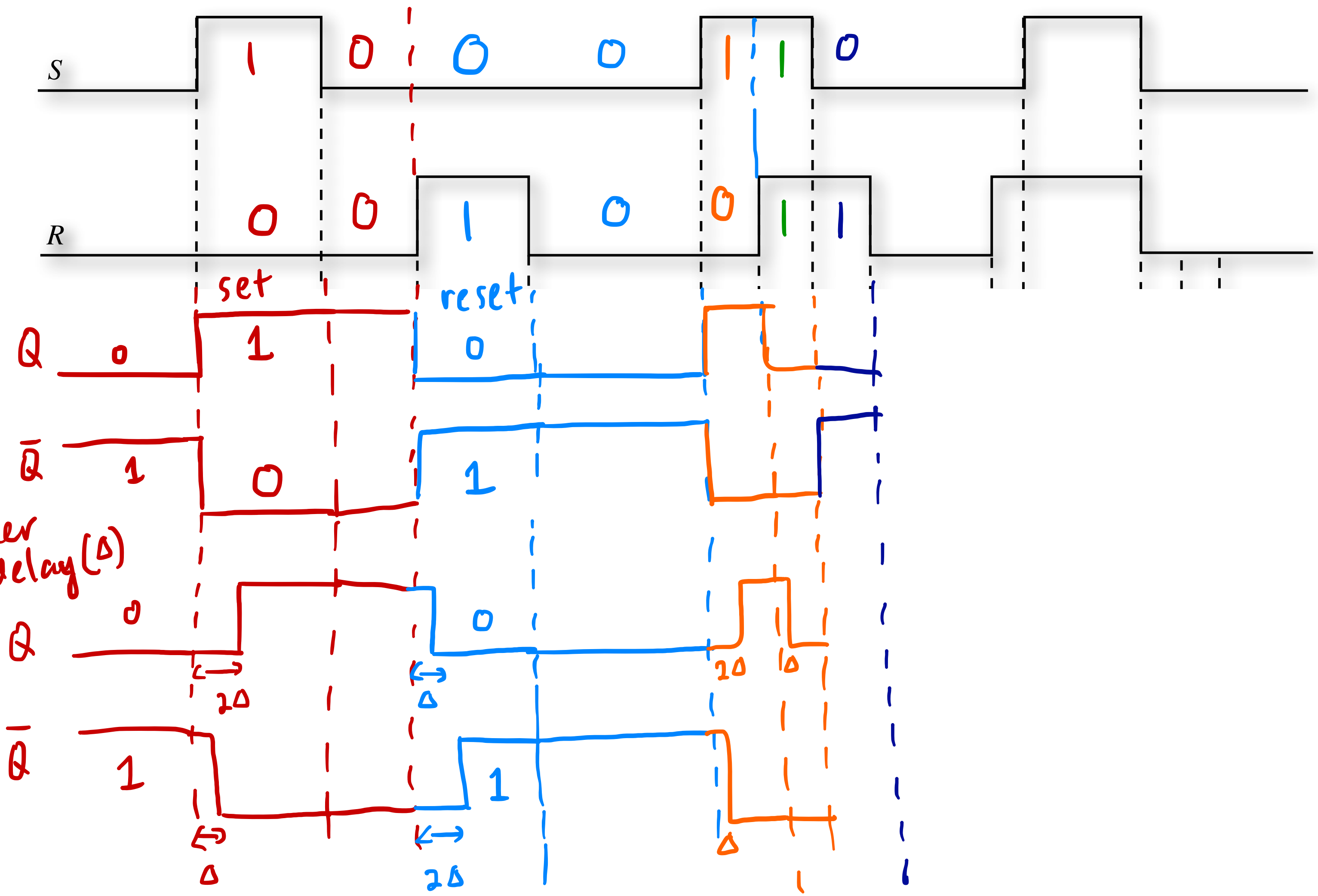
SR Latch with NORs



Assume initial $Q=0, \bar{Q}=1$

SR Latch with NORs Timing Diagram

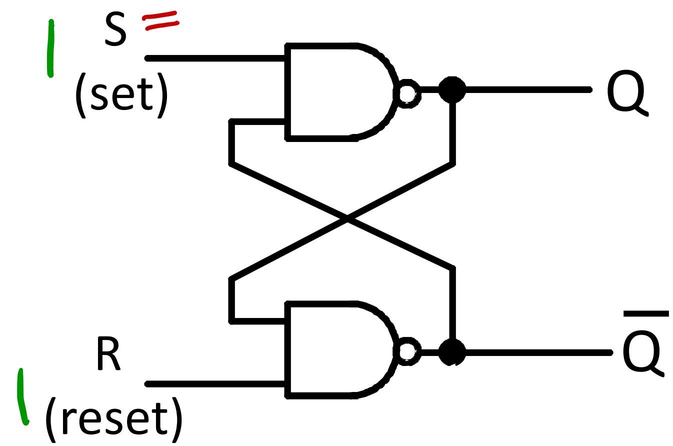
consider gate delay(Δ)



SR Latch with NANDs

Practice

- Under normal conditions, both inputs (R, S) = 1 unless the state is to be changed.

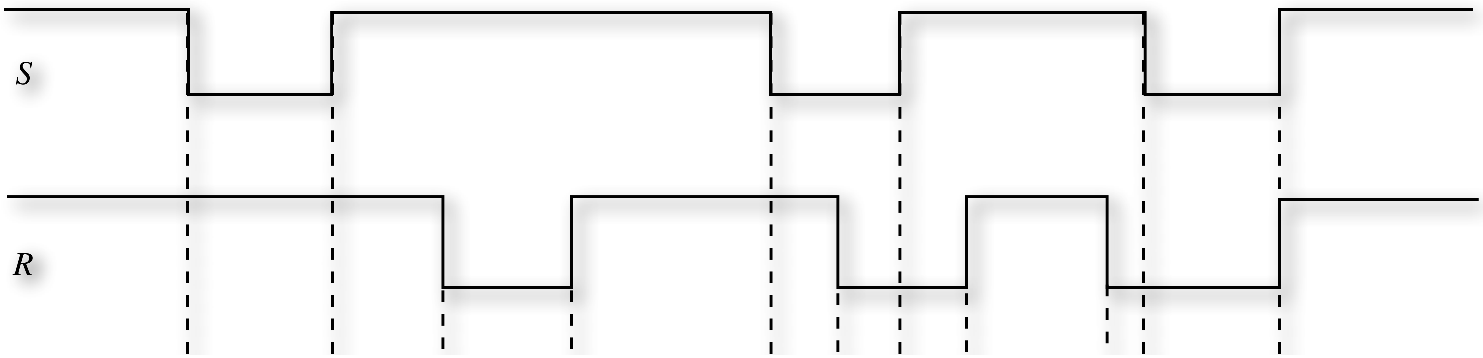


Function table

S	R	Q^+
0	0	
0	1	
1	0	
1	1	Q no change

Practice

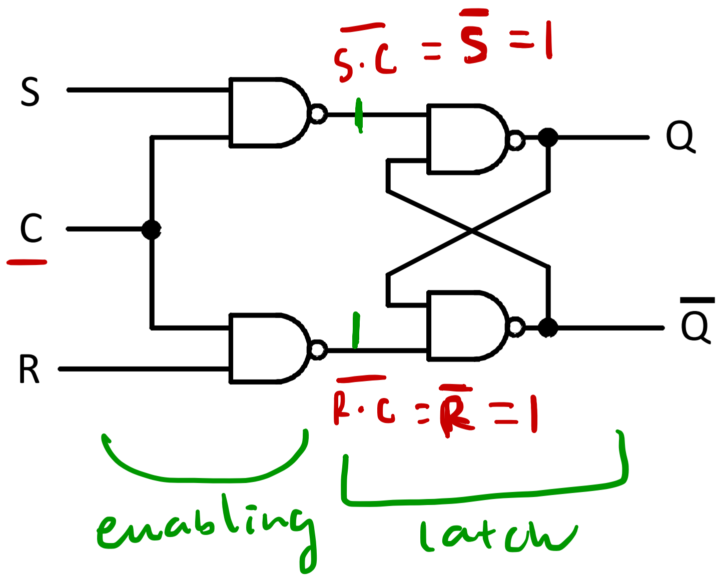
SR Latch with NANDs Timing Diagram



Clocked SR Latch (Gated SR Latch)

enabling input

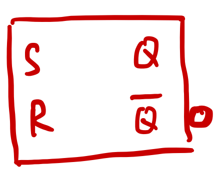
- Control input (C): determines when the state can be changed.



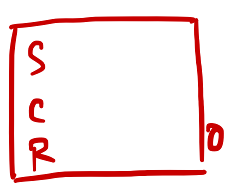
Function table

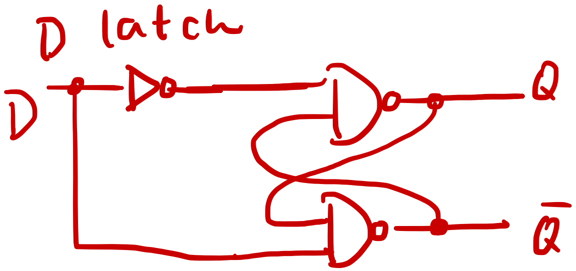
C	S	R	Q^+	
0	x	x	Q	no change/disabled
1	0	0	Q	no change
1	0	1	0	reset
1	1	0	1	set
1	1	1	x	undefined

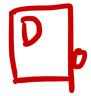
SR latch



clocked SR latch







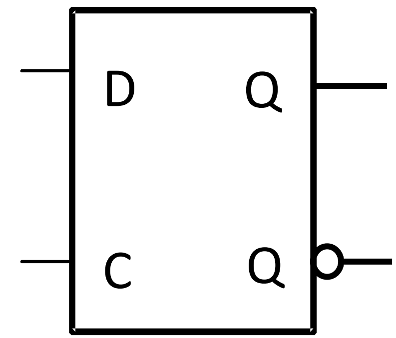
(clocked)

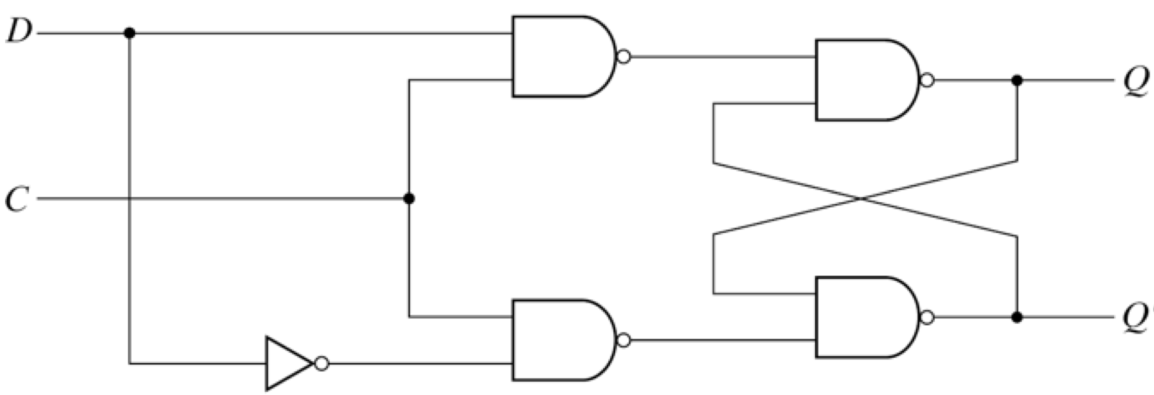
Gated D Latch

- Gated D latch has two inputs: D (data input), C (clock)
- No undefined state.

