Digital Design with Verilog

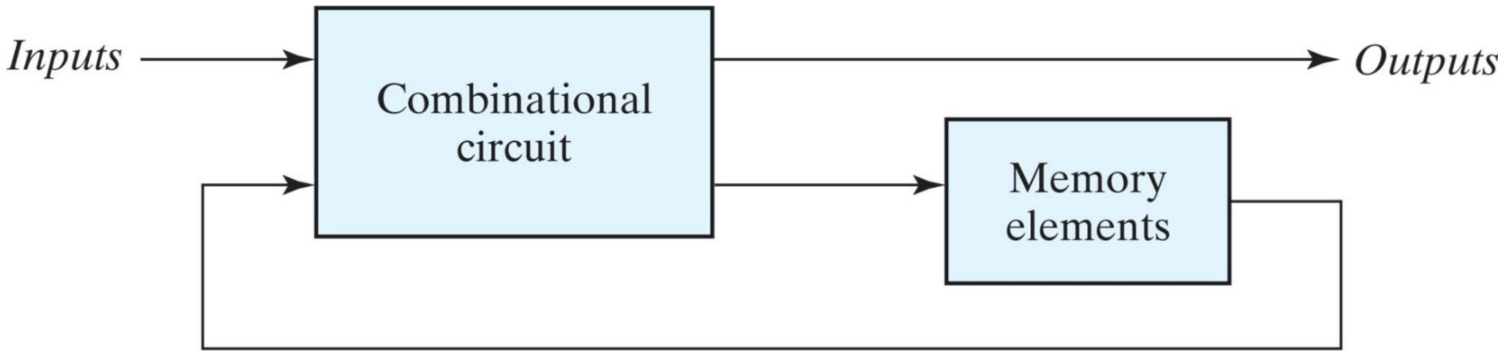
Lecture 3: Review of Sequential Logic Design

Objectives

- Storage Elements
  - Latches/ Flip-Flops
- Design of Sequential Machines
  - Mealy Type / Moore Type
  - State reduction techniques
- Design Examples
  - Counters
  - Shift registers
  - Sequence Detectors

Sequential Circuits

- Sequential logic includes storage elements.



- The binary information stored in these elements at any given time defines the *state* of the sequential circuit at that time.

Sequential Circuits

- The block diagram demonstrates that
  - *the outputs in a sequential circuit are a function not only of the inputs, but also of the present state of the storage elements.*
  - *The next state of the storage elements is also a function of external inputs and the present state.*
- Thus, **a sequential circuit is specified by a time sequence of inputs, outputs, and internal states** .
- In contrast, the outputs of combinational logic depend only on the present values of the inputs.

Types of Sequential Circuits

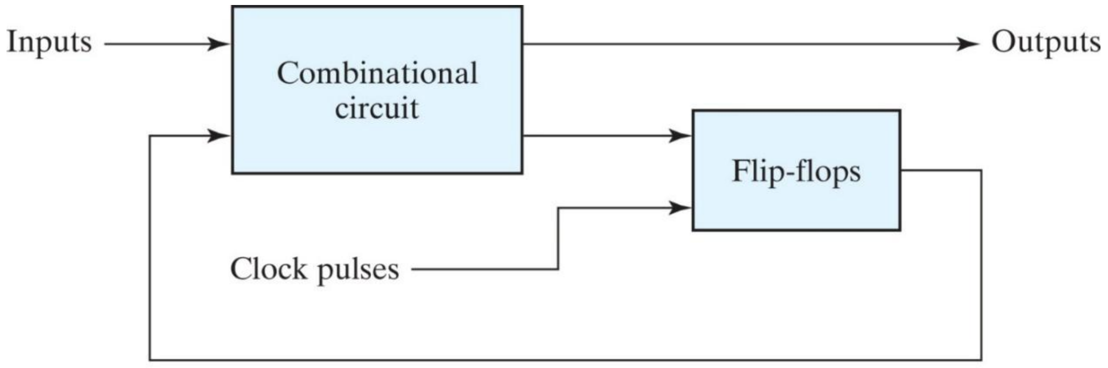
- There are two main types of sequential circuits, and their classification is a function of the timing of their signals.

- A *synchronous* sequential circuit is a system whose behavior can be defined from the knowledge of its signals at discrete instants of time.

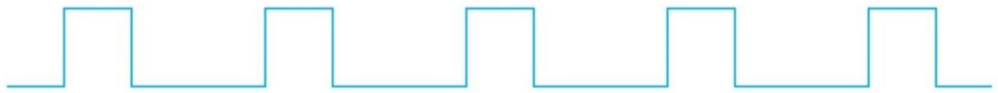
- The behavior of an *asynchronous* sequential circuit depends upon the input signals at any instant of time *and* the order in which the inputs change.

Synchronous Sequential Logic

- The storage elements (memory) used in clocked sequential circuits are called *flip-flops.*
- A flip-flop is a binary storage device capable of storing one bit of information.
  - In a stable state, the output of a flip-flop is either 0 or 1.



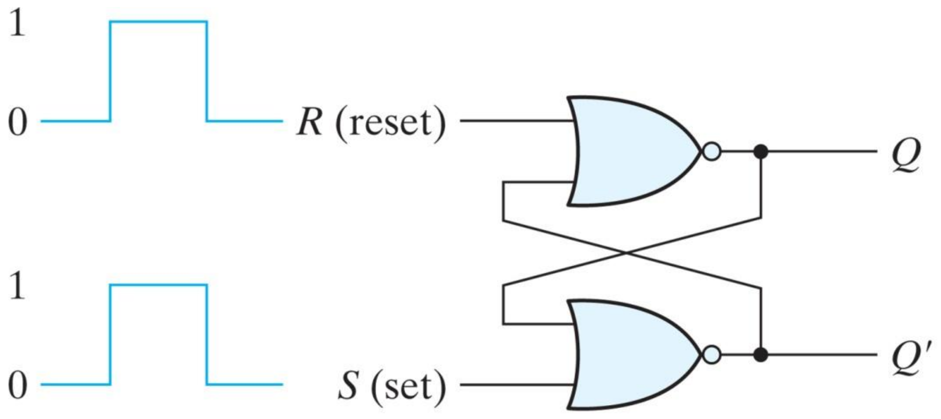
(a) Block diagram


(b) Timing diagram of clock pulses

Storage Elements: Latches and Flip-Flops

- Storage elements that operate with signal levels (rather than signal transitions) are referred to as *latches*; those controlled by a clock transition are *flip-flops*.
- Latches are said to be level sensitive devices; flip-flops are edge-sensitive devices.
- The two types of storage elements are related because latches are the basic circuits from which all flip-flops are constructed.

SR Latch



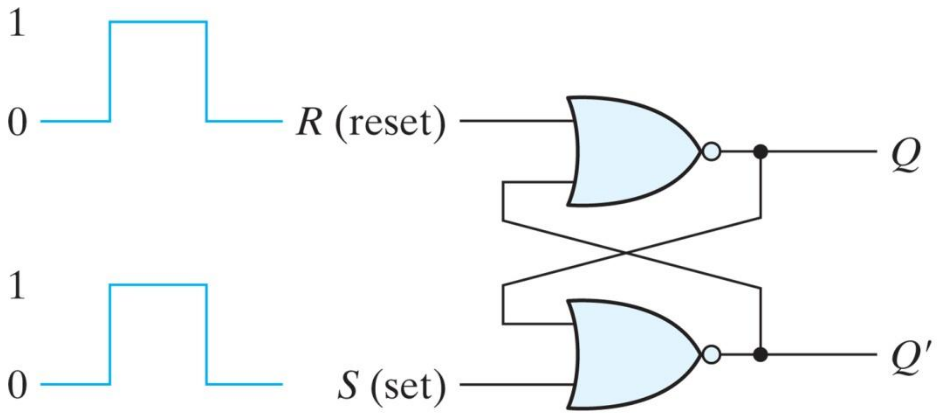
(a) Logic diagram

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

(b) Function table

- Under normal conditions, both inputs of the latch remain at 0 unless the state has to be changed.
- The application of a momentary 1 to the *S* input causes the latch to go to the set state.
- The *S* input must go back to 0 before any other changes take place.

SR Latch



(a) Logic diagram

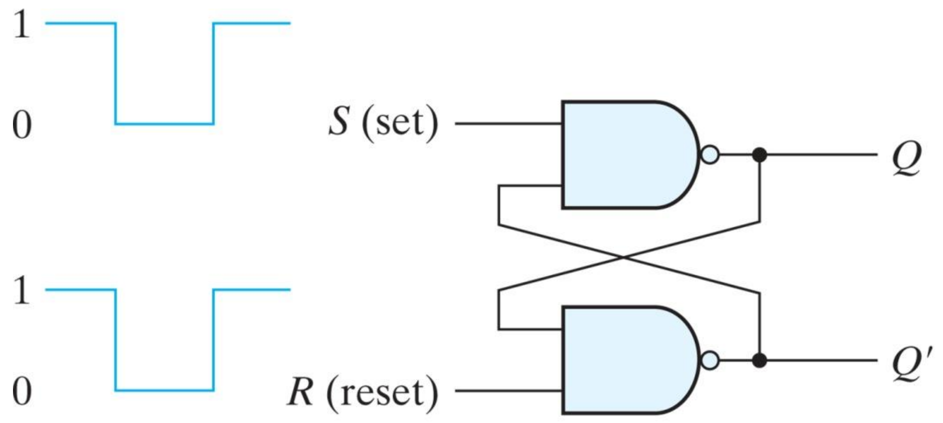
S	R	Q	Q'	
1	0	1	0	
0	0	1	0	(after $S = 1, R = 0$)
0	1	0	1	
0	0	0	1	(after $S = 0, R = 1$)
1	1	0	0	(forbidden)

(b) Function table

- After both inputs return to 0, it is then possible to shift to the reset state by momentary applying a 1 to the *R* input. The 1 can then be removed from *R*, where upon the circuit remains in the reset state.

- *Thus, when both inputs S and R are equal to 0, the latch can be in either the set or the reset state, depending on which input was most recently a 1.*

SR Latch / S’R’ Latch



(a) Logic diagram

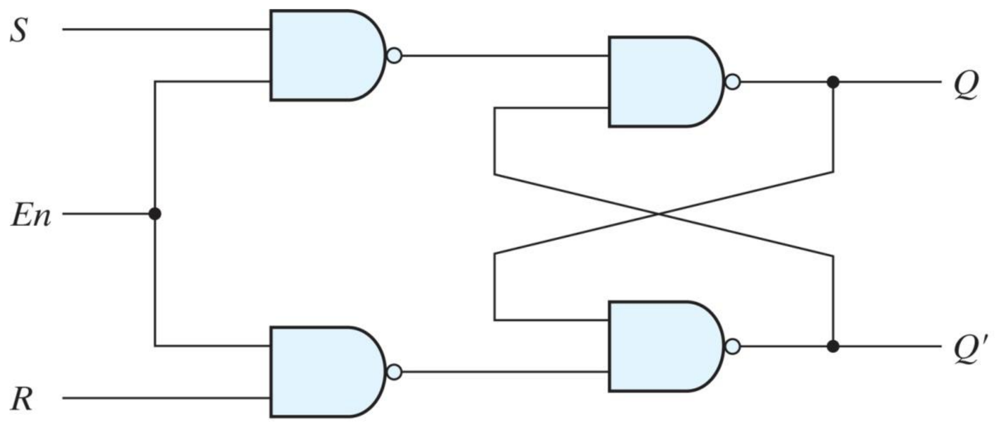
S	R	Q	Q'	
1	0	0	1	
1	1	0	1	(after $S = 1, R = 0$)
0	1	1	0	
1	1	1	0	(after $S = 0, R = 1$)
0	0	1	1	(forbidden)

(b) Function table

- SR latch with NAND gates requires a 0 signal to change its state.

- The inputs signals for the NAND-latch are the complement values used for the NOR latch.

SR Latch with control input



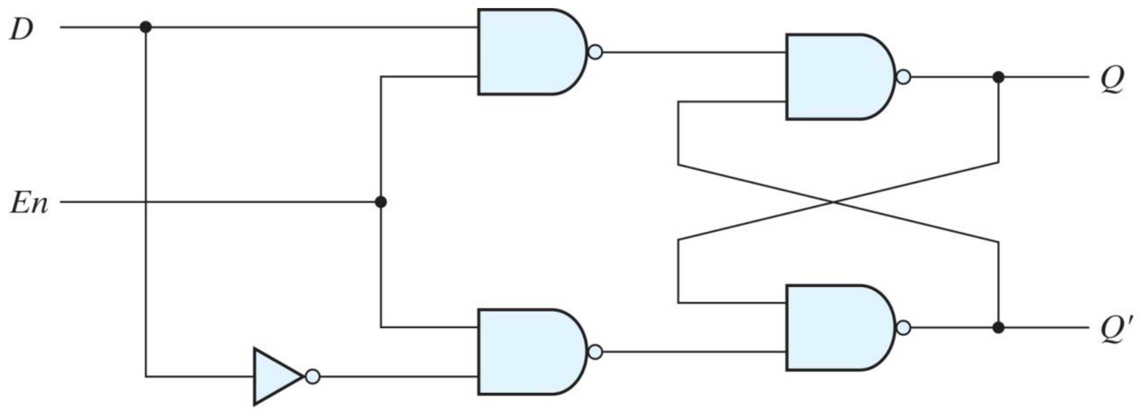
(a) Logic diagram

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

(b) Function table

- The outputs of the NAND gates stay at the logic-1 level as long as the enable signal remains at 0.

