Processor Architecture

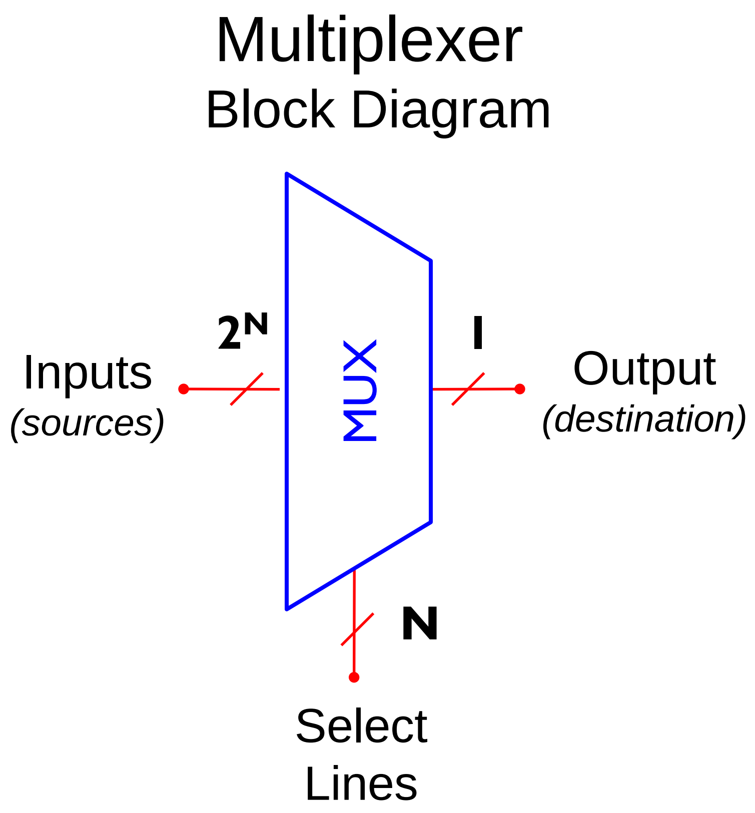
Module 1 :Processor design basics

Combinational Logic

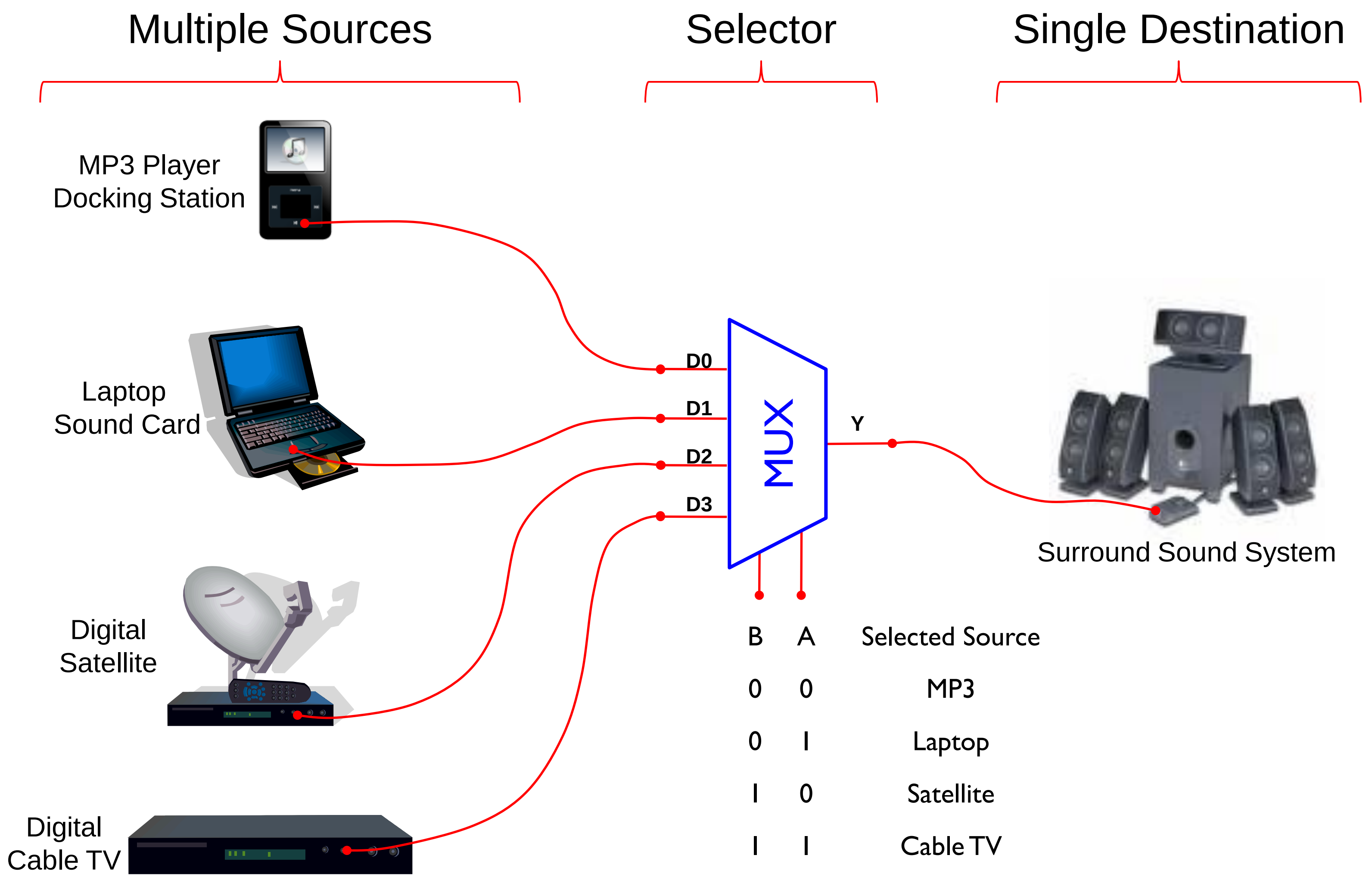
- Combinational Logic:
  - Output depends only on current input
  - Has no memory

What is a Multiplexer (MUX)?

- Data selector
- A MUX is a digital switch that has multiple inputs (sources) and a single output (destination).
- The select lines determine which input is connected to the output.
- MUX Types
  - → 2-to-1 (1 select line)
  - → 4-to-1 (2 select lines)
  - → 8-to-1 (3 select lines)
  - → 16-to-1 (4 select lines)