Function table

C	D	Q^+	
0	×	Q	no change / disabled
1	0	0	reset
1	1	1	set



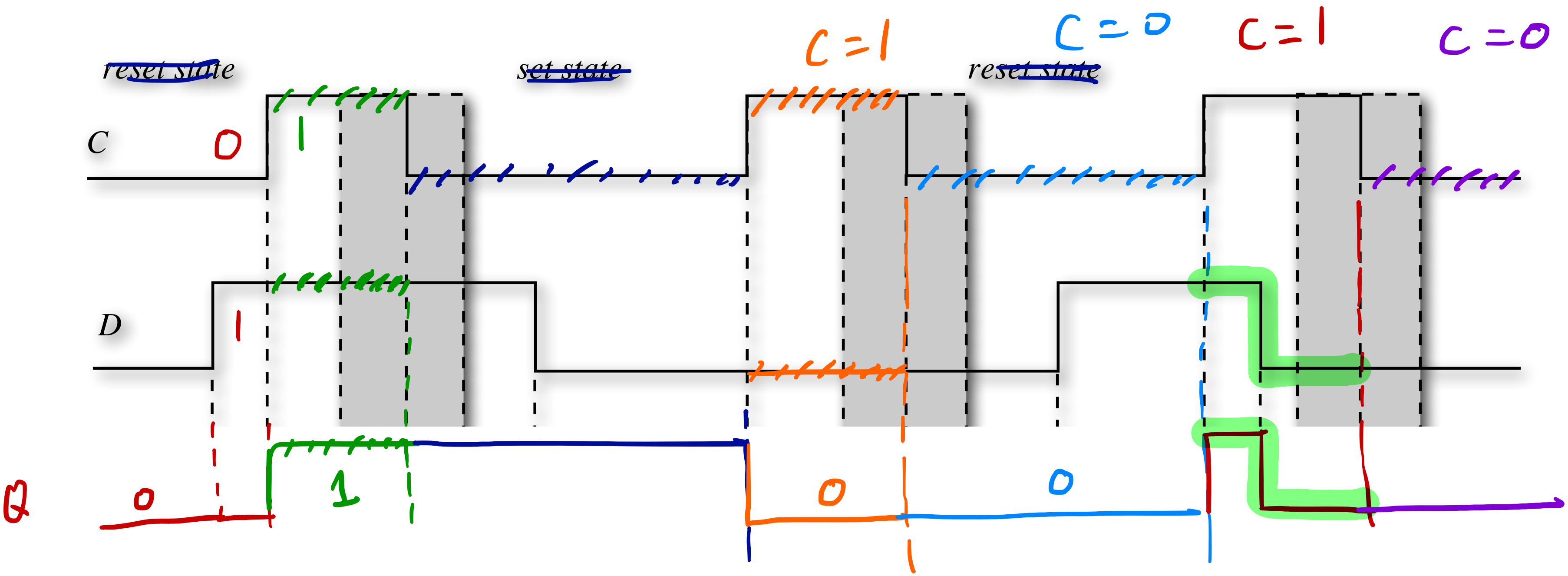


Assume
initial Q=0

clocked (positive level triggered)

c=1

D Latch Timing Diagram



C=1

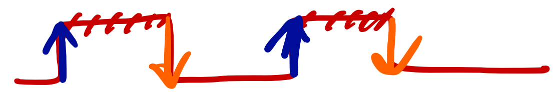
Storage Elements: Flip-Flops

Flip-Flops Outline

- Level-triggered vs. edge-triggered
- Edge-triggered flip-flop
- Standard symbols for storage elements
- Direct inputs

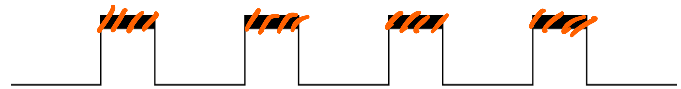
Trigger

- Trigger
  - The state of a latch or flip-flop is switched by a change of the control input.
- Level-triggered

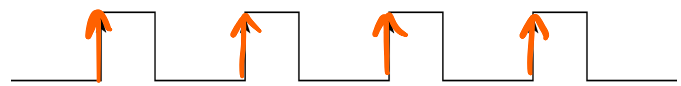


  - The state transition starts as soon as the clock is logic 1 (positive level-sensitive) or logic 0 (negative level-sensitive) level.
- Edge-triggered
  - The state transition starts only at positive (positive edge-triggered) or negative edge (negative edge-triggered) of the clock signal.

Level-Triggered vs. Edge-Triggered



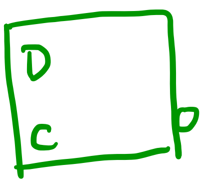
(a) Response to positive level

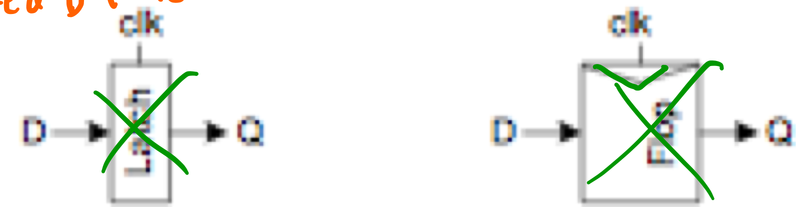


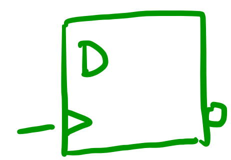
(b) Positive-edge response

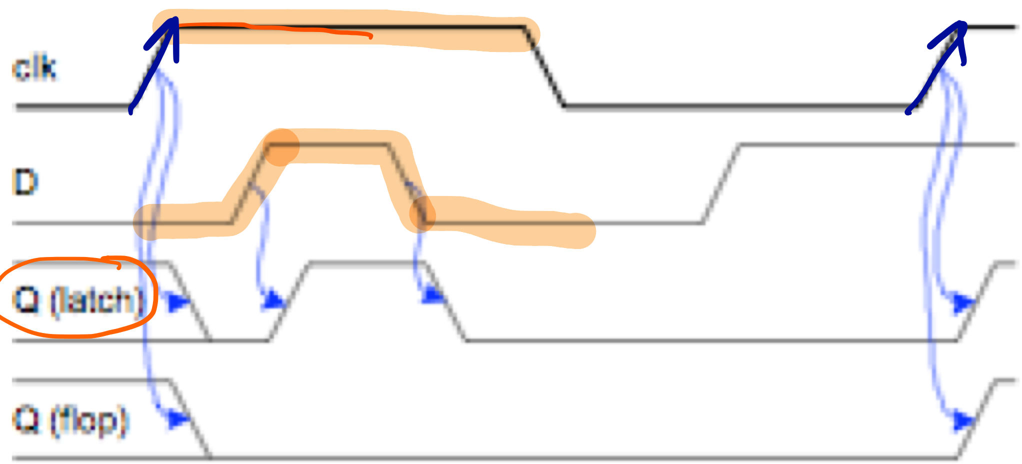
Edge-triggered

clocked D latch



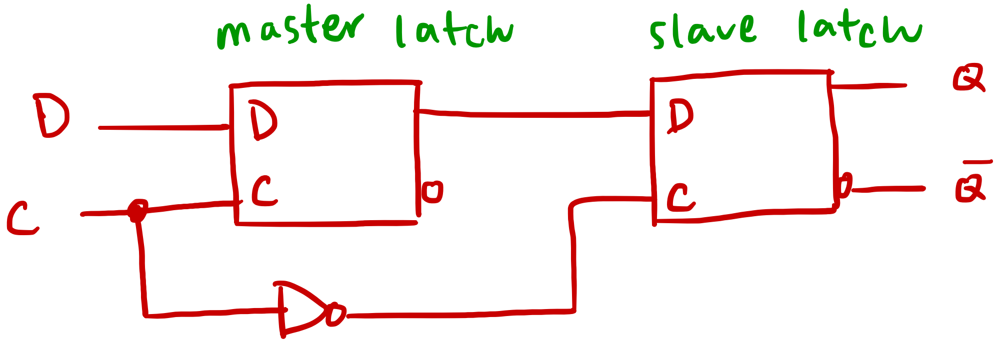


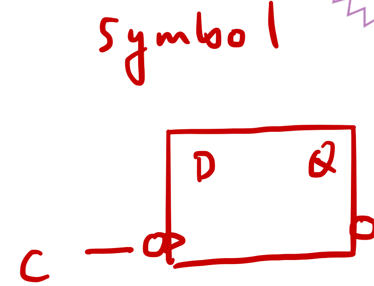




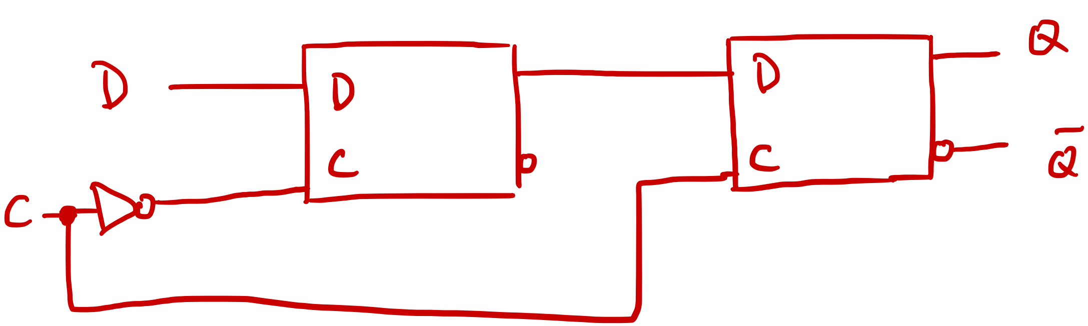
positive level sensitive

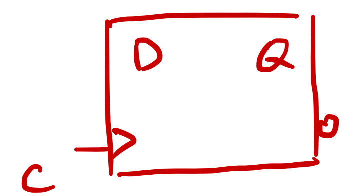
Edge-Triggered D Flip-Flop





negative edge triggered DFF





positive edge triggered DFF

Edge-Triggered DFF Timing

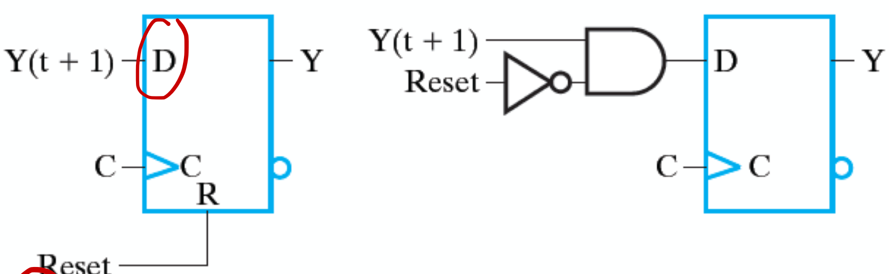
- A master-slave D flip-flop is formed by two separate latches
  - A master D latch (negative level sensitive)
  - A slave D latch (positive level sensitive)

Direct Inputs

- At power up or at reset, all or part of a sequential circuit usually is initialized to a known state before it begins operation.
- The direct input asynchronously sets the flip-flop.
  - Preset/set: set to 1
  - Reset/clear: set to 0
- The direct input can also be controlled by the clock, called synchronous direct input.