D Latch (Transparent Latch)



(a) Logic diagram

En	*D*	Next state of *Q*
0	X	No change
1	0	$Q = 0$; reset state
1	1	$Q = 1$; set state

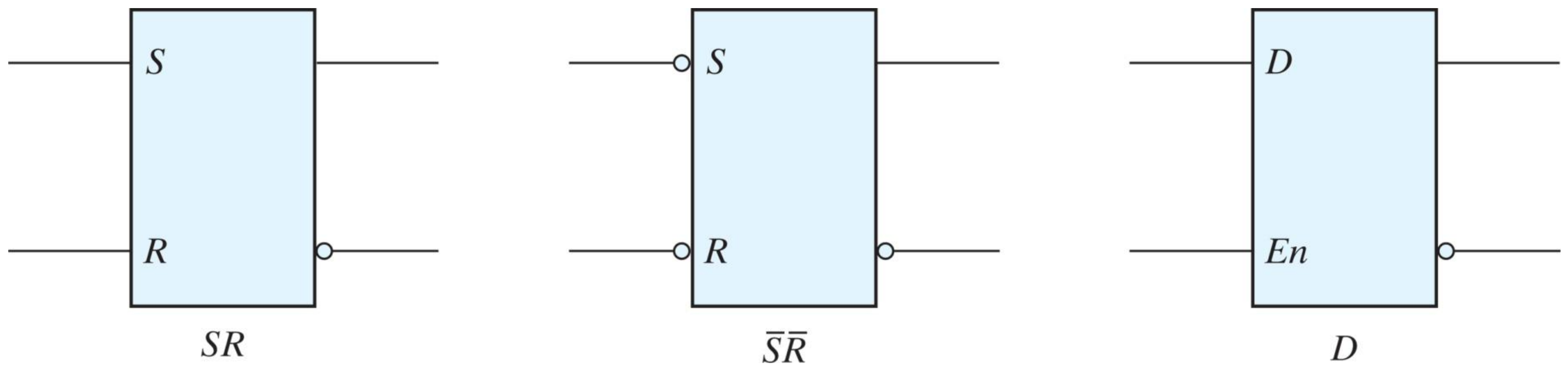
(b) Function table

- *How is D-latch structurally different than the SR latch?*
- *D* latch eliminates the undesirable condition of the indeterminate state that occurs in the *SR* latch (*Q* = *Q*’ = 1).

If *D* = 1, *Q* = 1 -> ‘set’ state

If *D* = 0, *Q* = 0 -> ‘reset’ state

Graphic Symbols for Latches



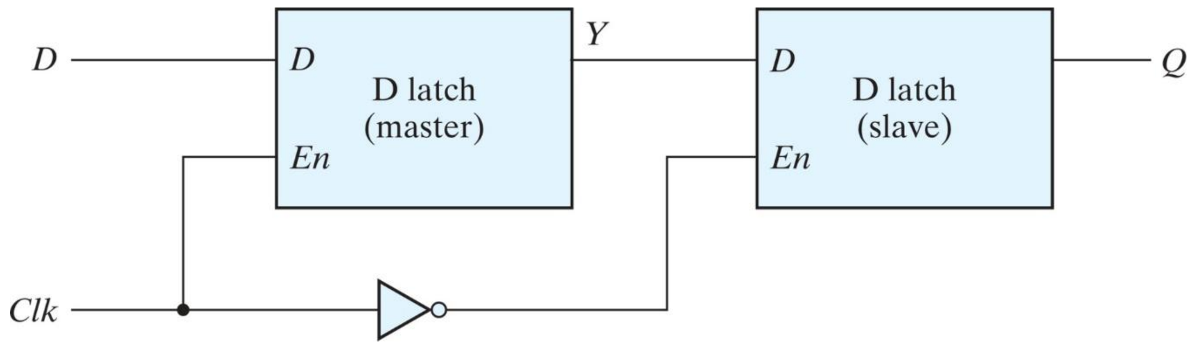
Flip-Flops

- A flip-flop is a state of a latch that can be switched by momentary change in the control input.

- This momentary change is called a trigger and the transition it causes is said to trigger the flip-flop.

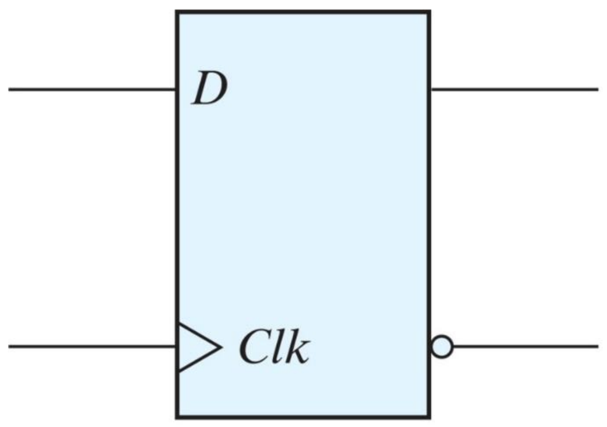
- The *D*-latch is a flip-flop that is triggered every time the pulse goes to a high or logic level 1.

Edge-Triggered Flip-Flop

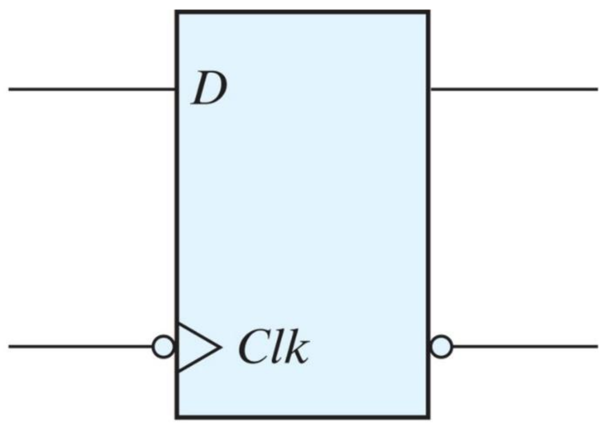


- The circuit samples the D input and changes its output at the negative edge of the clock, Clk.
- When the clock is 0, the output of the inverter is 1. The slave latch is enabled and its output *Q* is equal to the master output *Y*. The master latch is disabled (*Clk* = 0).
- When the *Clk* changes to high, *D* input is transferred to the master latch. The slave remains disabled as long as *En* is low. Any change in the input changes *Y*, but not *Q*.
- The output of the flip-flop can change when *CLK* makes a transition 1 $\rightarrow$ 0

Edge-Triggered Flip-Flop: Graphic Symbols



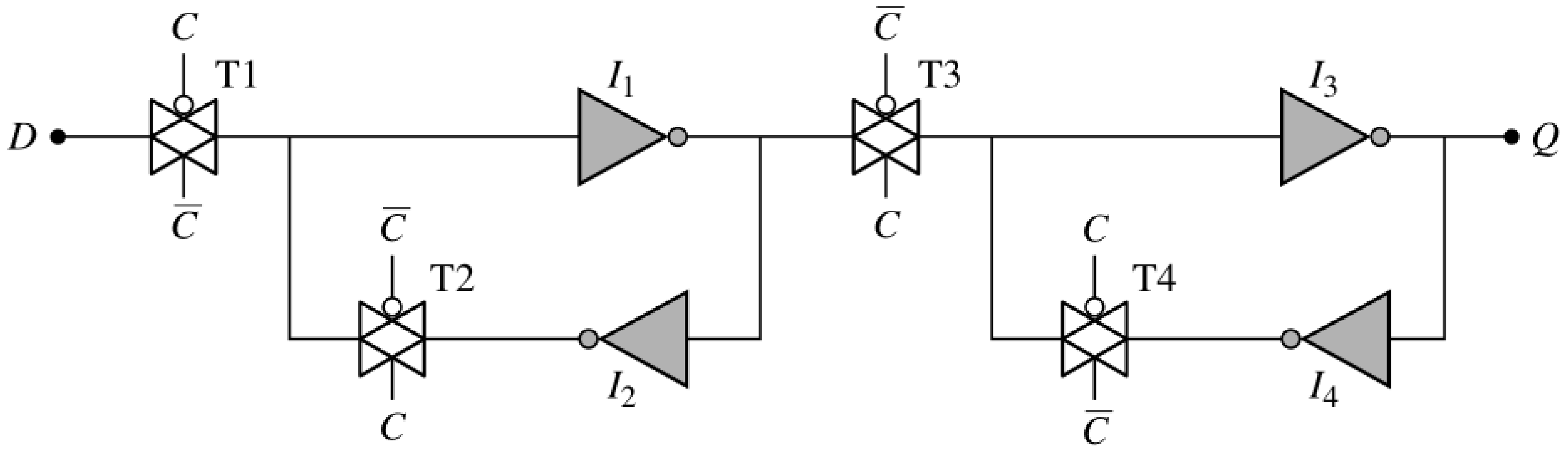
(a) Positive-edge



(a) Negative-edge

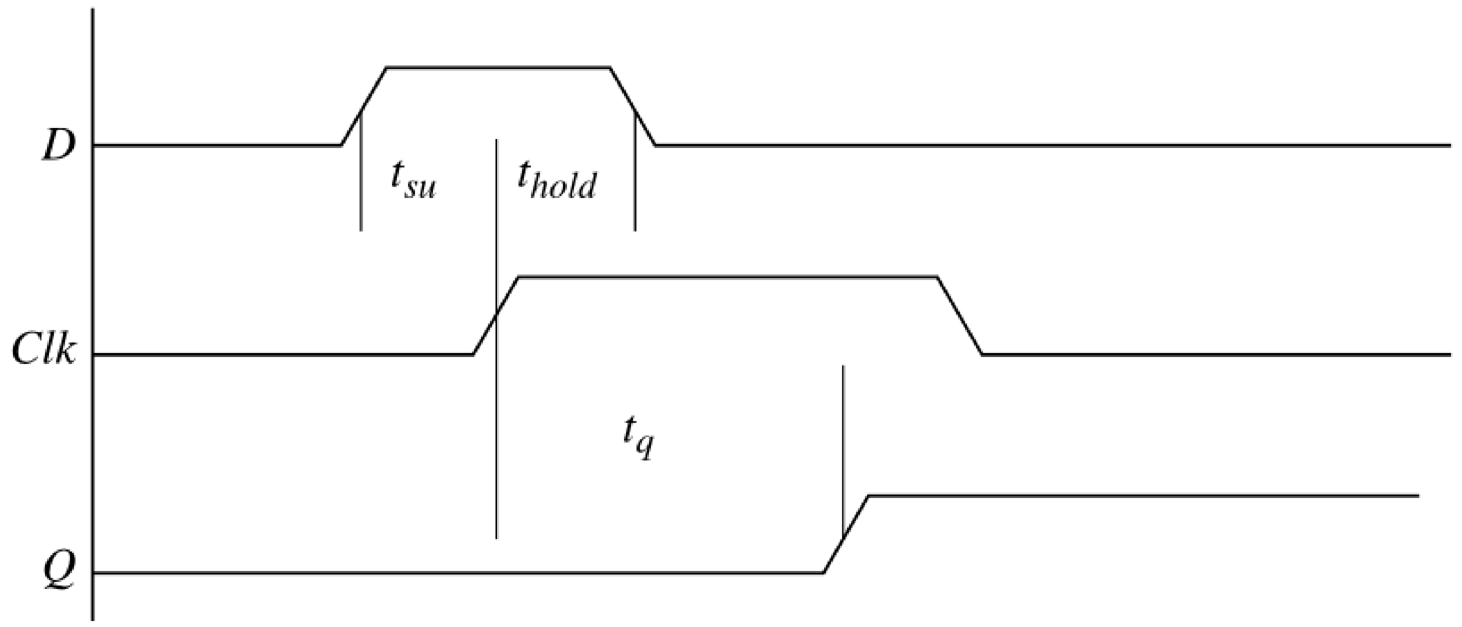
- The most economical and efficient flip-flop constructed is the edge-triggered *D* flip-flop since it requires the smallest number of gates.

CMOS Master Slave Circuit



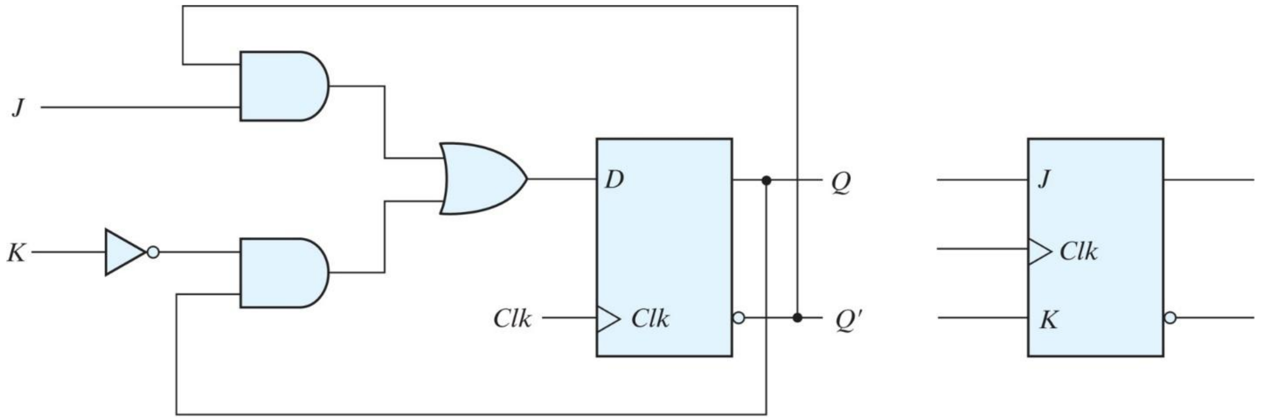
CMOS master–slave circuit of a D-type flip-flop: circuit schematic

CMOS Master Slave Circuit



Timing definitions for D, Clk, and Q waveforms.