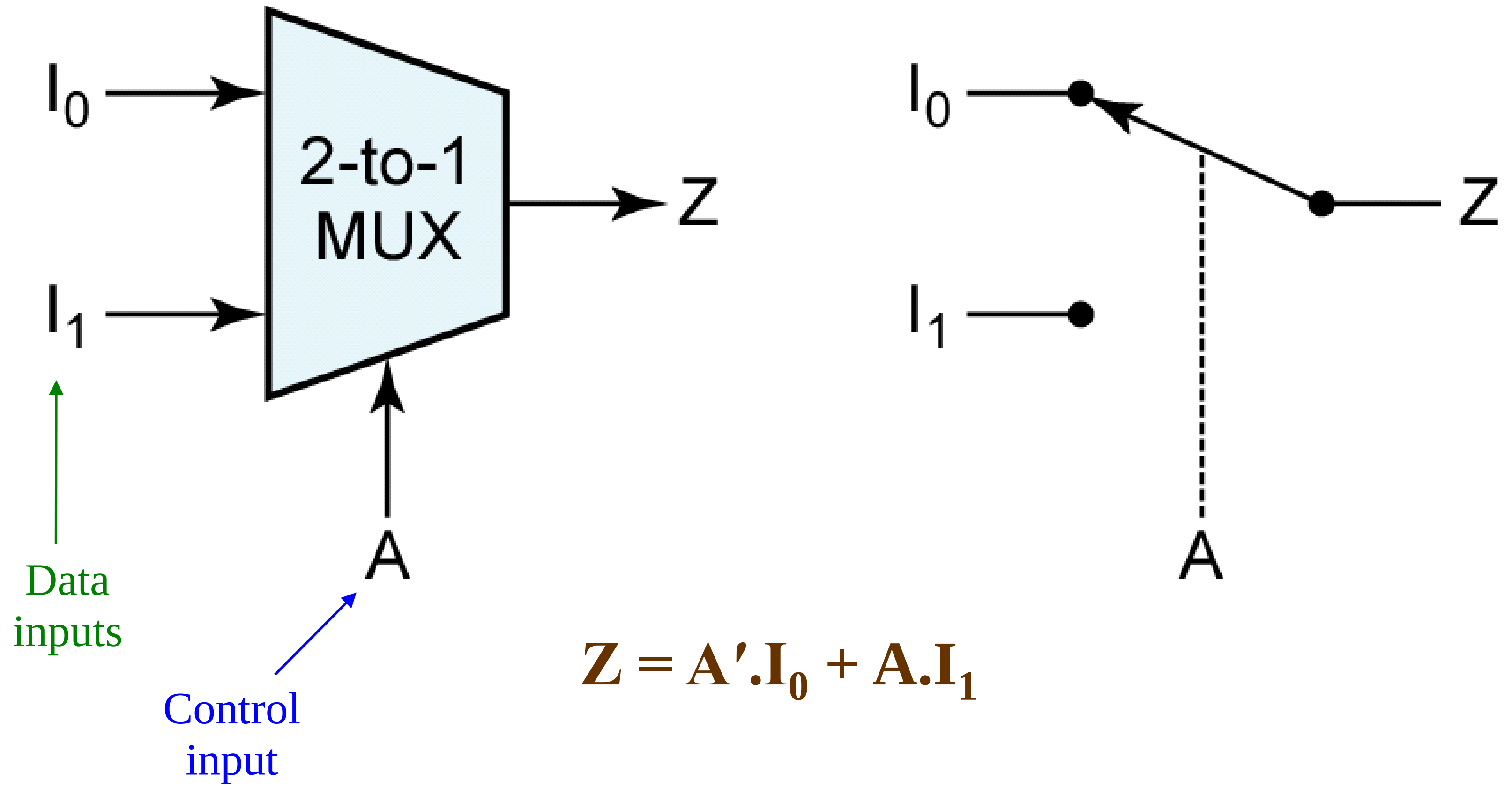


Typical Application of a MUX



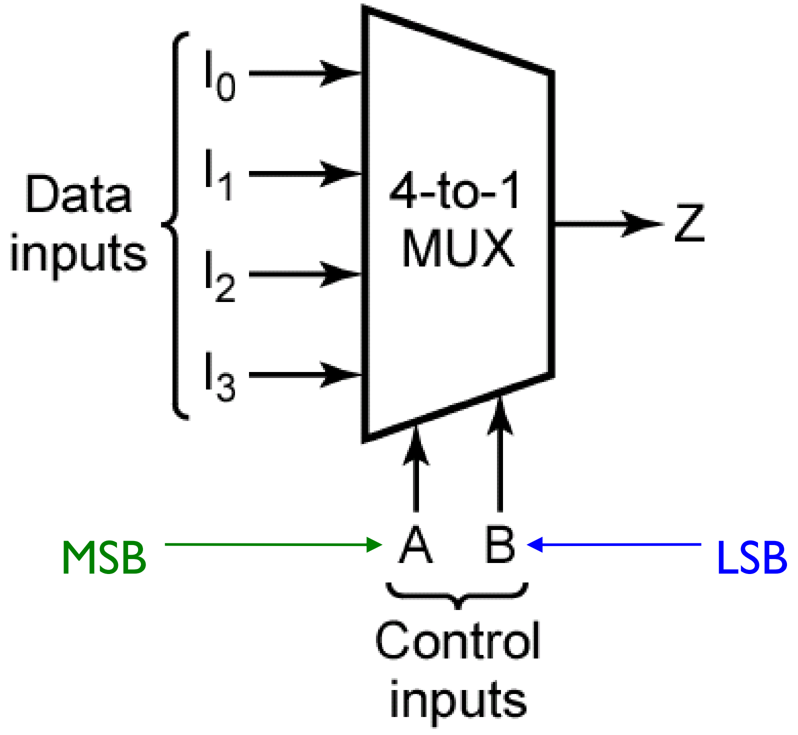
B	A	Selected Source
0	0	MP3
0	1	Laptop
1	0	Satellite
1	1	Cable TV