Function table (asynchronous reset)

R (Reset)	C	Y(t+1)	Y
1	X	X	0
0	↑	0	0
0	↑	1	1

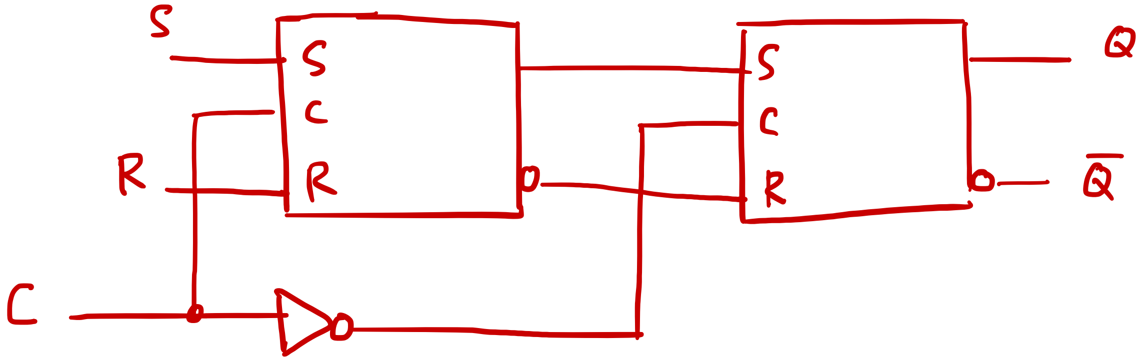


(a) Asynchronous Reset (b) Synchronous Reset

Other Flip-Flop Types

- Four major common FFs
  - SR (set-reset) 不考
  - D (data)
  - JK
  - T (toggle) 不考
- Basic descriptors for understanding and using different flip-flop types
  - Characteristic tables
  - Characteristic equations
  - Excitation tables

SR Flip-Flop



JK Flip-Flop

Characteristic table
(Function table)

Similar to SRFF
(J: set
K: reset)

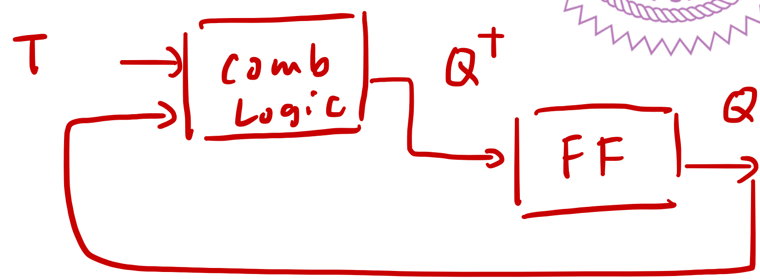
J	K	Q^+	
0	0	Q	no change
0	1	0	reset
1	0	1	set
1	1	$\bar{Q}$	complement

$Q^+ = J\bar{Q} + \bar{K}Q$ characteristic equation

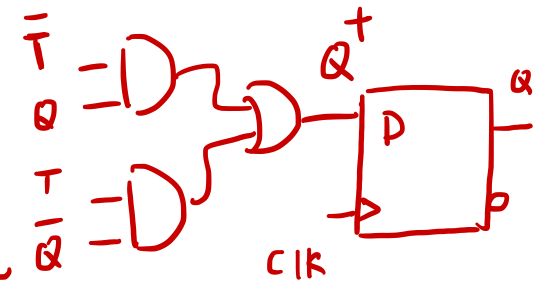
(Practice)

clock: clk

T Flip-Flop



- TFF behavior
  - Has a single input T
    - For T = 0, no change to state.
    - For T = 1, changes to opposite state.
- Same as a J-K flip-flop with J = K = T.



Characteristic table

T	Q^+
0	Q
1	$\bar{Q}$

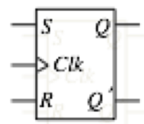
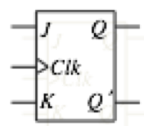
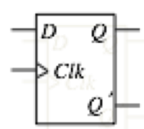
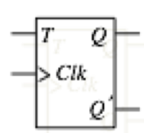
characteristic equation

$$Q^+ = \bar{T}Q + T\bar{Q}$$

Summary of Flip-Flops

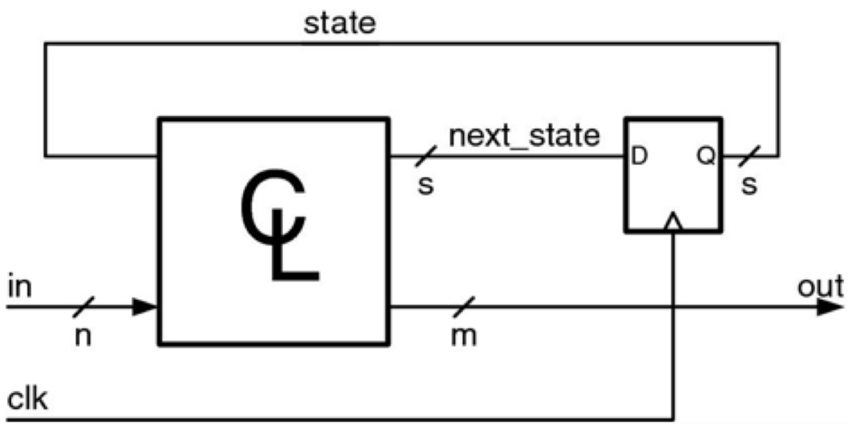
- Characteristic table: define next state in terms of inputs and current states.
- Characteristic equation: define next state as a Boolean function in terms of inputs and current state.
- Excitation table: define input variables as a function of current state and next state.

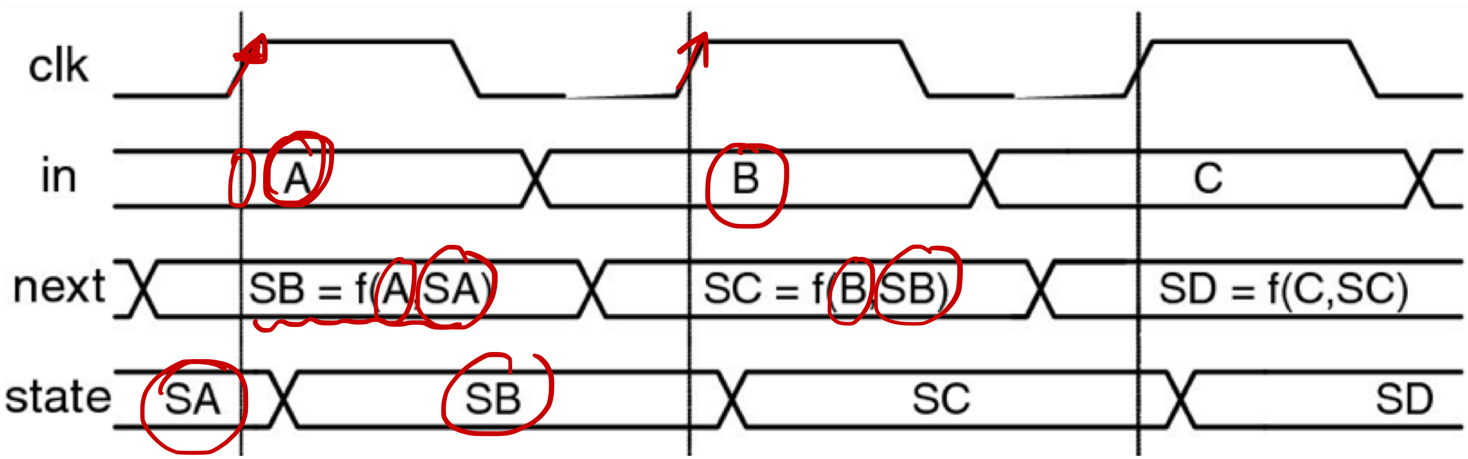
	S	R
reset	0	1
no change	0	0

Flip-flop name	Flip-flop symbol	Characteristic table	Characteristic equation	Excitation table
SR		S R | Q (next) 0 0 | Q 0 1 | 0 1 0 | 1 1 1 | NA	$Q(next) = S + R'Q$ $SR = 0$	Q Q(next) | S R 0 0 | 0 X 0 1 | 1 0 1 0 | 0 1 1 1 | X 0
JK		J K | Q (next) 0 0 | Q 0 1 | 0 1 0 | 1 1 1 | Q'	$Q(next) = JQ' + K'Q$	Q Q(next) | J K 0 0 | 0 X 0 1 | 1 X 1 0 | X 1 1 1 | X 0
✓ D		D | Q (next) 0 | 0 1 | 1	$Q(next) = D$	Q Q(next) | D 0 0 | 0 0 1 | 1 1 0 | 0 1 1 | 1
T		T | Q (next) 0 | Q 1 | Q'	$Q(next) = TQ' + T'Q$	Q Q(next) | T 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0