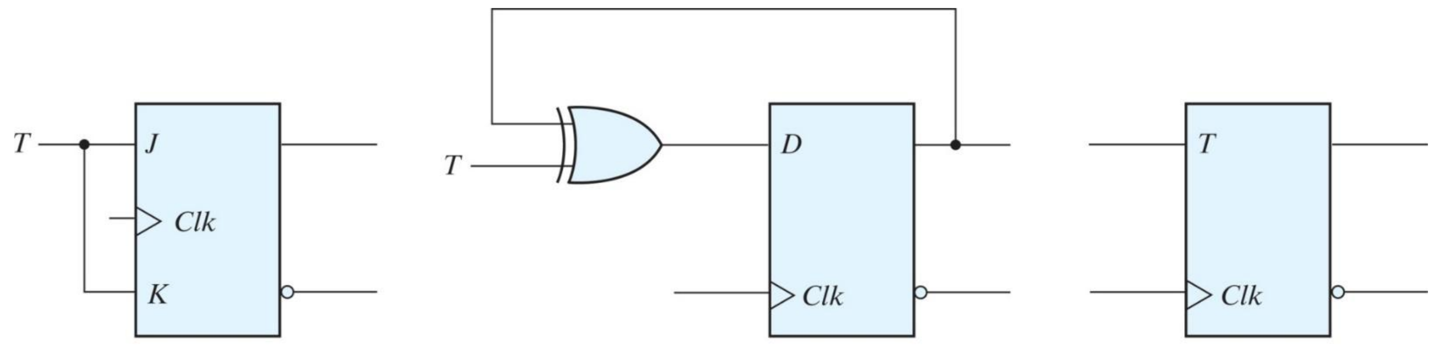
JK Flip-Flop



(a) Circuit diagram (b) Graphic symbol

- When $J = 1$ and $K = 0$, $D = 1$ $\longrightarrow$ next clock edge sets output to 1.
- When $J = 0$ and $K = 1$, $D = 0$ $\longrightarrow$ next clock edge resets output to 0.
- When $J = 1$ and $K = 1$, $D = Q'$ $\longrightarrow$ next clock edge complements output.
- When $J = 0$ and $K = 0$, $D = Q$ $\longrightarrow$ next clock edge leaves output unchanged.

T Flip-Flop



(a) From *JK* flip-flop (b) From *D* flip-flop (c) Graphic symbol

- The *T* (toggle) flip-flop is a complementing flip-flop and can be obtained from a *JK* flip-flop when inputs *J* and *K* are tied together.
- When *T* = 0 (*J* = *K* = 0), a clock edge does not change the output. When *T* = 1 (*J* = *K* = 1), a clock edge complements the output.
- The *T* flip-flop can also be constructed with a *D* flip-flop and an exclusive-OR gate.

Characteristic Tables and Equations

Flip-Flop Characteristic Tables

JK Flip-Flop

J	*K*	Q (t + 1)	
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$Q'(t)$	Complement

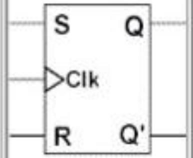
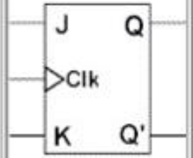
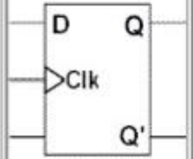
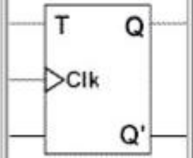
D Flip-Flop

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

T Flip-Flop

T	Q (t + 1)	
0	$Q(t)$	No change
1	$Q'(t)$	Complement

Excitation Tables

FLIP-FLOP NAME	FLIP-FLOP SYMBOL	CHARACTERISTIC TABLE	CHARACTERISTIC EQUATION	EXCITATION TABLE
SR		see below	$Q_{(next)} = S + R'Q$ $SR = 0$	see below
JK		see below	$Q_{(next)} = JQ' + K'Q$	see below
D		see below	$Q_{(next)} = D$	see below
T		see below	$Q_{(next)} = TQ' + T'Q$	see below

SR characteristic table:

S	R	Q(next)
0	0	Q
0	1	0
1	0	1
1	1	?

SR excitation table:

Q	Q(next)	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

JK characteristic table:

J	K	Q(next)
0	0	Q
0	1	0
1	0	1
1	1	Q'

JK excitation table:

Q	Q(next)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

D characteristic table:

D	Q(next)
0	0
1	1

D excitation table:

Q	Q(next)	D
0	0	0
0	1	1
1	0	0
1	1	1

T characteristic table:

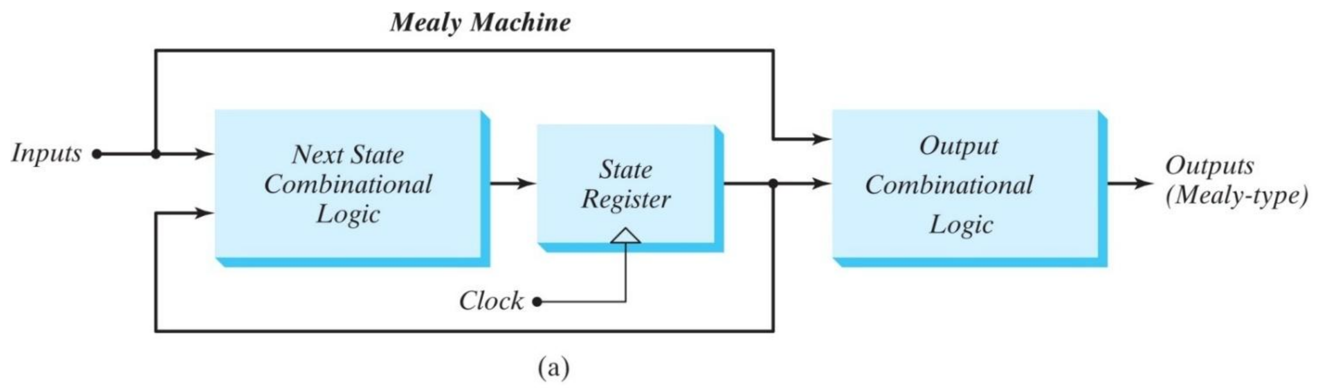
T	Q(next)
0	Q
1	Q'

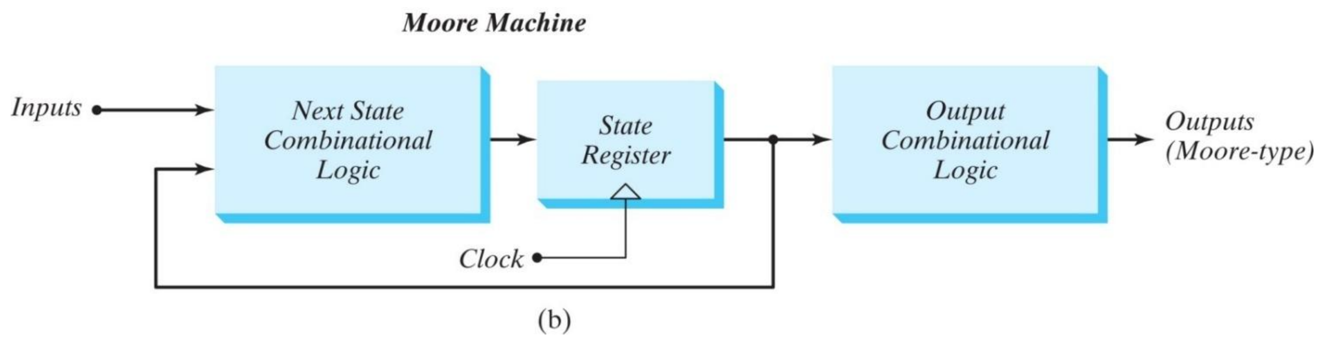
T excitation table:

Q	Q(next)	T
0	0	0
0	1	1
1	0	1
1	1	0

Mealy and Moore Models of FSMs

- Mealy model, the output is a function of both the present state and the input.
- Moore model, the output is a function of only present state





State Reduction and Equivalent States

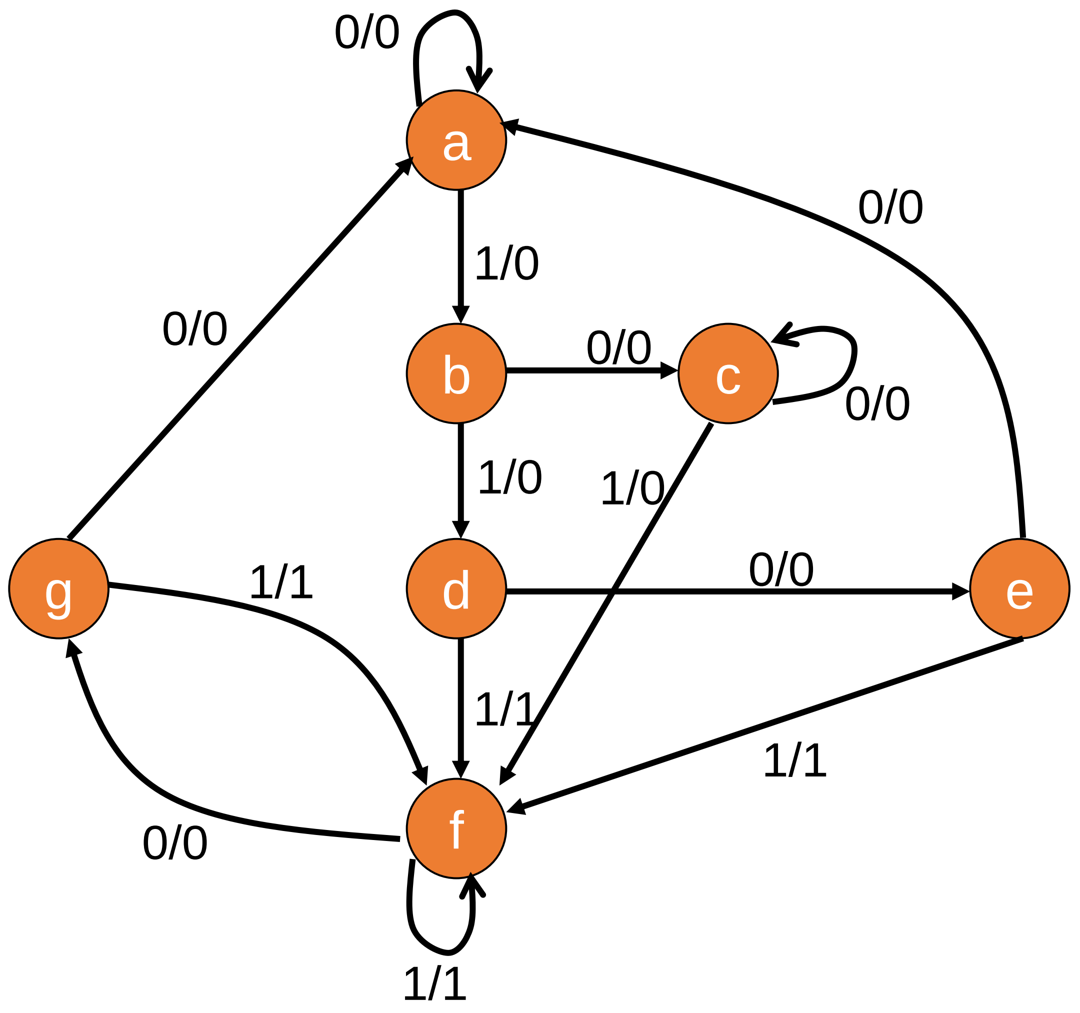
State Reduction and Equivalent States

- Two states of a sequential machine are equivalent (≡) if they have the same output sequence for all possible input sequences.

- Such states of the machine cannot be distinguished from each other based on observed outputs.

- Equivalent states can be combined without changing the input-output behavior of the machine.
  - Leads to a reduction in hardware without compromising functionality

Example: State Reduction



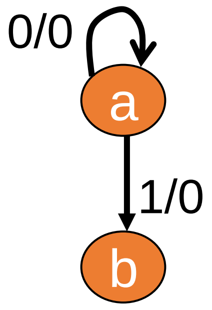
Note that we use letters to designate the states for the time being

Example: State Reduction

state	a	a	b	c	f	g	f	f	g	a	a		
input	0	1	0	1	0	1	1	0	0	0	0		
output	0	0	0	0	0	1	1	0	0	0			

- What is important
  - not the states
  - but the output values the circuit generates
- Therefore, the problem is to find a circuit
  - with fewer number of states,
  - but that produces the same output pattern for any given input pattern, starting with the same initial state

State Reduction Technique 1/7



- Step 1: get a state table

present state	next state		Output	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	c	d	0	0
c	c	f	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

State Reduction Technique 2/7

- Step 2: Inspect the state table for equivalent states
  - Equivalent states: Two states,
    1. that produce the same output
    2. whose next states are identical

   for each input combination

State Reduction Technique 3/7

present state	next state		Output	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	c	d	0	0
c	c	f	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

- States “e” and “g” are equivalent
- One of them can be removed

State Reduction Technique 4/7

present state	next state		Output	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	c	d	0	0
c	c	f	0	0
d	e	f	0	1
e	a	f	0	1
f	e	f	0	1

- We keep looking for equivalent states

State Reduction Technique 5/7

present state	next state		Output	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	c	d	0	0
c	c	d	0	0
d	e	d	0	1
e	a	d	0	1

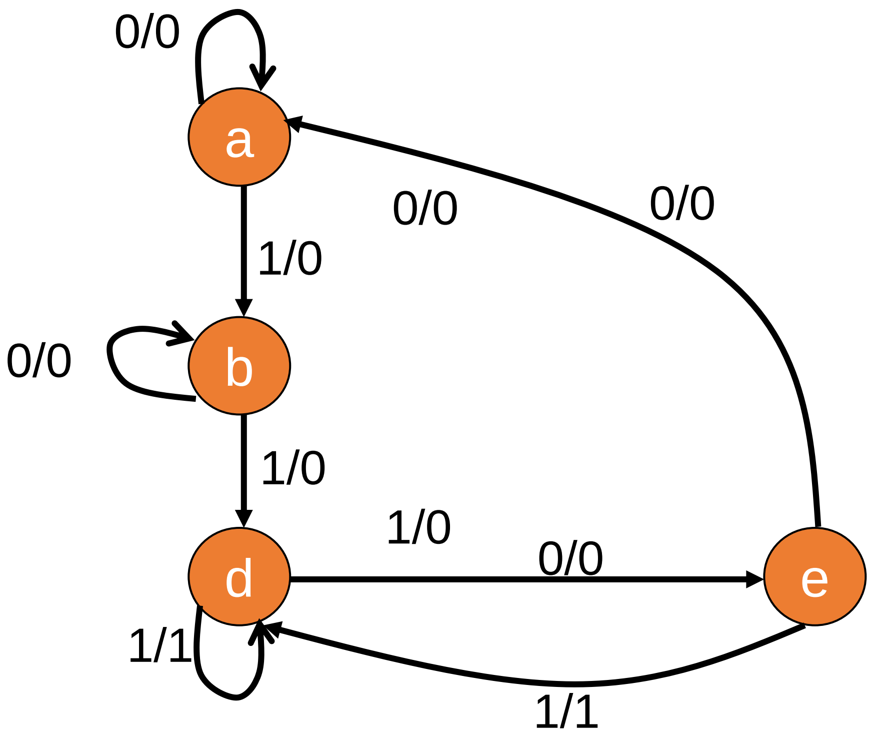
- We keep looking for equivalent states

State Reduction Technique 6/7

present state	next state		Output	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	b	d	0	0
d	e	d	0	1
e	a	d	0	1