Multiplexers



$$Z = A'.I_0 + A.I_1$$

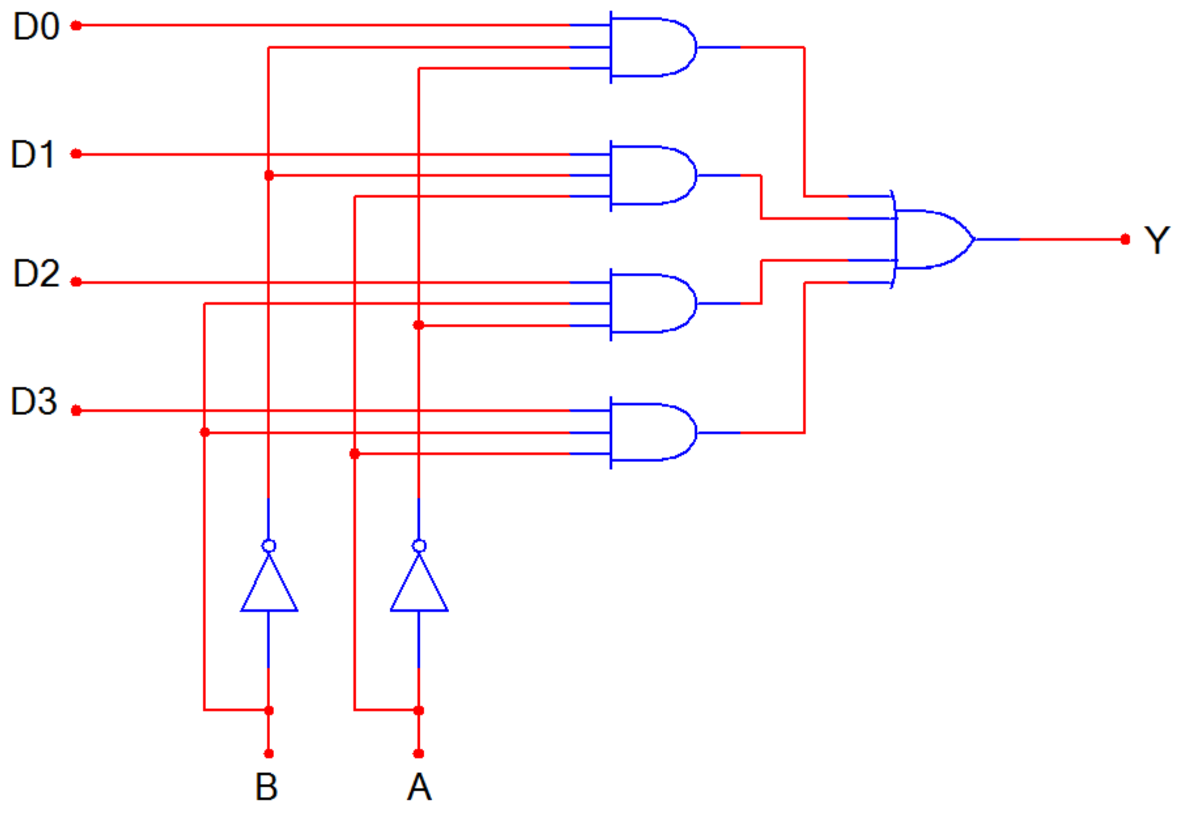
Multiplexers



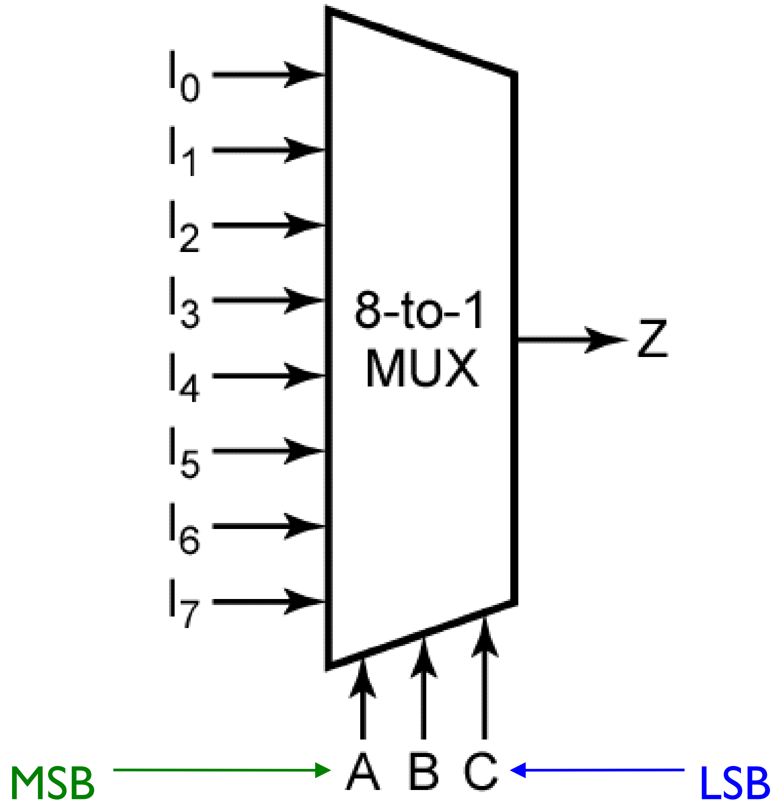
A	B	F
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

$$Z = A'.B'.I_0 + A'.B.I_1 + A.B'.I_2 + A.B.I_3$$