Analysis and Design of Sequential Circuits

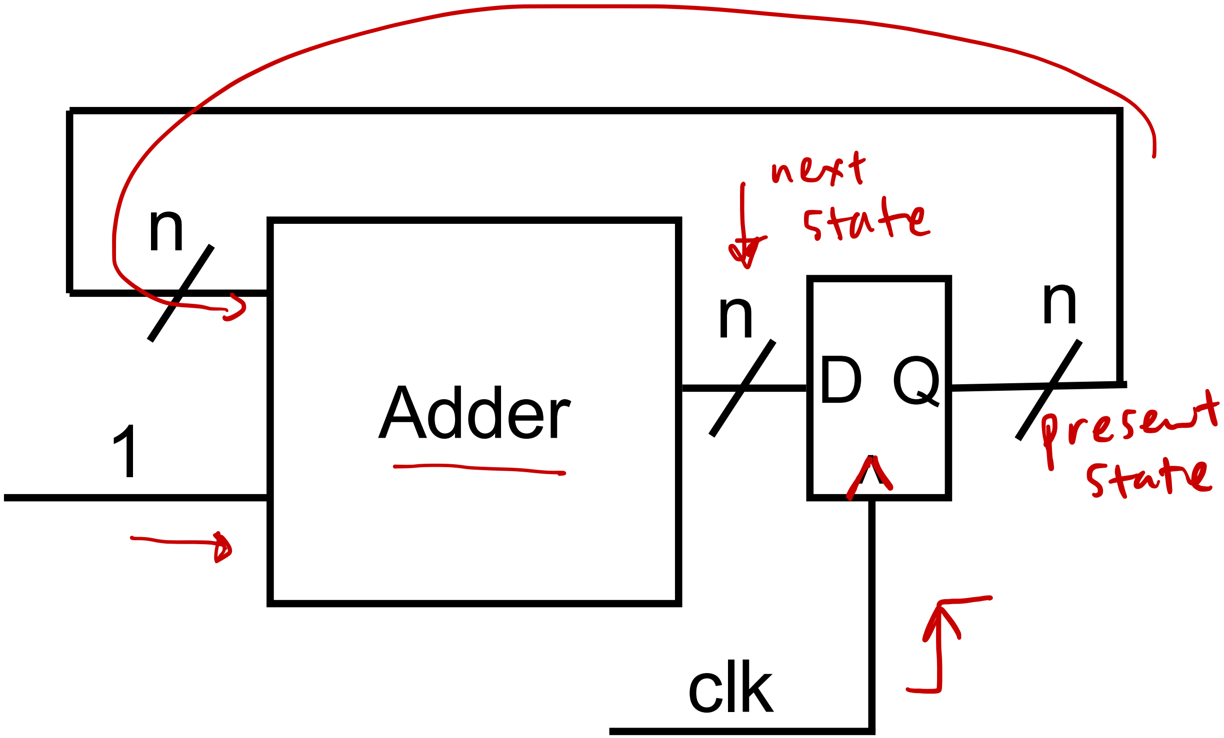
Synchronous Sequential Circuits





Example: Incrementer

- Increment a count on every clock tick.



Sequential Circuit Analysis

- Obtain a table or diagram for the sequence of inputs, outputs, and internal states
- General model
  - Current State at time (t) is stored in an array of flip-flops
  - Next State at time (t+1) is a Boolean function of State and inputs
  - Outputs at time (t) are a Boolean function of state (t) and (sometimes) inputs (t)
- Analysis procedure
  - Derive excitation (input) equation
  - Derive next-state and output equations
  - Generate next-state and output tables
  - Generate state diagram
  - Develop timing diagram
  - Simulate logic circuit

↔ truth table

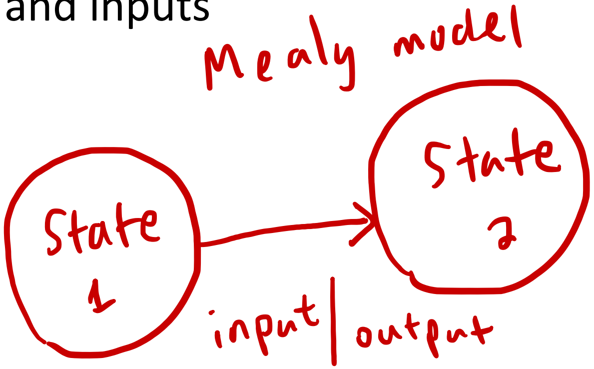
State Table Characteristics

- State table – a multiple variable table with the sections:
  - inputs { – Present State
  - – Input
  - outputs { – Next-state
  - – Output
- From the viewpoint of a truth table:
  - The inputs are Input, Present State
  - The outputs are Output, Next State

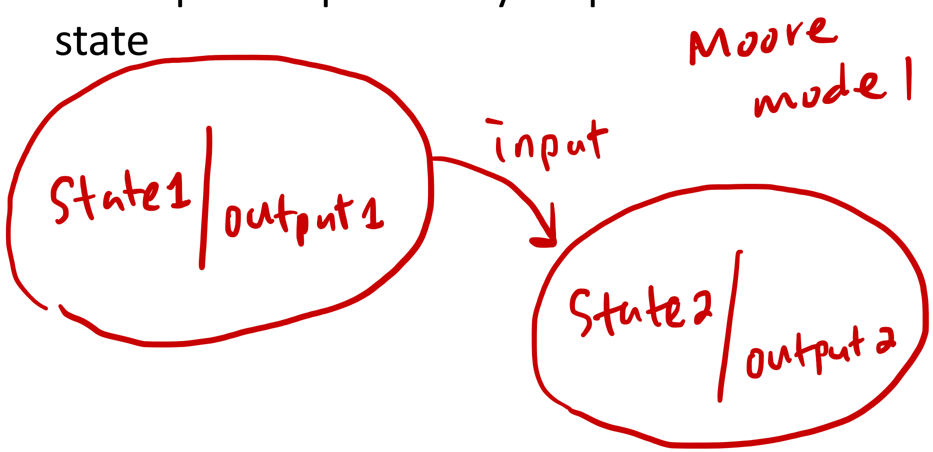
State Diagram

- The sequential circuit can be represented in graphical form as a state diagram with the following components:
  - Circle: with the name of the state in it.
  - Directed line: from the present state to the next state.

1. Outputs depend on present state and inputs



2. Outputs depend only on present state

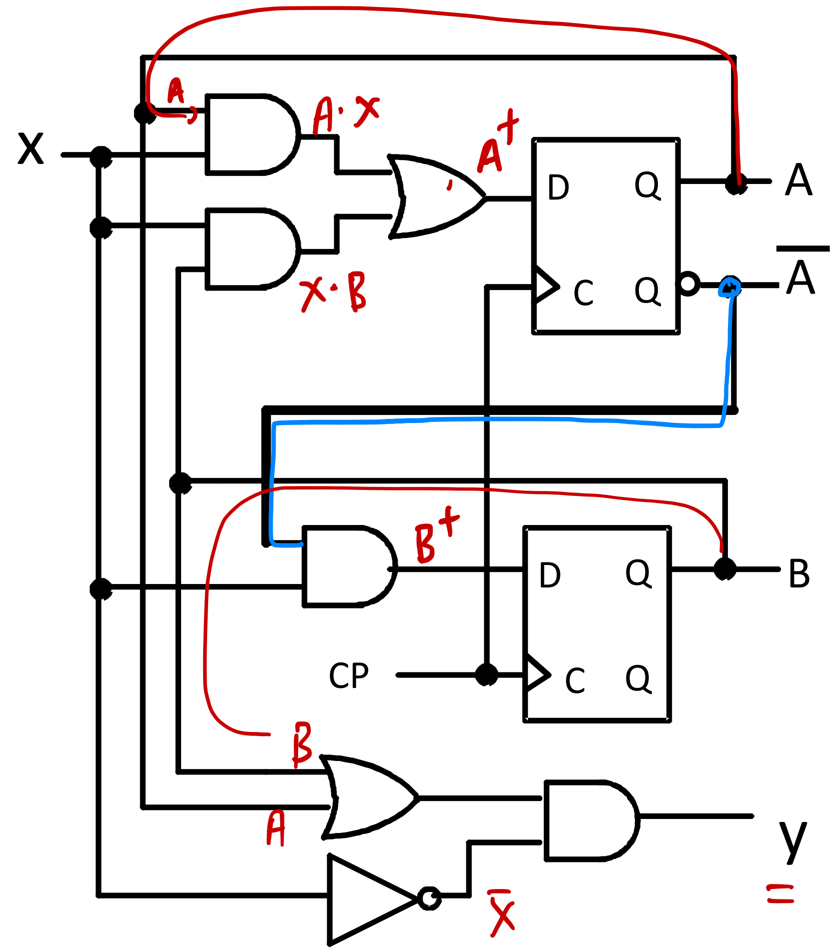


Sequential Circuit Example I (1/2)

- Input: x(t)
- Output: y(t)
- States: A(t), B(t)
- First, find the inputs of the FFs

$A^+(t) = A(t+1) = A(t)\cdot X(t) + B(t)\cdot X(t)$

$B^+(t) = B(t+1) = \bar{A}(t)\cdot X(t)$



- Output equation:

$y(t) = [A(t) + B(t)]\cdot \bar{x}(t)$

- Next state equation:

$A(t+1) = A(t)\cdot X(t) + B(t)\cdot X(t)$

$B(t+1) = \bar{A}(t)\cdot X(t)$

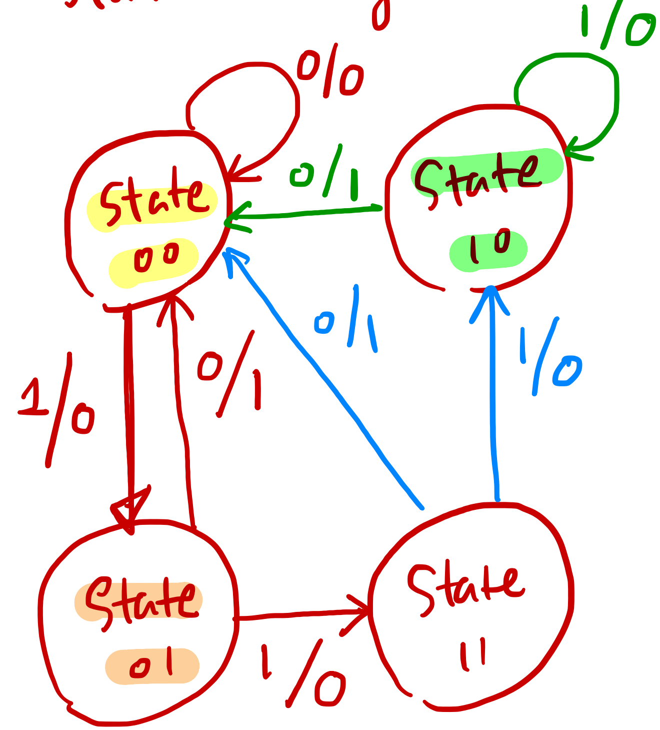
• State table

Sequential Circuit Example I (2/2)

inputs

Present State		Input	Next State		Output
A(t)	B(t)	x(t)	A(t+1)	B(t+1)	y(t)
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 diagram



• $y(t) = B\bar{x} + A\bar{x}$

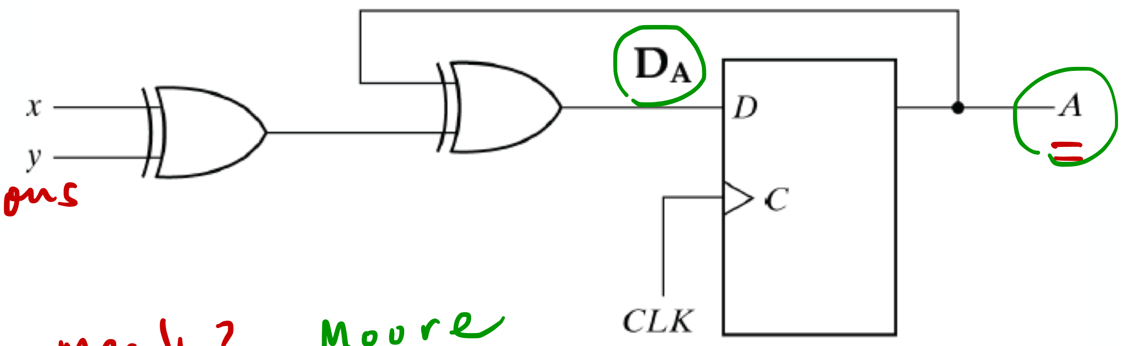
A \ Bx	00	01	11	10
0	0	0	0	1
1	1	0	0	1

Mealy

Sequential Circuit Example II

output : A

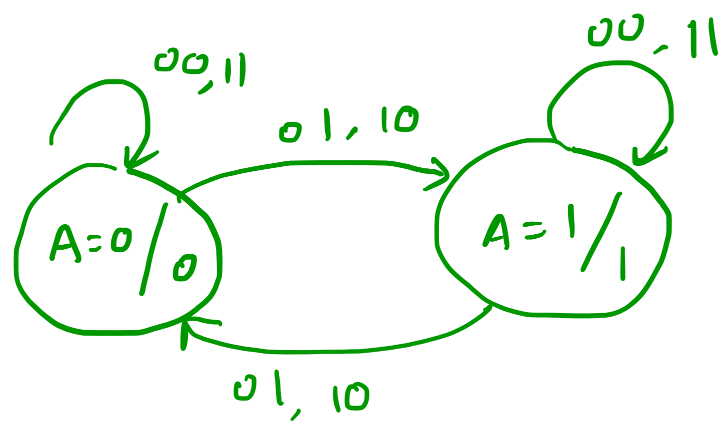
input : x, y.