- We stop when there are no equivalent states

State Reduction Technique 7/7



present state	next state		Output	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	b	d	0	0
d	e	d	0	1
e	a	d	0	1

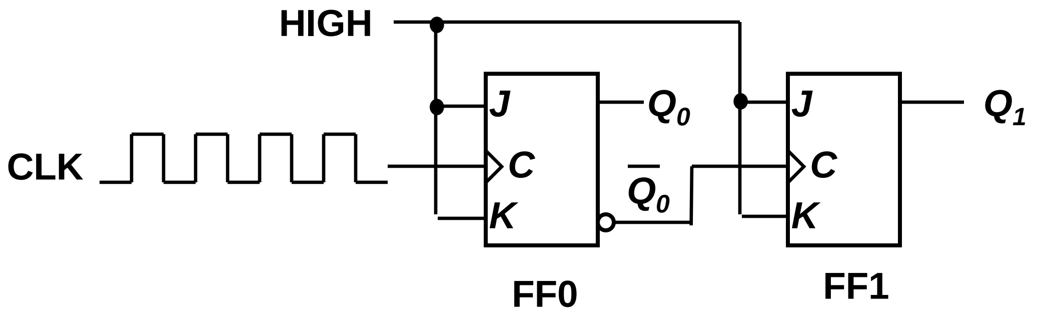
We need two flip-flops

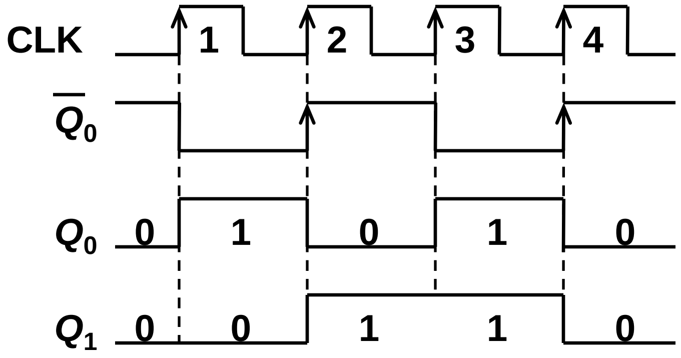
state	a	a	b	b	d	e	d	d	e	d	e		
input	0	1	0	1	0	1	1	0	1	0	0		
output	0	0	0	0	0	1	1	0	0	0	0		

Counters

Asynchronous (Ripple) Counters

- Example: 2-bit ripple binary counter.
- Output of one flip-flop is connected to the clock input of the next more-significant flip-flop.



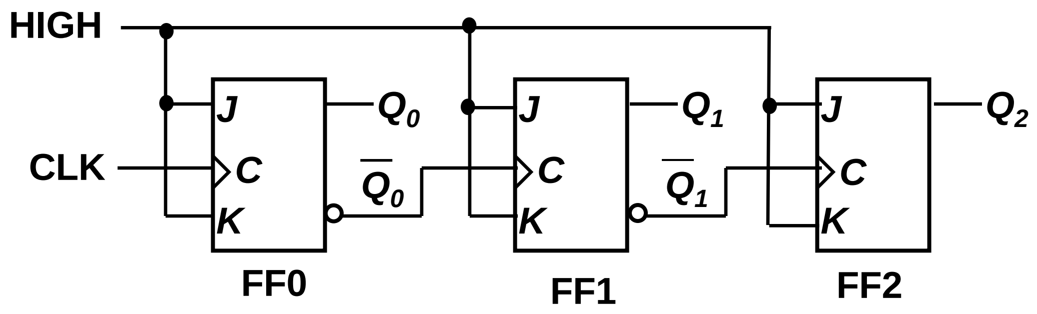


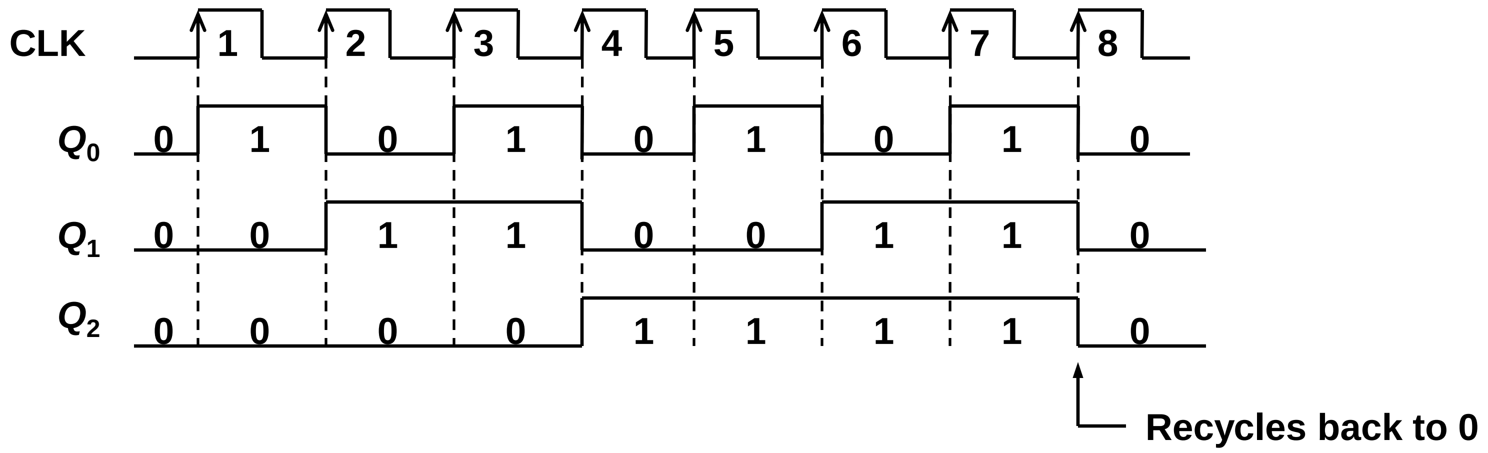
Timing diagram

$00 \rightarrow 01 \rightarrow 10 \rightarrow 11 \rightarrow 00 \ldots$

Asynchronous (Ripple) Counters

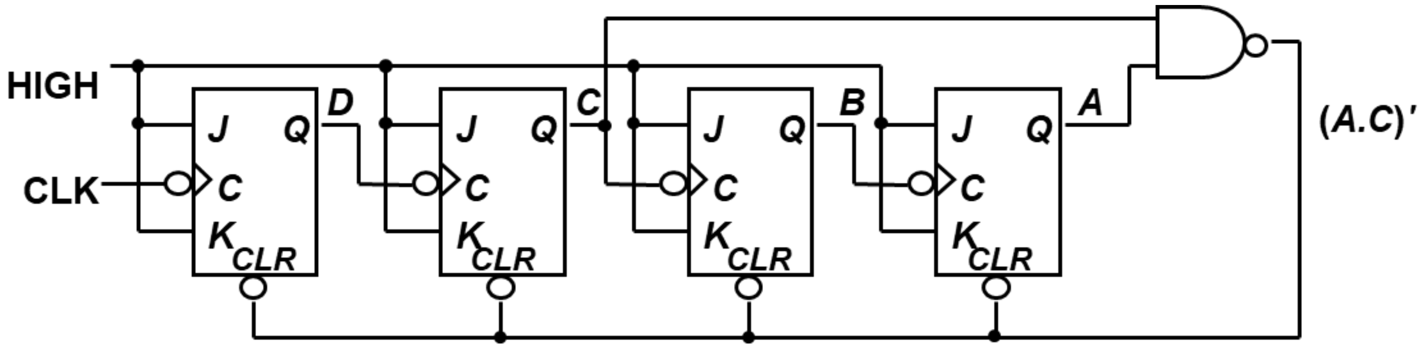
- Example: 3-bit ripple binary counter.

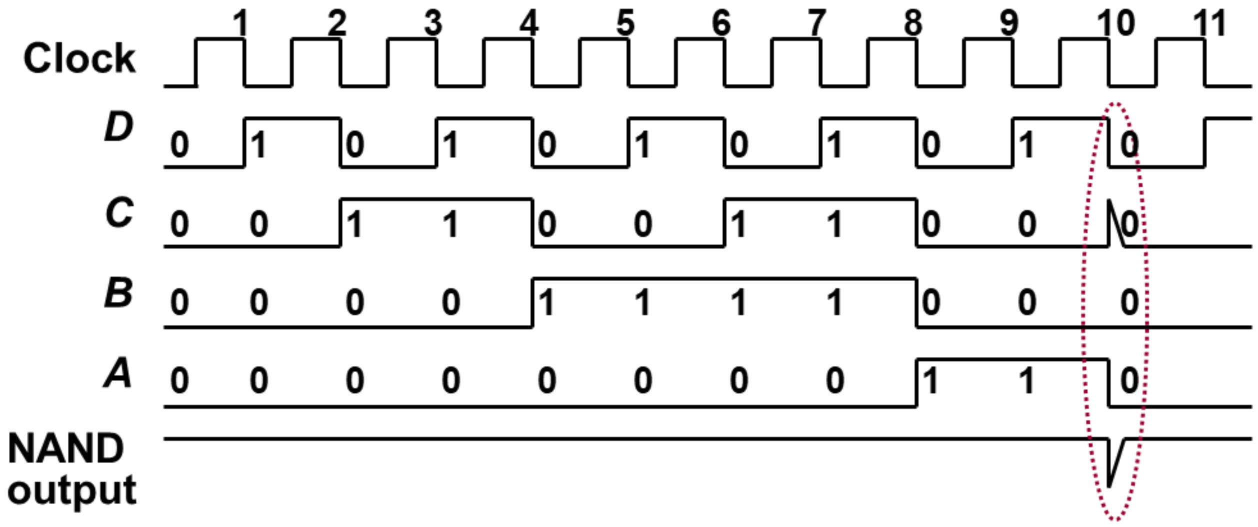




Asyn. Counters with MOD no. < 2^n

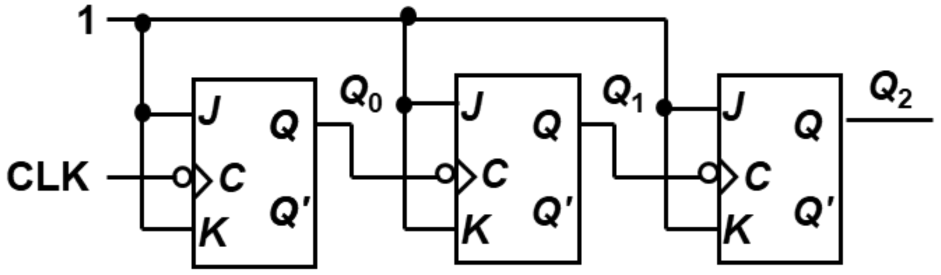
- Asynchronous decade/BCD counter (cont'd).



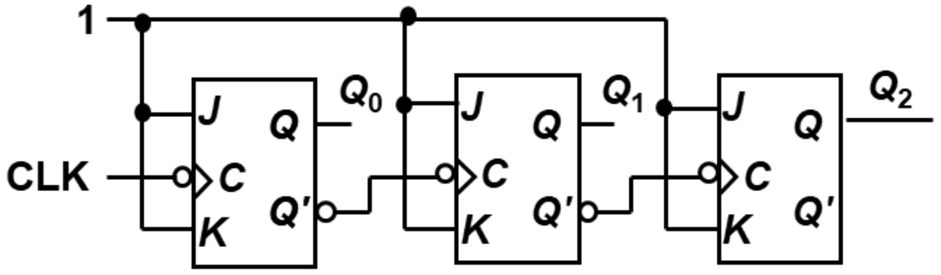


Asynchronous Down Counters

- So far we are dealing with up counters. Down counters, on the other hand, count downward from a maximum value to zero, and repeat.
- Example: A 3-bit binary (MOD-8) down counter.



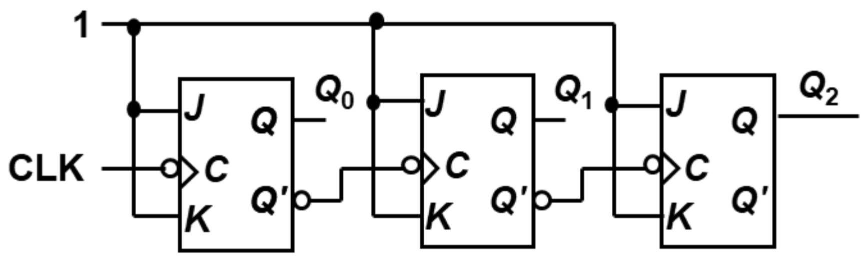
3-bit binary up counter

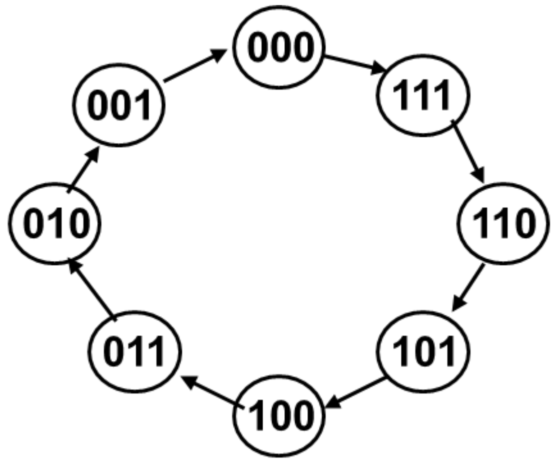


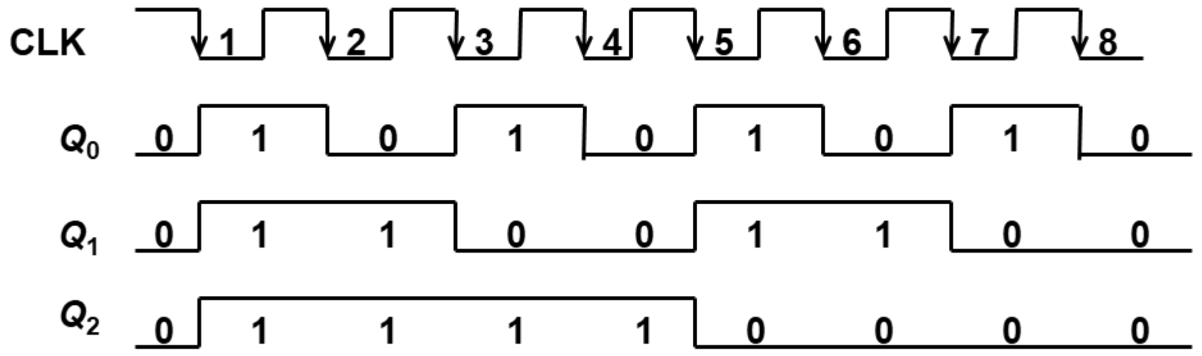
3-bit binary down counter

Asynchronous Down Counters

- Example: A 3-bit binary (MOD-8) down counter.