4-to-1 Multiplexer (MUX)



Multiplexers



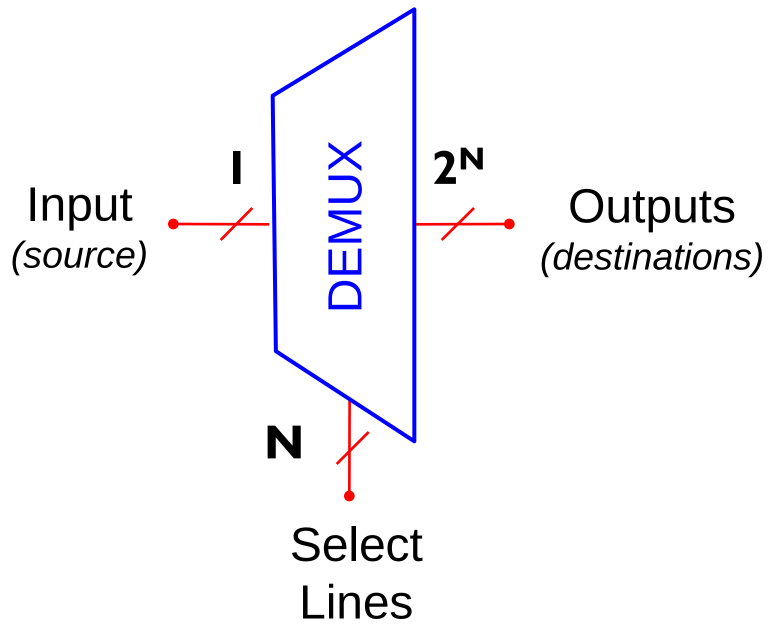
A	B	C	**F**
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

$$\mathbf{Z = A'.B'.C'.I_0 + A'.B'.C.I_1 + A'.B.C'.I_2 + A'.B.C.I_3 + A.B'.C'.I_0 + A.B'.C.I_1 + A'.B.C'.I_2 + A.B.C.I_3}$$