- Output, next-state equations
- State table
- State diagram. Moore or Mealy? Moore

$$D_A(t) = A(t+1) = x(t) \oplus y(t) \oplus A(t)$$

x	y	A	D_A
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



State diagram.

Finite State Machine (FSM)

- A synchronous sequential circuit can be modeled by a finite state machine (FSM).
- Two models for FSM:
  - Moore: outputs are function of only the present state
  - Mealy: outputs are function of both the present state and inputs

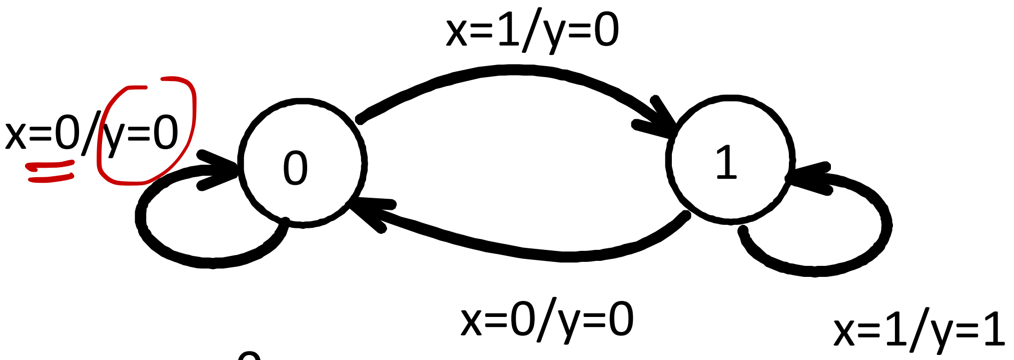
Moore and Mealy Example Diagrams

Practice

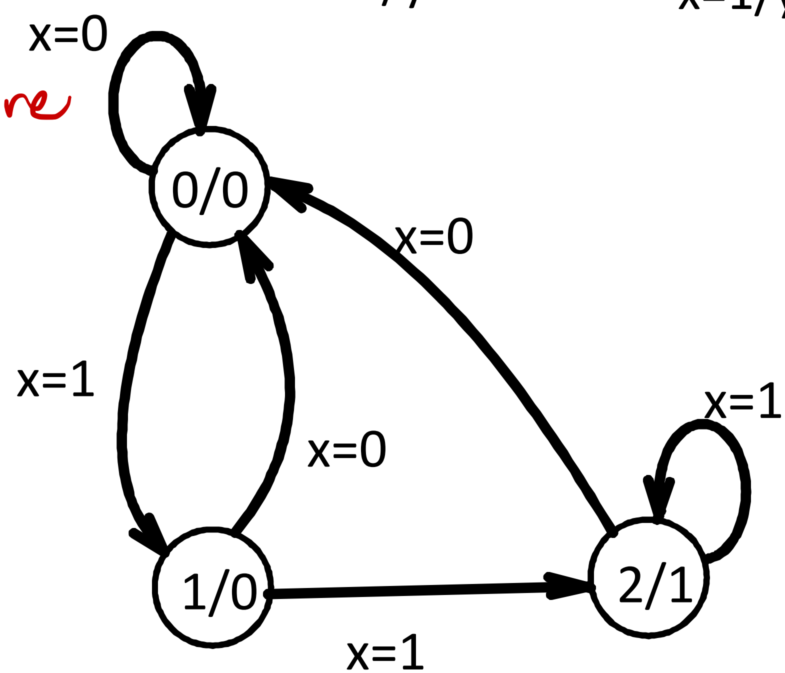
logic diagram

State table

Mealy

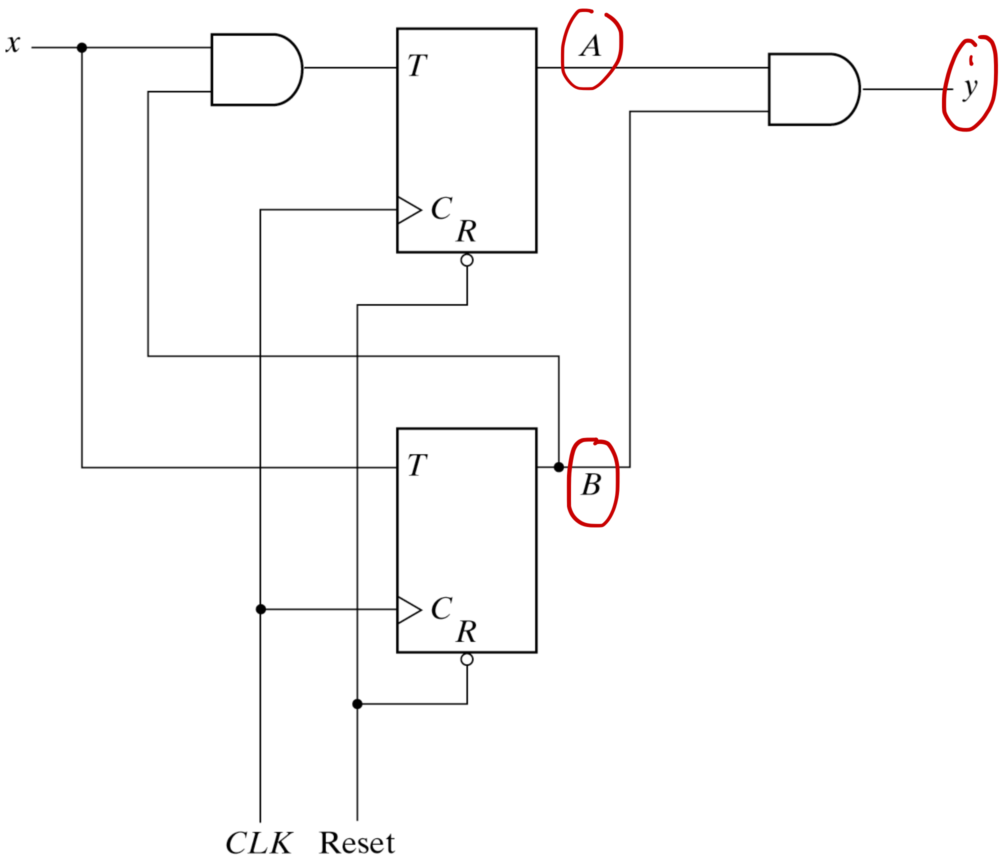


Moore

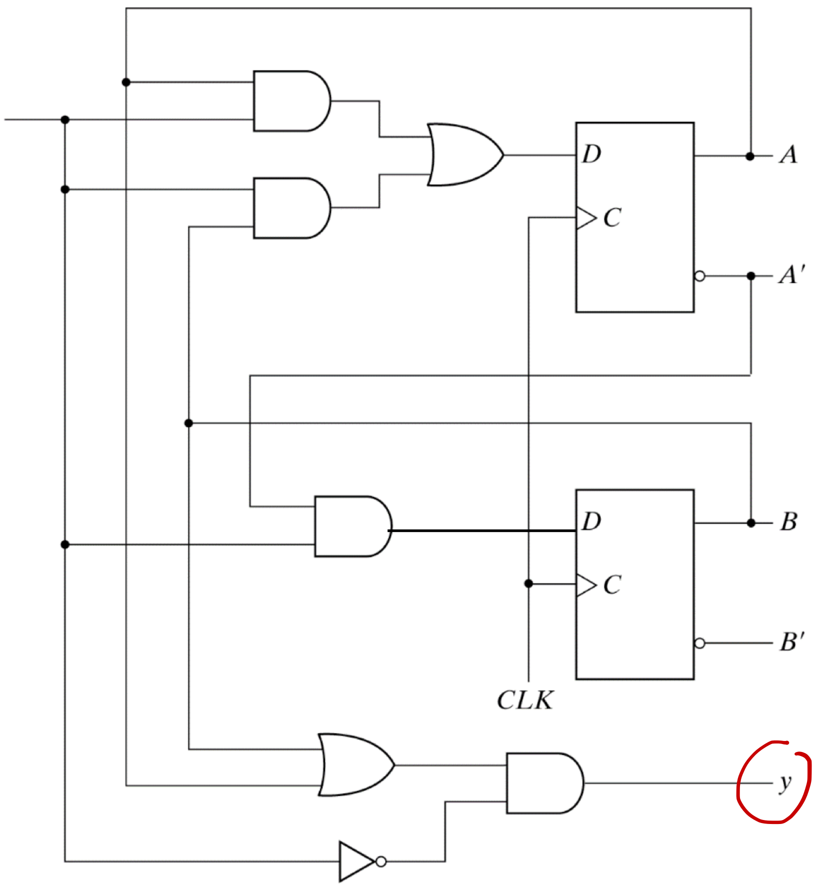


Moore and Mealy Examples

Moore



Mealy



Moore and Mealy Example Tables

Moore

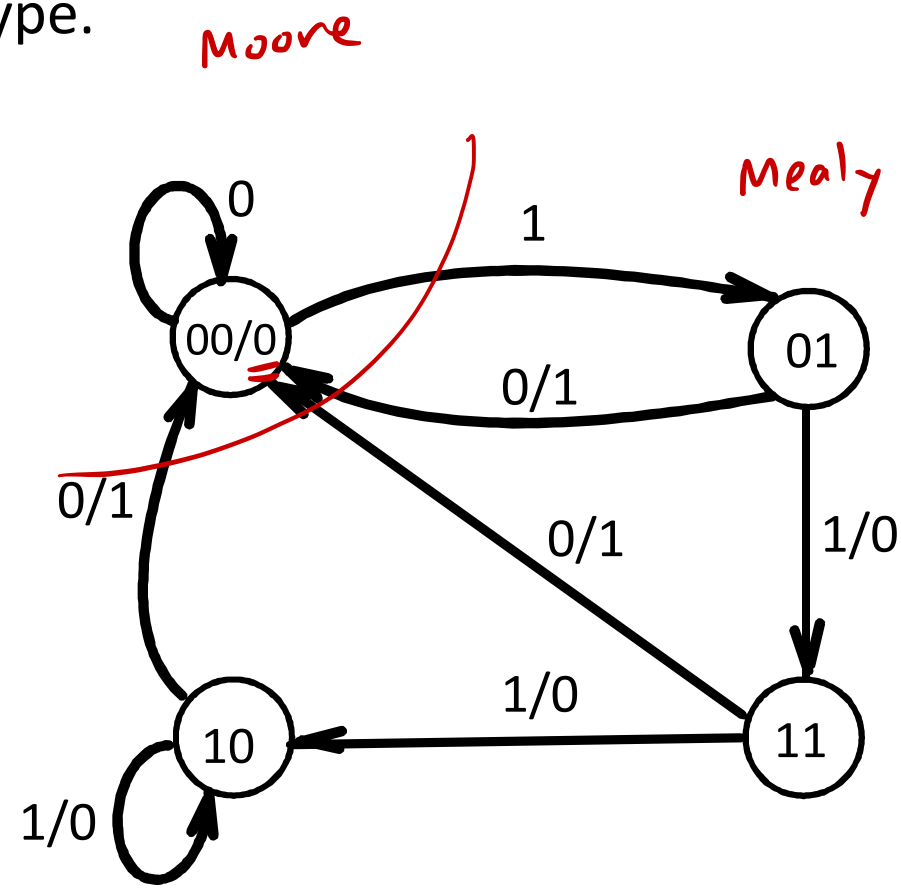
Present State	Next State x=0	Next State x=1	Output
0	0	1	0
1	0	2	0
2	0	2	1

Mealy

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

Mixed Moore and Mealy Outputs

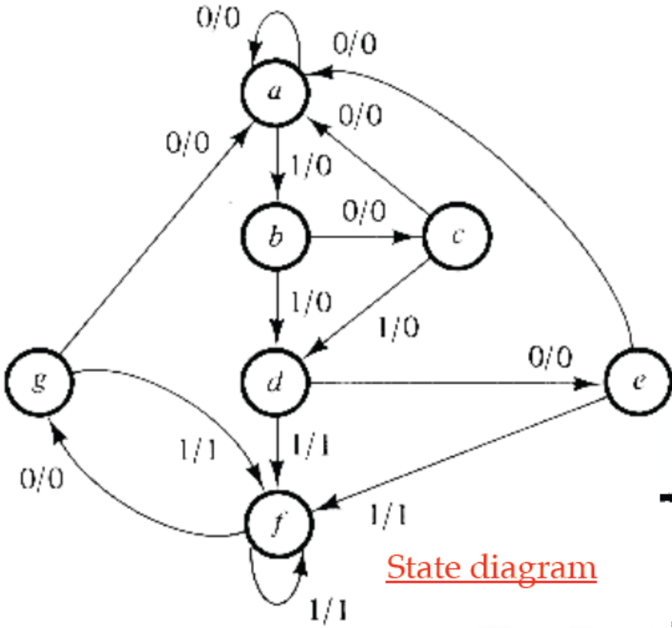
- In real designs, some outputs may be Moore type and other outputs may be Mealy type.



State Minimization

- State reduction
  - Reductions on the number of flip-flops (states) and the number of gates.
  - For an FSM with m states, we need $\lceil \log_2 m \rceil$ FFs.
- Steps
  - Find rows in the state table that have identical next state and output entries. They are equivalent state. One of them can be removed.
  - Update the state table reflecting the change. Continue until there is no equivalent state.

State Minimization Example (1/2)



State diagram

1. e, g : equivalent state

2. d, f :

State table

Present State	Next State $x = 0$	Next State $x = 1$	Output $x = 0$	Output $x = 1$
a	*a*	*b*	0	0
b	*c*	*d*	0	0
c	*a*	*d*	0	0
d	*e*	~~*f*~~ d	0	1
e	*a*	~~*f*~~ d	0	1
~~*f*~~	~~*g*~~ e	~~*f*~~	~~0~~	~~1~~
~~*g*~~	~~*a*~~	~~*f*~~	~~0~~	~~1~~

State Minimization Example (2/2)

- Practice

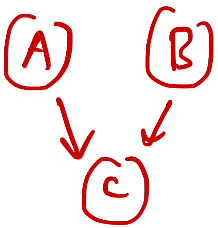
  Draw the updated state diagram.

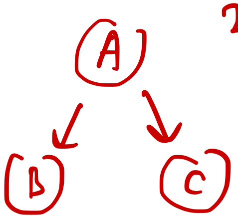
State Assignment

- Each of the *m* states must be assigned a unique binary code.
- Minimum number of bits required is *n* such that $\lceil \log_2 m \rceil$
- There are useful state assignments that use more than the minimum number of bits.
- Different state assignments result in different circuits for the intended FSM.
- There is no easy state-assignment procedure that guarantees a minimal-cost or minimum-delay combinational circuits.
  - Exploration of all possibilities are impossible
    - Minimum-bit change