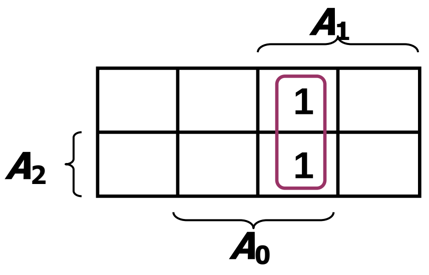




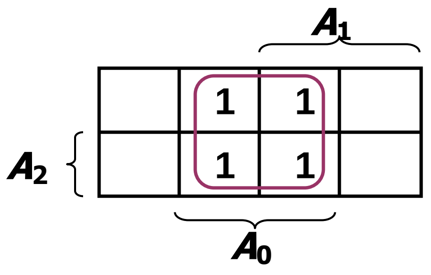
Synchronous Counters

- Example: 3-bit synchronous binary counter (using T flip-flops, or JK flip-flops with identical J, K inputs).

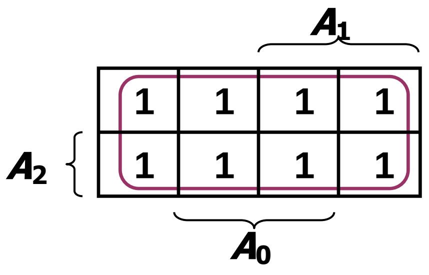
Present state			Next state			Flip-flop inputs		
A_2	A_1	A_0	A_2^+	A_1^+	A_0^+	TA_2	TA_1	TA_0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1



$TA_2 = A_1.A_0$



$TA_1 = A_0$

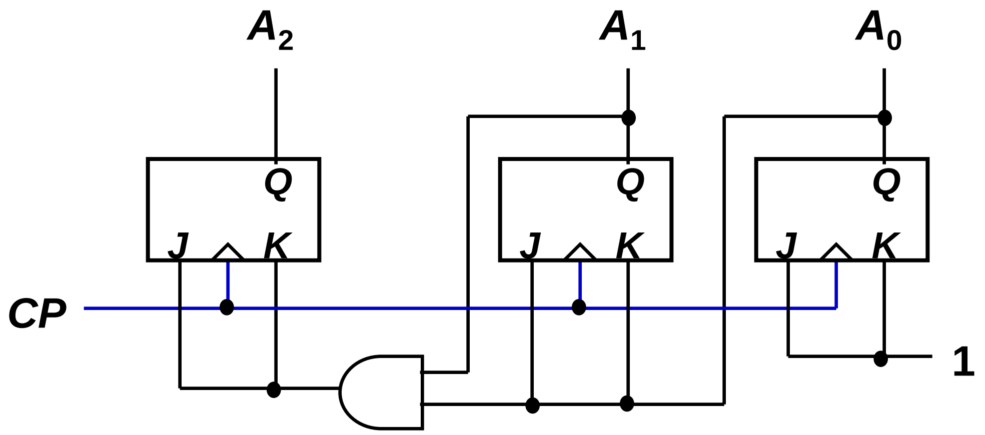


$TA_0 = 1$

Synchronous Counters

- Example: 3-bit synchronous binary counter (cont'd).

$TA_2 = A_1.A_0$ $TA_1 = A_0$ $TA_0 = 1$



Up/Down Synchronous Counters

- Example: A 3-bit up/down synchronous binary counter.

Clock pulse	*Up*	Q_2	Q_1	Q_0	*Down*
0		0	0	0	
1		0	0	1	
2		0	1	0	
3		0	1	1	
4		1	0	0	
5		1	0	1	
6		1	1	0	
7		1	1	1	

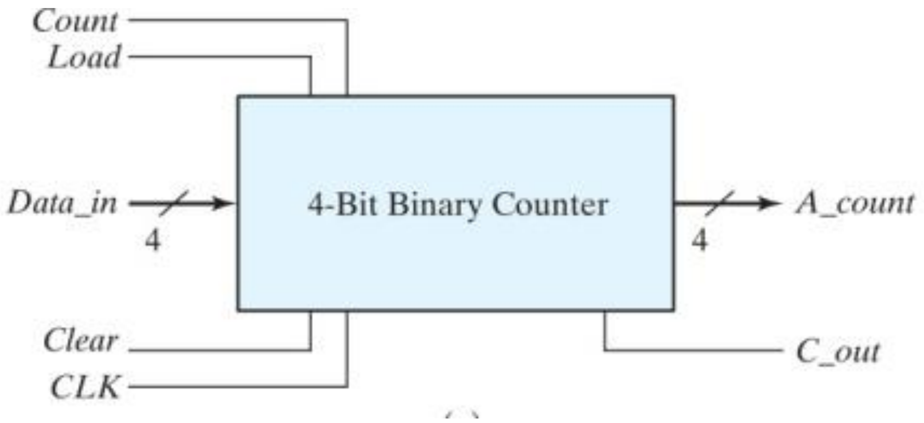
$TQ_0 = 1$

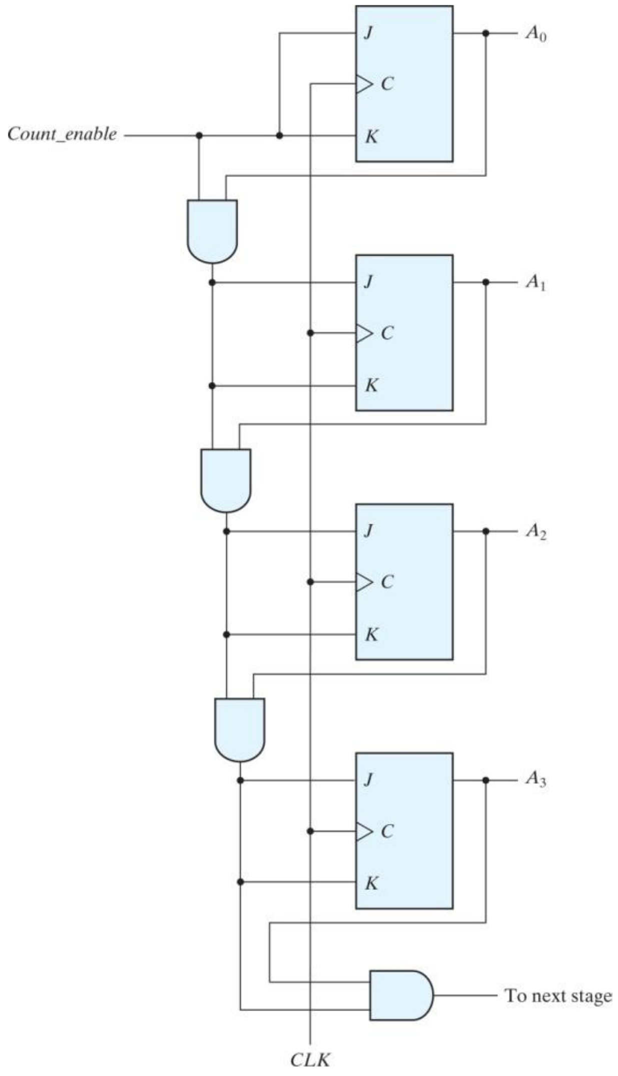
$TQ_1 = (Q_0.Up) + (Q_0'.Up')$

$TQ_2 = (Q_0.Q_1.Up) + (Q_0'.Q_1'.Up')$

Up counter	Down counter
$TQ_0 = 1$	$TQ_0 = 1$
$TQ_1 = Q_0$	$TQ_1 = Q_0'$
$TQ_2 = Q_0.Q_1$	$TQ_2 = Q_0'.Q_1'$

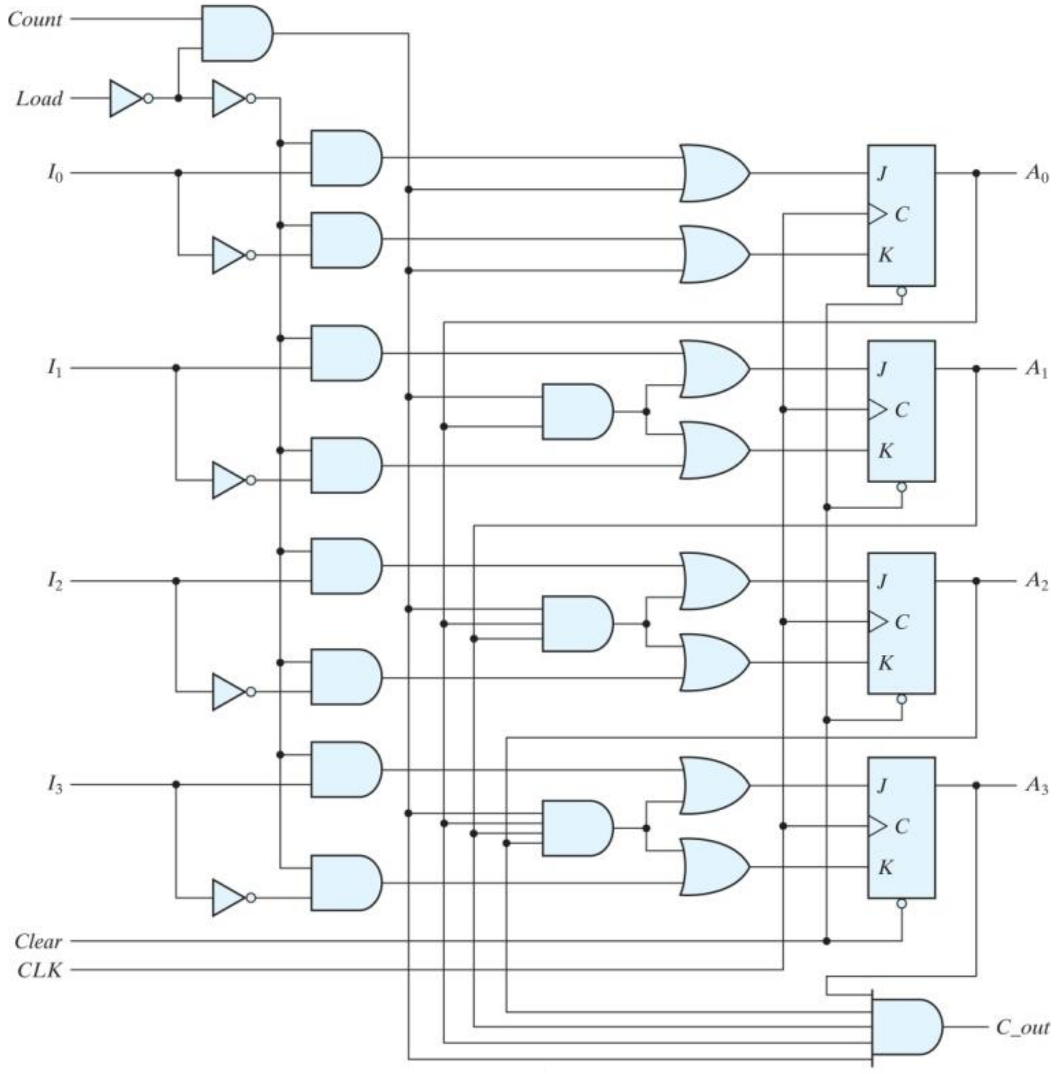
Counters with Parallel Load





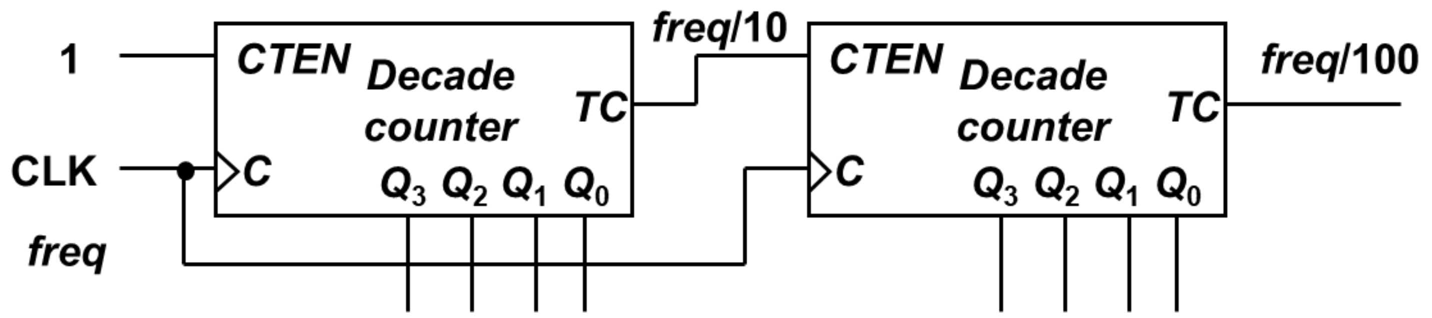
Clear	CLK	Load	Count	Function
0	X	X	X	Clear to 0
1	↑	1	X	Load inputs
1	↑	0	1	Count next binary state
1	↑	0	0	No change

Counters with Parallel Load



Cascading Synchronous Counters

- If counter is a not a binary counter, requires additional output.
- Example: A modulus-100 counter using two-decade counters.