    - Prioritized adjacency
    - One-hot encoding

5

000 ✓
001 ✓
010 ✓
011 ✓
100 ✓





1) Gray code
2) Counting order (000, 001, 010, ...)
3) One hot

State Assignment Example (1/2)

- Counting Order Assignment: A = 0 0, B = 0 1, C = 1 0, D = 1 1
- The resulting coded state table:

Gray code

Present State	Next State x = 0	Next State x = 1	Output x = 0	Output x = 1
0 0	0 0	0 1	0	0
0 1	0 0	~~1 0~~ 11	0	0
~~1 0~~ 11	~~1 1~~ 10	~~1 0~~ 11	0	0
~~1 1~~ 10	0 0	0 1	0	1

- Gray Code Assignment: A = 0 0, B = 0 1, C = 1 1, D = 1 0

State Assignment Example (2/2)

- One-hot assignment: for m states, use m bits to form the codes that contain only one "1" in each code.

Present State	Next State x = 0	Next State x = 1	Output x = 0	Output x = 1
0 0 → 0001	0 0 → 0001	0 1 → 0010	0	0
0 1 → 0010	0 0 → 0001	1 0 → 0100	0	0
1 0 → 0100	1 1 → 1000	1 0 → 0100	0	0
1 1 → 1000	0 0 → 0001	0 1 → 0010	0	1

Choice of Memory Elements

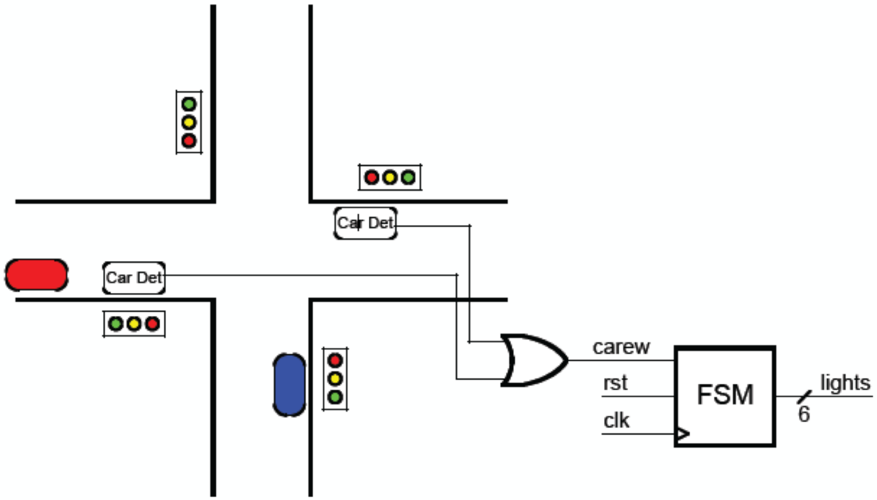
- Given the state table, we need to find the FF input conditions that cause the required transition.
  - Excitation table can be used.
  - SRFFs are used when different signals set/reset FFs.
  - DFFs are good for applications requiring data transfer.
  - TFFs are good for applications involving complementation.
  - Many digital circuits are constructed entirely with JKFFs because of their versatility.

Design Procedure of Clocked Sequential Circuits

1. Specifications - describe the design.
2. Formulation - obtain a state diagram and state table.
3. State minimization.
4. State assignment.
5. Input and output equations derivation.
6. Choose memory elements.
7. Map the circuit to the memory and gates (logic diagram).
8. Simulation
9. Verification

Design Example: Traffic Light Counter (1/7)

- Spec
  - Reset to green in north south direction.
  - If light is green or yellow in one direction, it must be red in the other side.
  - A light must be yellow between changing from green to red.
  - If there is a car waiting at east west (carew=1), make the light green in east west and return to green in north south.

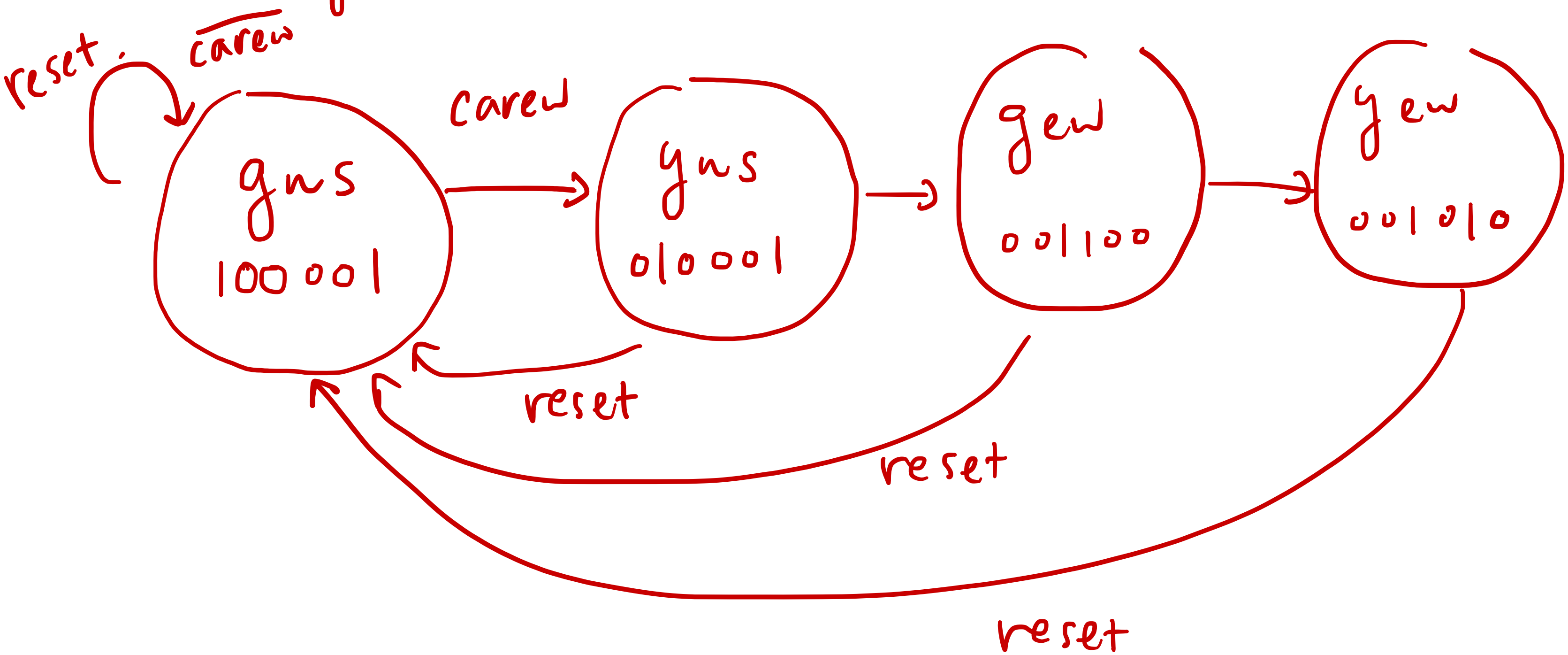


Traffic Light Counter FSM (2/7)

- Four states:
  - gns: green north south (red east west)
  - yns: yellow north south (red east west)
  - gew: green east west (red north south)
  - yew: yellow east west (red north south)
- Input
  - Carew: indicate if there is a car waiting at east west
  - Reset: return to initial state (need not go through yellow)
- Output
  - 100 001: NS green, EW red; 001 010 NS red, EW yellow

Traffic Light Counter FSM (3/7)

- State diagram.



Traffic Light Counter FSM (4/7)

- State assignment

5

Binary Code

State	Encoding
GNS	00
YNS	01
GEW	10
YEW	11

Gray Code

State	Encoding
GNS	00
YNS	01
GEW	11
YEW	10

State Assignment with Gray Code

current state		carew	next state		next state		output					
$s_1(t)$	$s_0(t)$	c	$s_1(t+1)$	$s_0(t+1)$	$ns_1(t)$	$ns_0(t)$	lgns	lyns	lrns	lgew	lyew	lrew
0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	1	0	1	0	1	1	0	0	0	0	1
0	1	0	1	1	1	1	0	1	0	0	0	1
0	1	1	1	1	1	1	0	1	0	0	0	1
1	1	0	1	0	1	0	0	0	1	1	0	0
1	1	1	1	0	1	0	0	0	1	1	0	0
1	0	0	0	0	0	0	0	0	1	0	1	0
1	0	1	0	0	0	0	0	0	1	0	1	0

Traffic Light Counter FSM (5/7)

- Next state equations (use DFFs)

current state		carew	next state		excitation		output					
$s_1(t)$	$s_0(t)$	c	$s_1(t+1)$	$s_0(t+1)$	$ns_1(t)$	$ns_0(t)$	lgns	lyns	lrns	lgew	lyew	lrew
0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	1	0	1	0	1	1	0	0	0	0	1
0	1	0	1	1	1	1	0	1	0	0	0	1
0	1	1	1	1	1	1	0	1	0	0	0	1
1	1	0	1	0	1	0	0	0	1	1	0	0
1	1	1	1	0	1	0	0	0	1	1	0	0
1	0	0	0	0	0	0	0	0	1	0	1	0
1	0	1	0	0	0	0	0	0	1	0	1	0

Traffic Light Counter FSM (6/7)

$s_1 \backslash s_0c$	00	01	11	10
0	0	0	1	1
1	0	0	1	1

$s_1(t+1) = s_1^+(t) = s_0$

$s_1 \backslash s_0c$	00	01	11	10
0	0	1	1	1
1	0	0	0	0

$s_0^+(t) = \bar{s}_1 c + \bar{s}_1 s_0$

current state		carew	next state		excitation		output					
$s_1(t)$	$s_0(t)$	c	$s_1(t+1)$	$s_0(t+1)$	$ns_1(t)$	$ns_0(t)$	lgns	lyns	lrns	lgew	lyew	lrew
0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	1	0	1	0	1	1	0	0	0	0	1
0	1	0	1	1	1	1	0	1	0	0	0	1
0	1	1	1	1	1	1	0	1	0	0	0	1
1	1	0	1	0	1	0	0	0	1	1	0	0
1	1	1	1	0	1	0	0	0	1	1	0	0
1	0	0	0	0	0	0	0	0	1	0	1	0
1	0	1	0	0	0	0	0	0	1	0	1	0

$c \backslash s_1s_0$	00	01	11	10
0	1			
1	1			

$\text{lgns} = s_1' s_0'$

$c \backslash s_1s_0$	00	01	11	10
0		1		
1		1		

$\text{lyns} = s_1' s_0$

$c \backslash s_1s_0$	00	01	11	10
0			1	1
1			1	1

$\text{lrns} = s_1$

$c \backslash s_1s_0$	00	01	11	10
0			1	
1			1	

$\text{lgew} = s_1 s_0$

$c \backslash s_1s_0$	00	01	11	10
0				1
1				1

$\text{lyew} = s_1 s_0'$

$c \backslash s_1s_0$	00	01	11	10
0	1	1		
1	1	1		

$\text{lrew} = s_1'$

Traffic Light Counter FSM (7/7)

Excitation equation (Input equation)

$$\mathrm{ns}_1 = s_0$$

$$\mathrm{ns}_0 = cs_1' + s_1' s_0 = (c + s_0) s_1'$$

Output equation

$$\mathrm{lgns} = s_1' s_0'$$

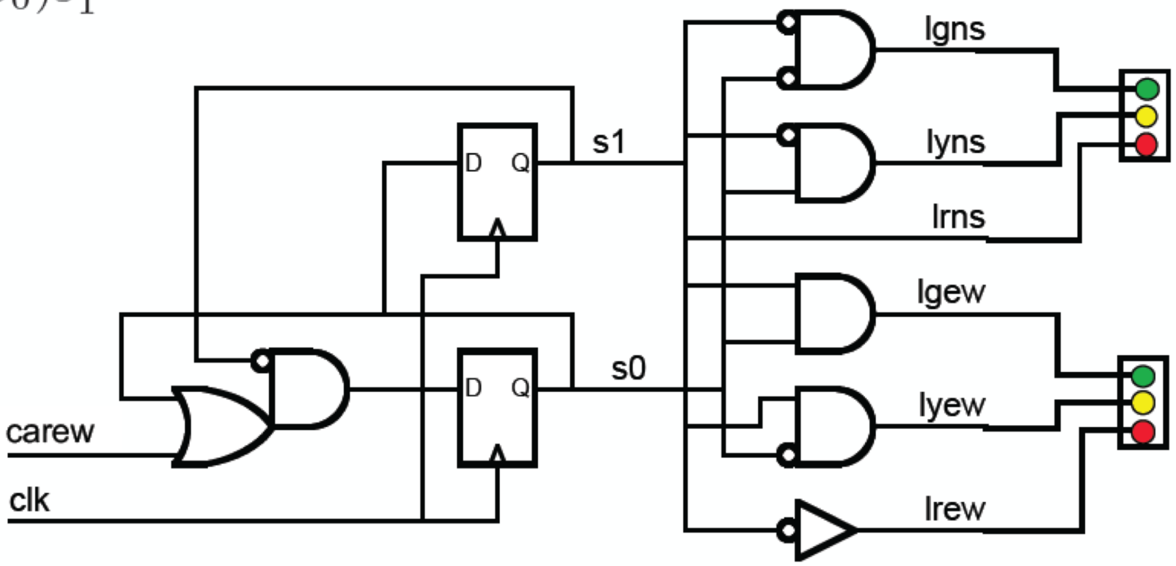
$$\mathrm{lyns} = s_1' s_0$$

$$\mathrm{lrns} = s_1$$

$$\mathrm{lgew} = s_1 s_0$$

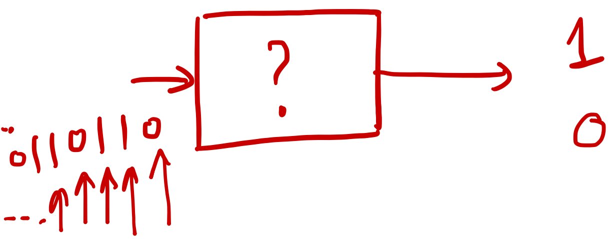
$$\mathrm{lyew} = s_1 s_0'$$

$$\mathrm{lrew} = s_1'$$



Sequence Recognizer Example (1/3)

- Example: A circuit that recognizes the sequence 1101



- one input : x

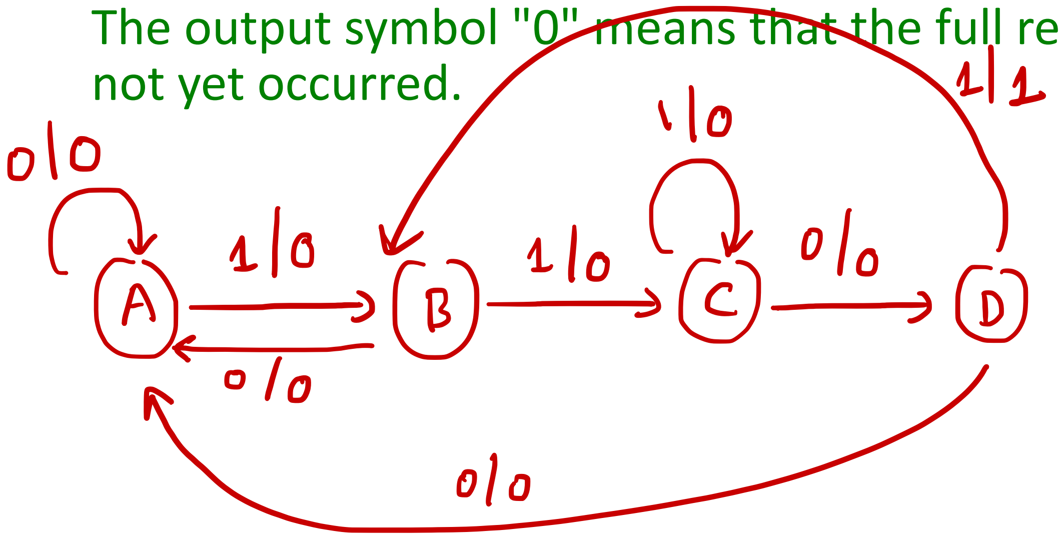
  one output: z.

  clock

  (reset, set)

Sequence Recognizer Example (2/3)

- Starting in the initial state (arbitrarily named "A"):
  - Add a state that recognizes the first ”1”.
  - State "A" is the initial state, and state "B" is the state which represents the fact that the "first" one in the input subsequence has occurred. The output symbol "0" means that the full recognized sequence has not yet occurred.



Mealy

  - Output 1 on the arc from D means the sequence has been recognized.