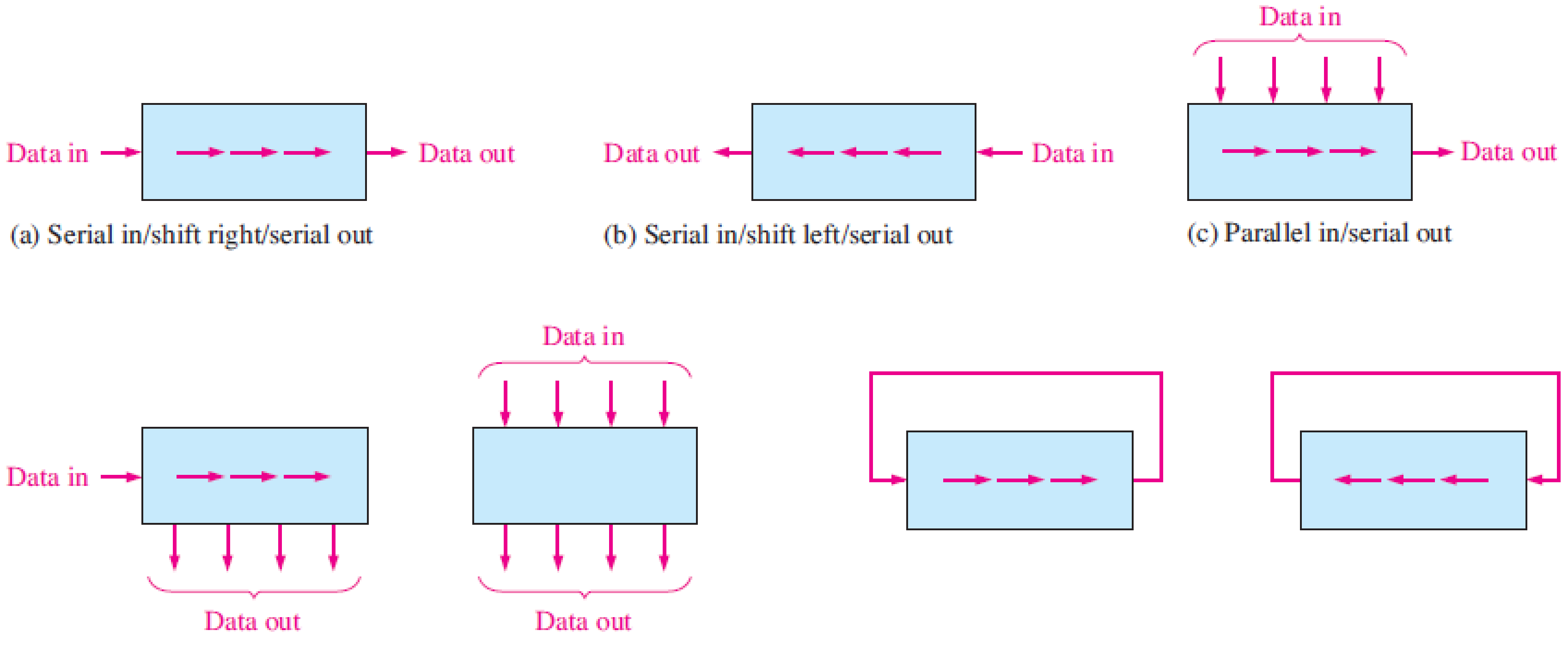
TC = 1 when counter recycles to 0000

- CTEN---Counter Enable, TC= Terminal Count

Shift Registers

Shift Registers

- Basic data movement in shift registers



(a) Serial in/shift right/serial out
(b) Serial in/shift left/serial out
(c) Parallel in/serial out
(d) Serial in/parallel out
(e) Parallel in/parallel out
(f) Rotate right
(g) Rotate left

Serial In/Serial Out Shift Registers

- Accepts data serially – one bit at a time – and also produces output serially.

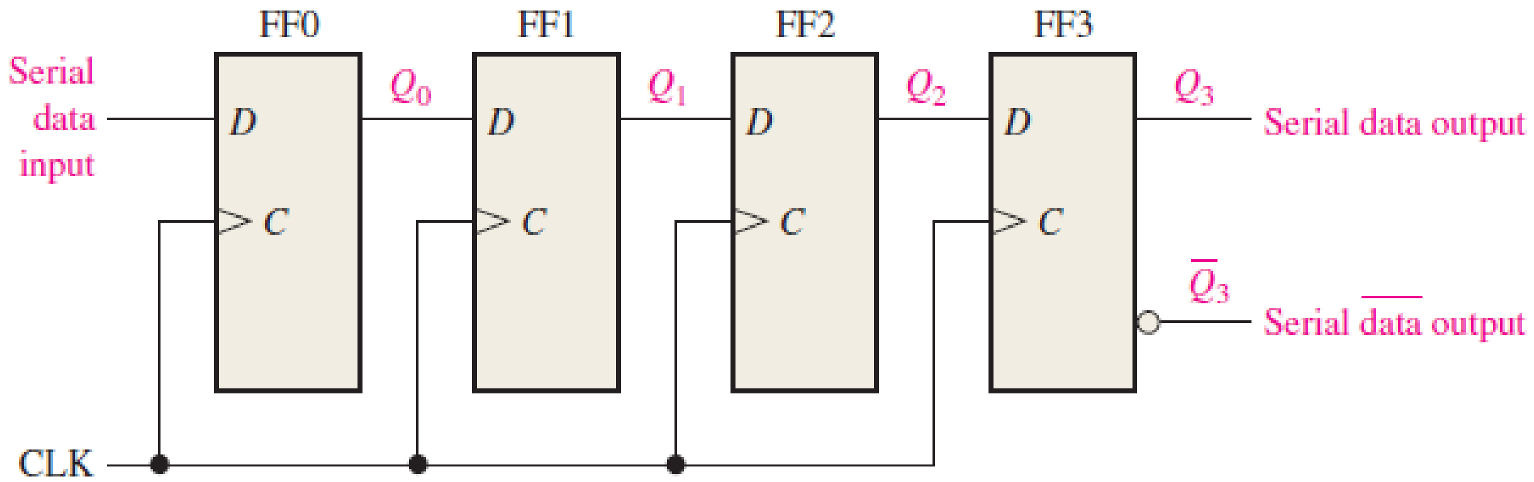
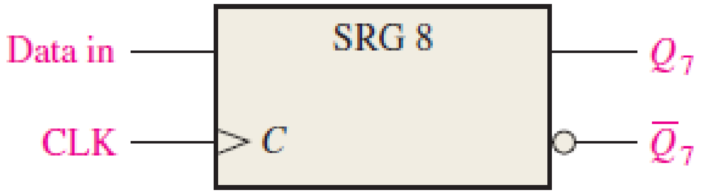
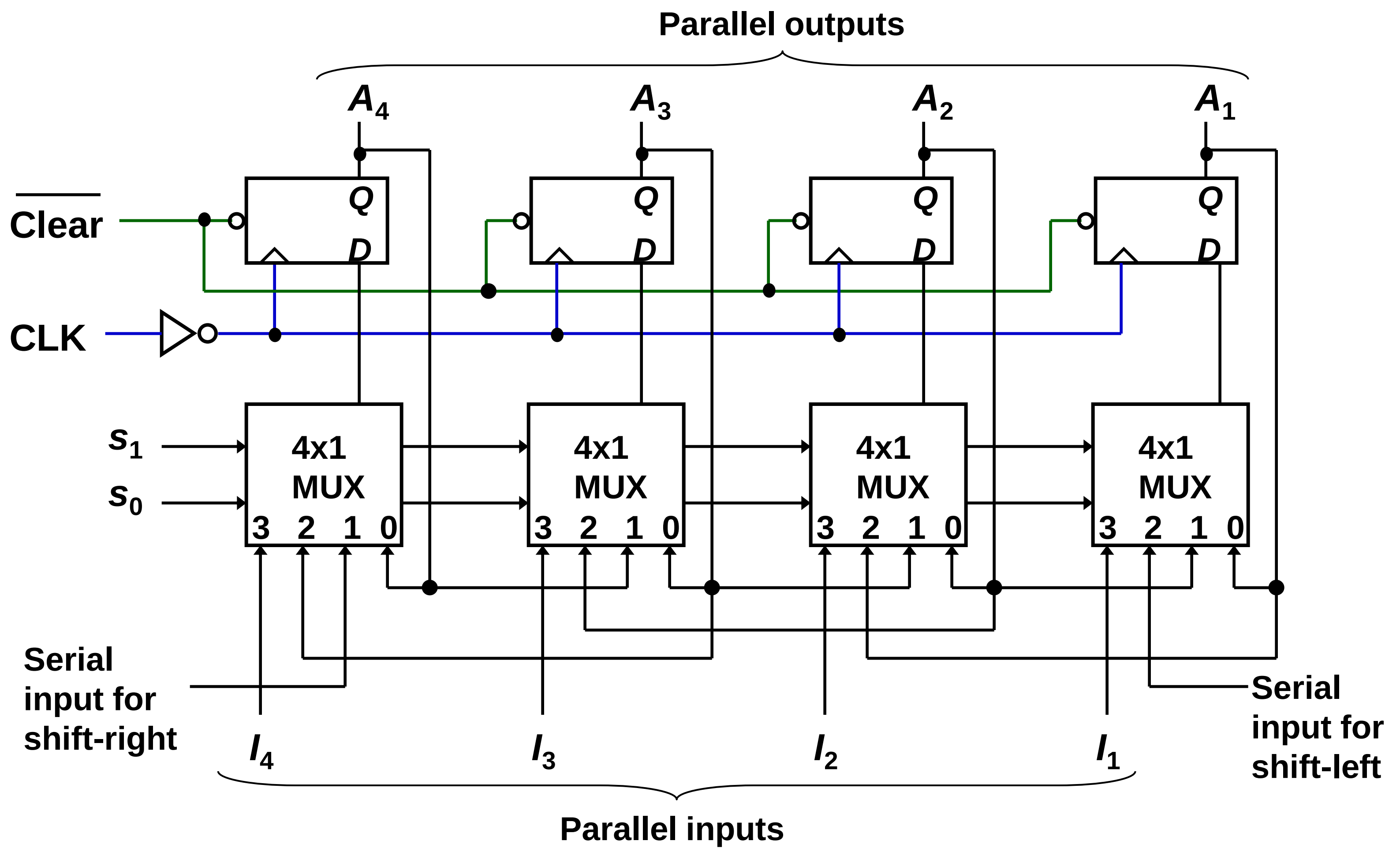


FIGURE 8–3 Serial in/serial out shift register.



Bidirectional Shift Registers

- 4-bit bidirectional shift register with parallel load.



Bidirectional Shift Registers

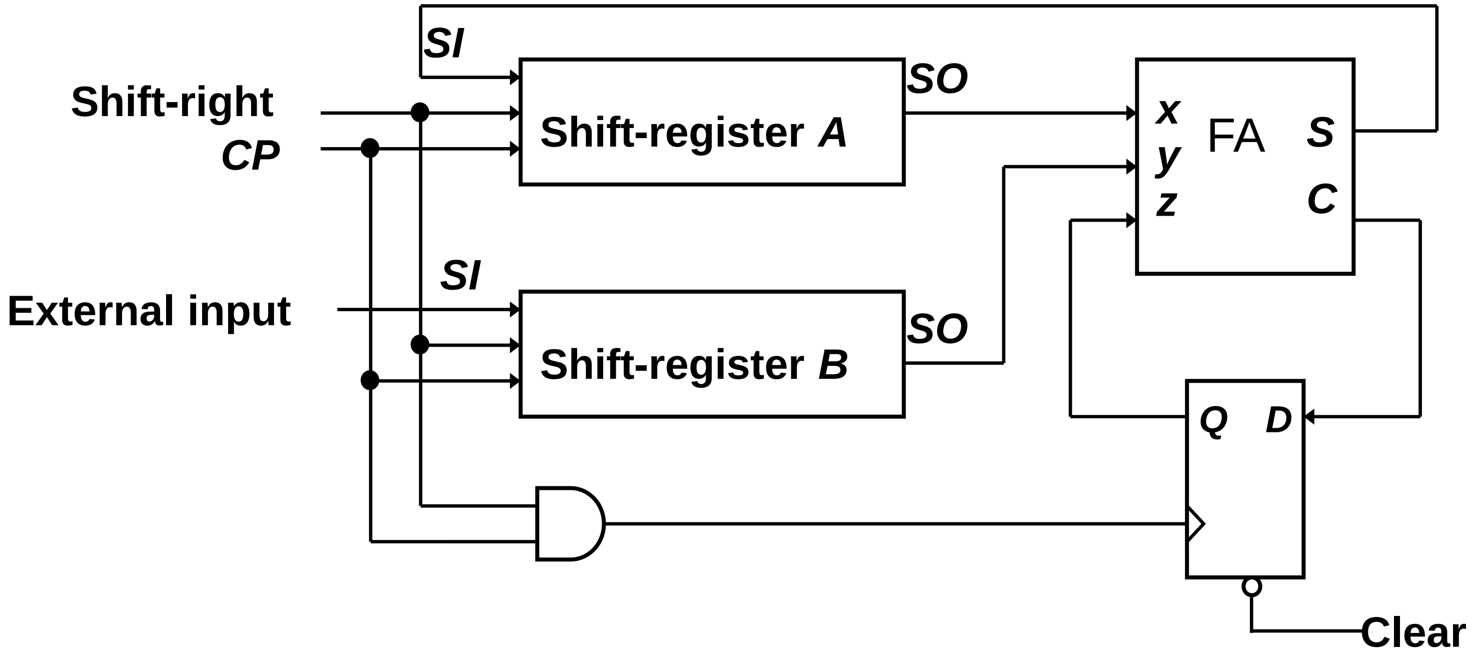
- 4-bit bidirectional shift register with parallel load.

Mode Control		
s_1	s_0	***Register Operation***
0	**0**	**No change**
0	**1**	**Shift right**
1	**0**	**Shift left**
1	**1**	**Parallel load**

An Application – Serial Addition

- Most operations in digital computers are done in parallel. Serial operations are slower but require less equipment.
- A serial adder is shown below.

$$A \leftarrow A + B.$$



Sequence Detectors

Sequence Detector

- Mealy machine, 101 sequence

- For non overlapping case

  Input :0110101011001
  Output:0000100010000

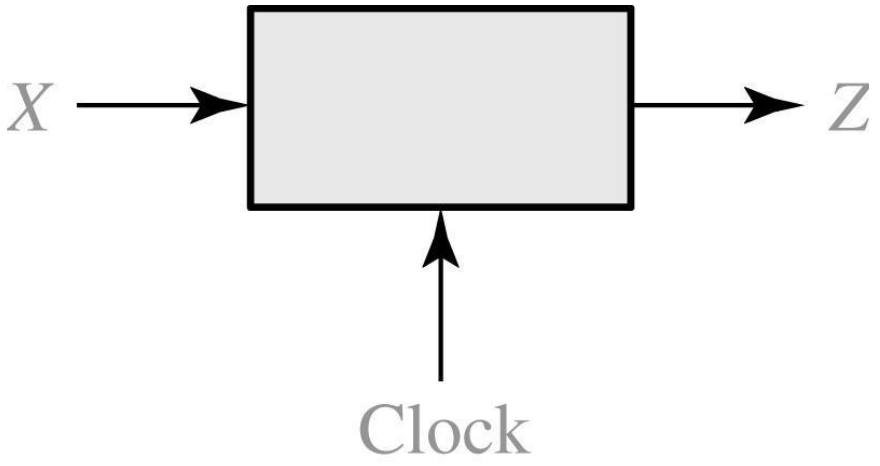
- For overlapping case

  Input: 0110101011001

  Output: 0000101010000

Sequence Detector

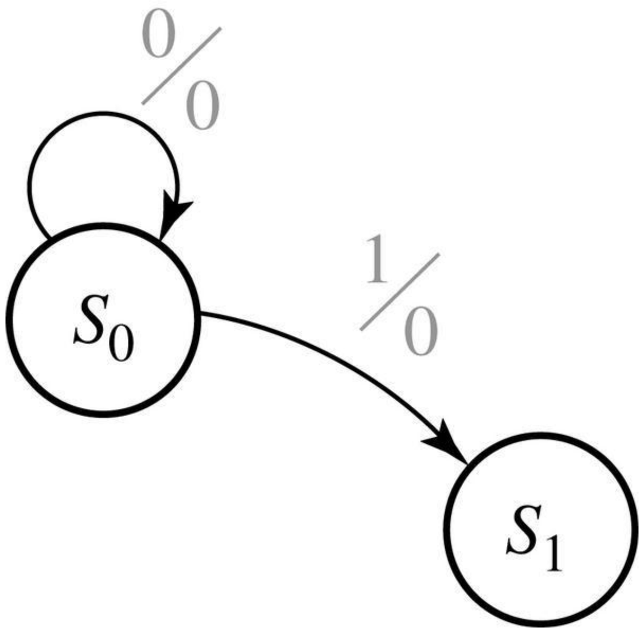
- 101 sequence Detector

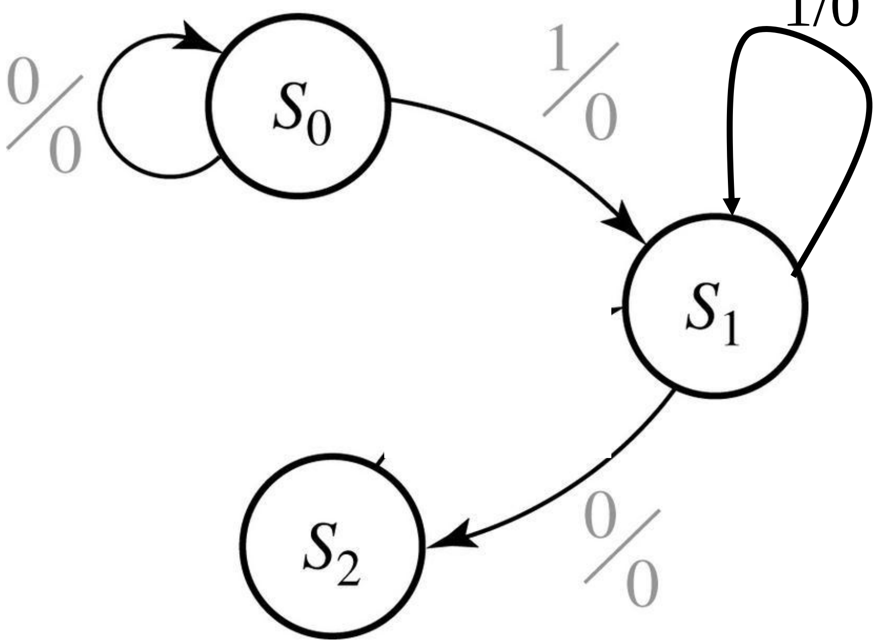


X	=	0	0	1	1	0	1	1	0	0	1	0	1	0	1	0	0
Z	=	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0
(time:		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15)

Design of 101 Sequence Detector (Mealy)

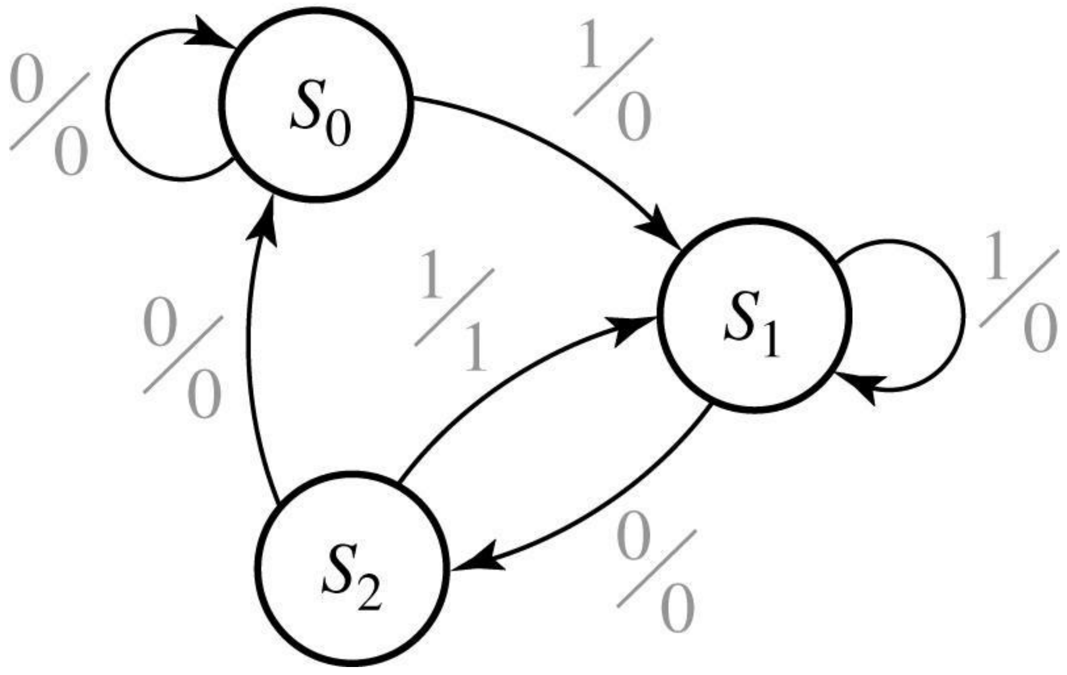
- State Diagram:



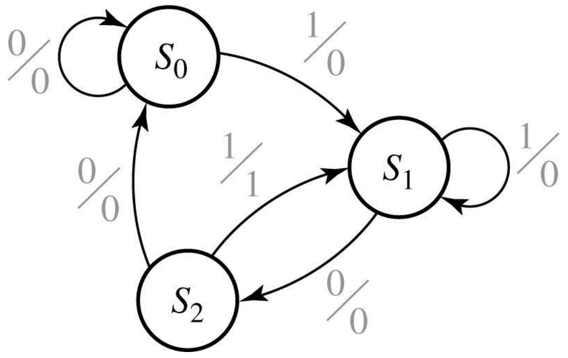


Design of 101 Sequence Detector

- State Diagram (final):



Design of 101 Sequence Detector



- State Table:

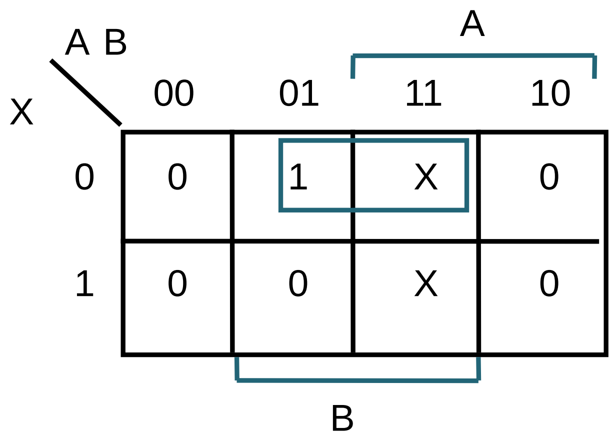
Present state	Next State X = 0	Next State X = 1	Present Output X = 0	Present Output X = 1
S_0	S_0	S_1	0	0
S_1	S_2	S_1	0	0
S_2	S_0	S_1	0	1

- State Table with State Assignment:

AB	A^+B^+ X = 0	A^+B^+ X = 1	Z X = 0	Z X = 1
00	00	01	0	0
01	10	01	0	0
10	00	01	0	1

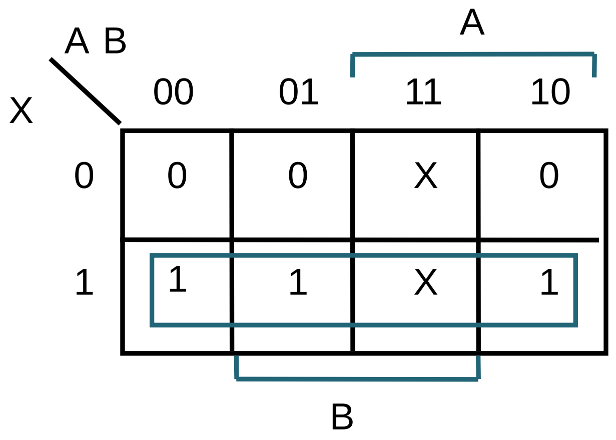
Design of 101 Sequence Detector

- Derive Boolean Equations:



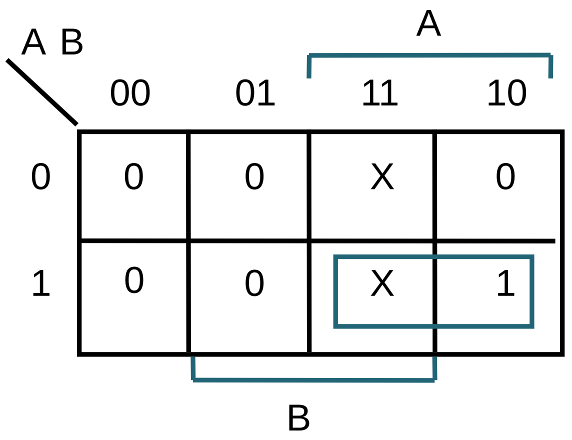
X \ AB	00	01	11	10
0	0	1	X	0
1	0	0	X	0

$$D_A = X'.B$$



X \ AB	00	01	11	10
0	0	0	X	0
1	1	1	X	1

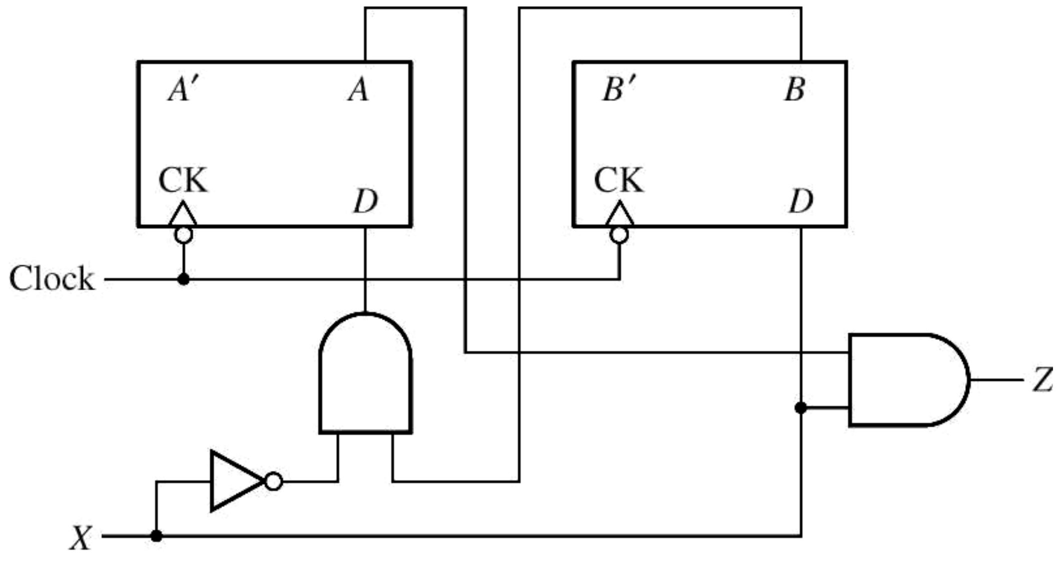
$$D_B = X$$

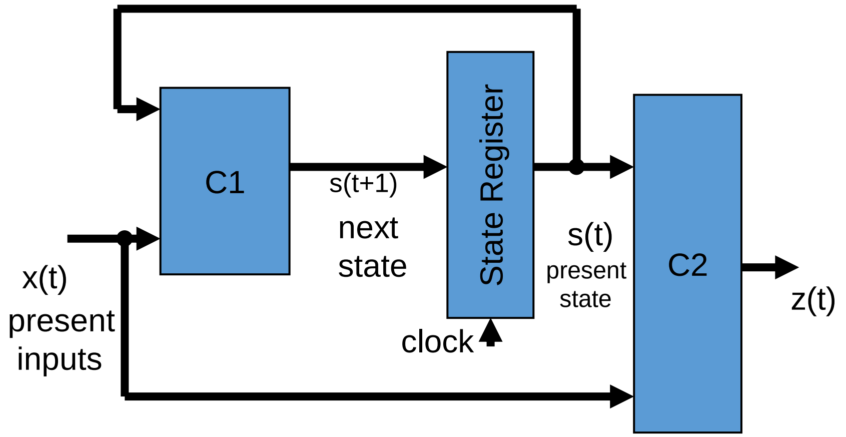


X \ AB	00	01	11	10
0	0	0	X	0
1	0	0	X	1

$$Z = X.A$$

Design of Sequence Detector

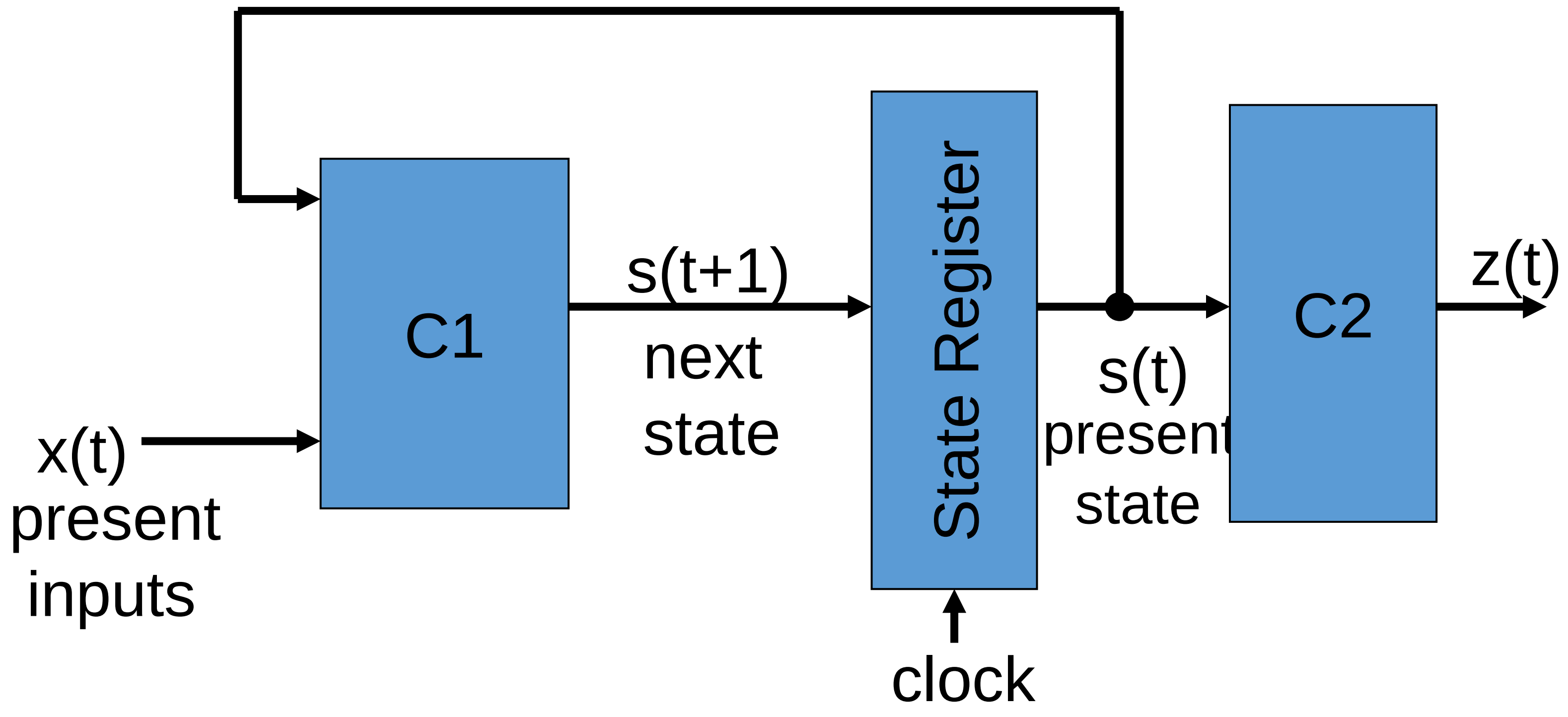




Compare with Typical Mealy Machine

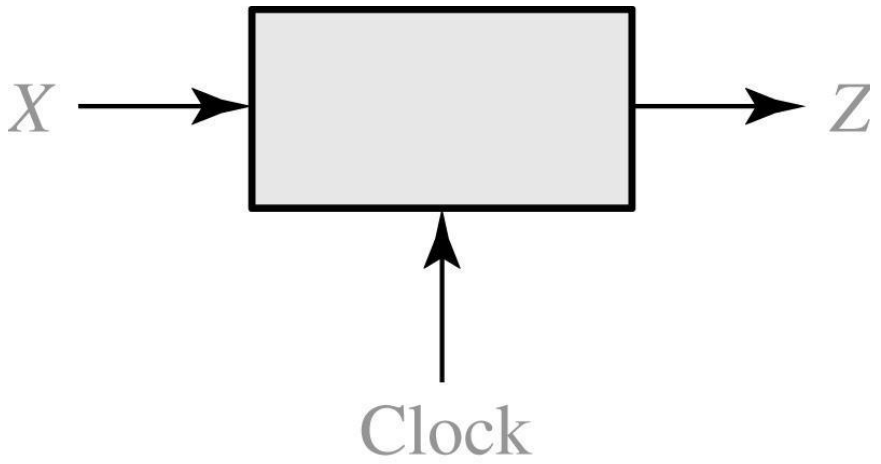
Design of Sequence Detector

- A Moore Sequence Detector:



Sequence Detector

- 101 sequence Detector



X	=	0	0	1	1	0	1	1	0	0	1	0	1	0	1	0	0
Z	=	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0
(time:		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15)

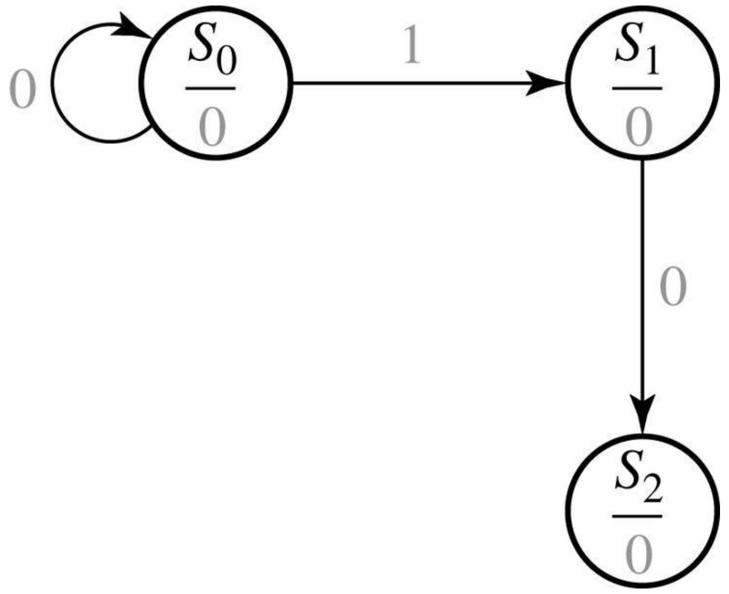
Design of a Sequence Detector

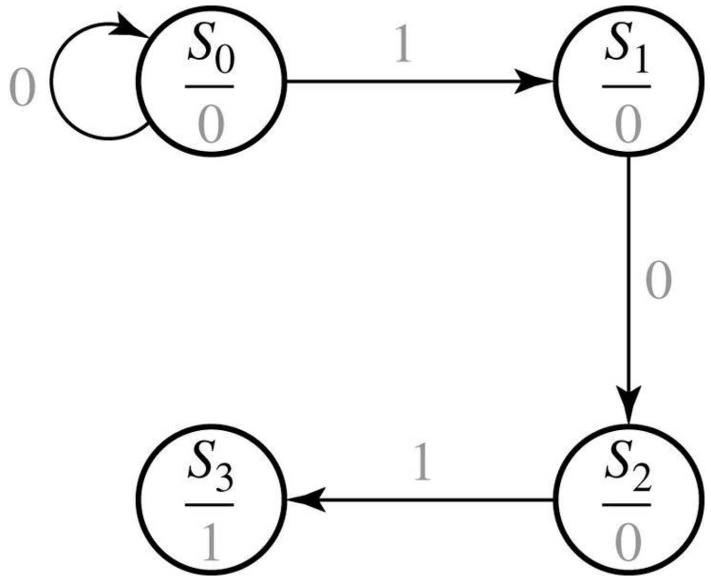
S_0: start

S_1: got 1

S_2: got 10

S_3: got 101





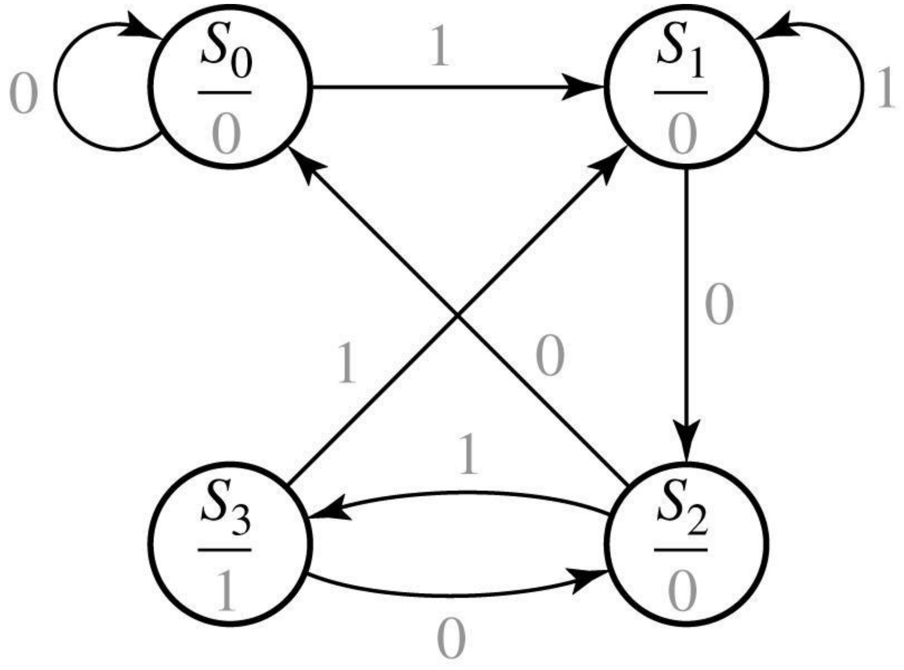
Design of a Sequence Detector

S_0: start

S_1: got 1

S_2: got 10

S_3: got 101



Design of a Sequence Detector

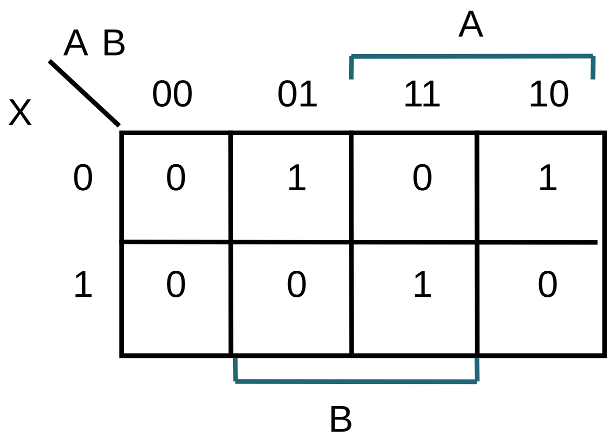
- State Table

Present state	Next State X = 0	Next State X = 1	Present Output (Z)
S_0	S_0	S_1	0
S_1	S_2	S_1	0
S_2	S_0	S_3	0
S_3	S_2	S_1	1

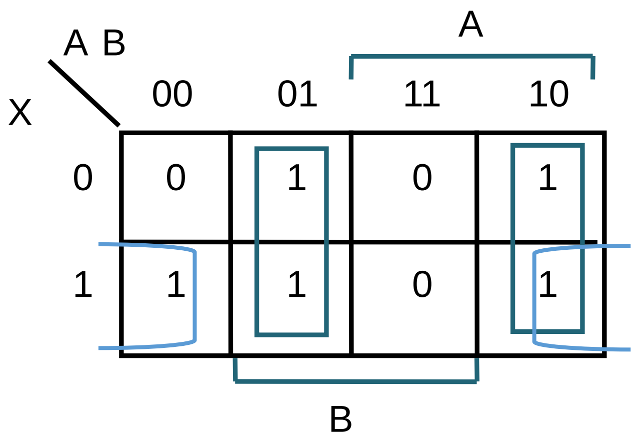
- Transition Table with State assignment

AB	D_A D_B / A^+ B^+ X = 0	D_A D_B / A^+ B^+ X = 1	Z
00	00	01	0
01	11	01	0
11	00	10	0
10	11	01	1

Design of a Sequence Detector



$$D_A = \bar{X}\bar{A}B + XAB + \bar{X}A\bar{B}$$



$$D_B = \bar{A}B + X\bar{B} + A\bar{B}$$

$$Z = A\bar{B}$$

References

- Chapter 9, Digital Fundamentals by Thomas L. Floyd
- Chapter 5 & 6, Digital Design by M. Morris Mano
- Disclaimer: I don’t own all the slides, few of these slides are copied and adopted from various public resources available on the internet.

Thank you