Sequence Recognizer Example (3/3)

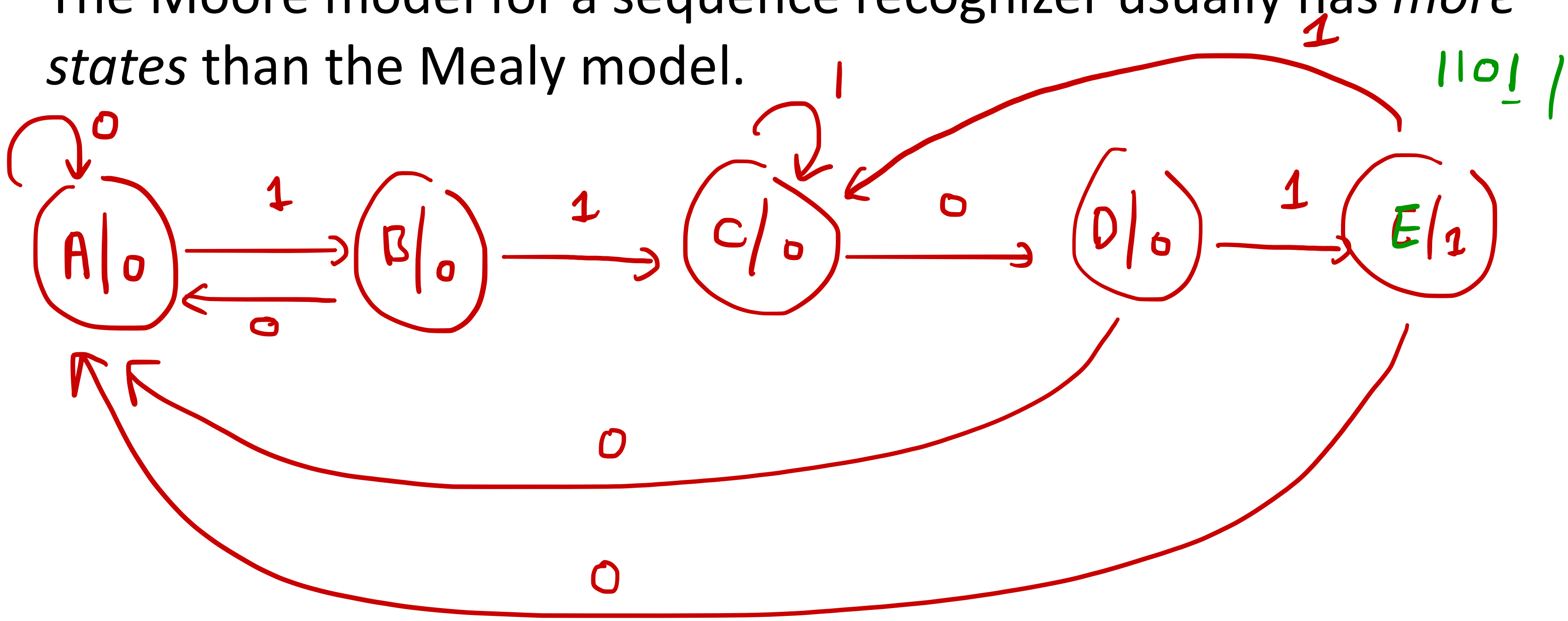
- The state have the following abstract meanings:
  - A: no proper sub-sequence of the sequence has occurred
  - B: the sub-sequence 1 has occurred
  - C: the sub-sequence 11 has occurred
  - D: the sub-sequence 110 has occurred
  - the 1/1 on the arc from D to B means that the last 1 has occurred and thus, the sequence is recognized

Formulation: Find State Table

Present State	Next State x=0	Next State x=1	Output x=0	Output x=1
A	A	B	0	0
B	A	C	0	0
C	D	C	0	0
D	A	B	0	1

Example: Moore Model for Sequence 1101

- For the Moore Model, outputs are associated with states.
- We need to add a state "E" with output value 1 for the final 1 in the recognized input sequence.
- The Moore model for a sequence recognizer usually has *more states* than the Mealy model.



Example: Moore Model (2/3)

- We mark outputs on states for Moore model.
- Arcs now show only state transitions.
- The new state, E produces the same behavior in the future as state B, but it gives a different output at the present time. Thus these states do represent a *different abstraction* of the input history.

Practice

Example: Moore Model (3/3)

- State table

Present State	Next State x=0	Next State x=1	Output y
A			
B			
C			
D			
E			

- Equations.

- logic diagram.

Sequence Recognition Design Example (1/2)

- Use Mealy model (p. 62)
- Use counting order state assignment

Present State (ab)		Next State $x=0$ (a^+b^+)	Next State $x=1$ (a^+b^+)	Output (z) $x=0$	Output (z) $x=1$
A	00	00	01	0	0
B	01	00	10	0	0
C	10	11	10	0	0
D	11	00	01	0	1

$x \rightarrow$? $\rightarrow z$

a	b	x	a^+	b^+	z
0	0	0			
0	0	1			
⋮	⋮	⋮			
1	1	1			

∘ Equations.

a^+

$x \backslash ab$	00	01	11	10
0	0	0	0	1
1	0	1	0	1

$$a^+ = a \cdot \bar{b} + x\bar{a}b$$

b^+

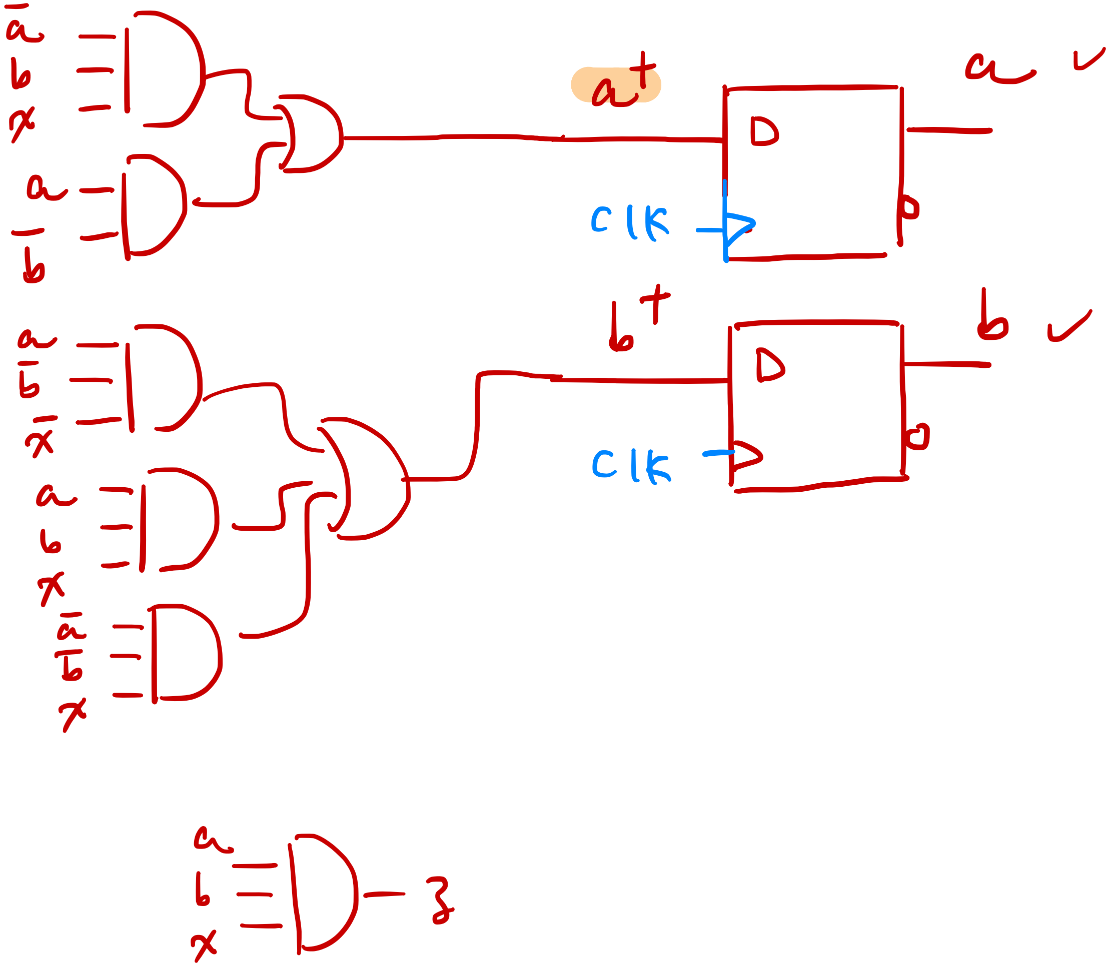
$x \backslash ab$	00	01	11	10
0	0	0	0	1
1	1	0	1	0

$$b^+ = \bar{x}a\bar{b} + x\bar{a}\bar{b} + xab$$

$x \backslash ab$	00	01	11	10
0	0	0	0	0
1	0	0	1	0

$$z = x \cdot a \cdot b$$

Sequence Recognition Design Example (2/2)

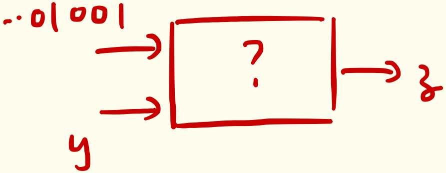


- Example. Design a serial odd parity generator.

Two inputs: x (serial data input)

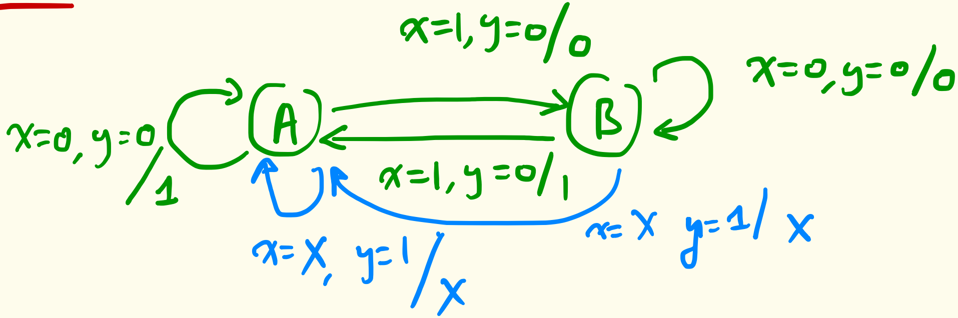
y ($y=1$ indicating the end of the sequence)

One output: z



State A: even number of 1's

State B: odd number of 1's



State table

Q	x	y	Q^t	z	J	K
0	0	0	0	1	?	?
0	0	1	0	X		
0	1	0	1	0		
0	1	1	0	X		
1	0	0	1	0		
1	0	1	0	X		
1	1	0	0	1		
1	1	1	0	X		

$Q \backslash xy$	00	01	11	10
0	0	0	0	1
1	1	0	0	0

$$Q^+ = Q\bar{x}\bar{y} + \bar{Q}x\bar{y}$$

$Q \backslash xy$	00	01	11	10
0	1	X	X	0
1	0	X	X	1

$$z = \bar{Q}\bar{x} + Qx$$

$J = f(Q, x, y)$

$K = f(Q, x, y)$