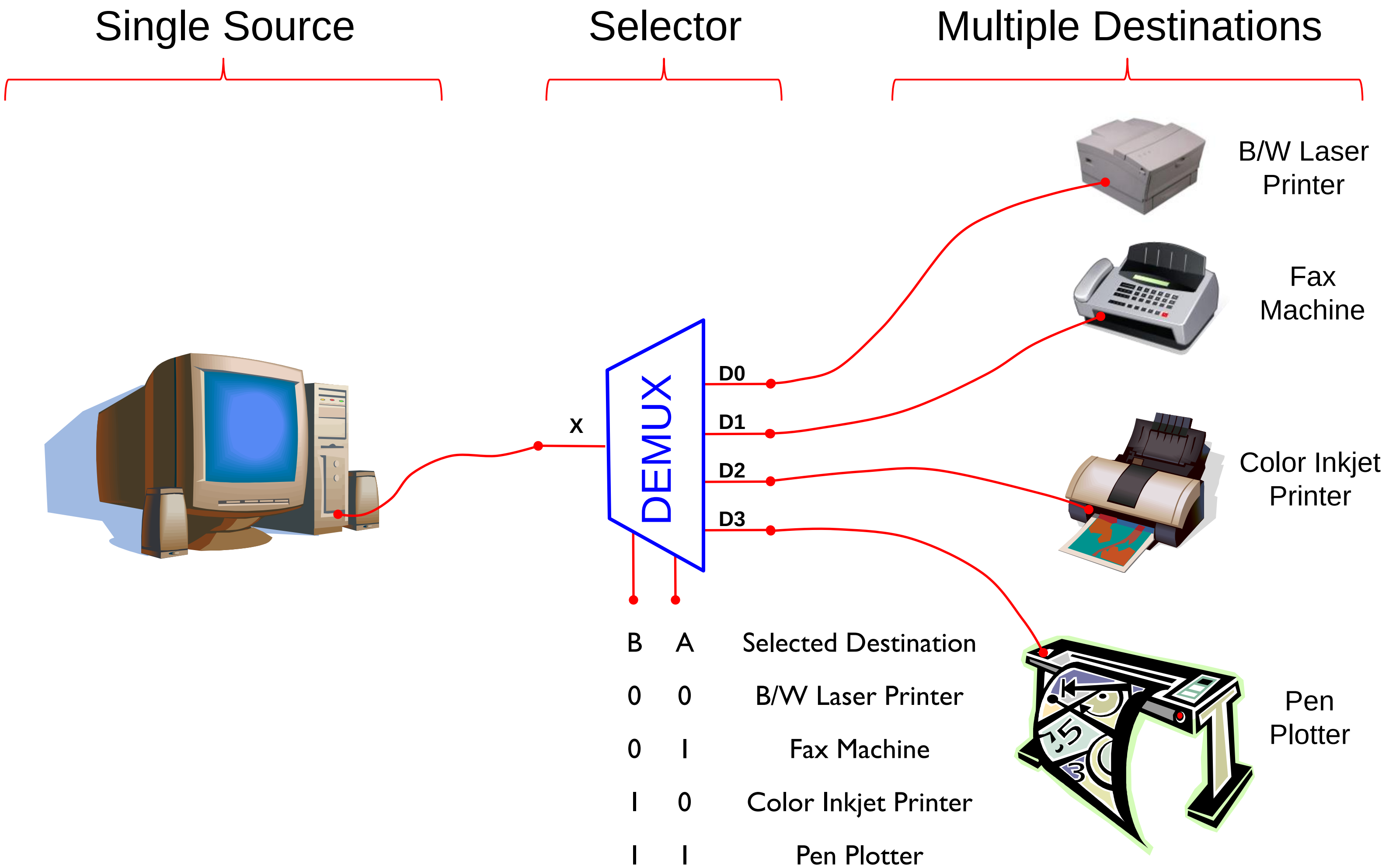
What is a Demultiplexer (DEMUX)?

- Data distributer
- A DEMUX is a digital switch with a single input (source) and a multiple outputs (destinations).
- The select lines determine which output the input is connected to.
- DEMUX Types
  - → 1-to-2 (1 select line)
  - → 1-to-4 (2 select lines)
  - → 1-to-8 (3 select lines)
  - → 1-to-16 (4 select lines)

Demultiplexer
Block Diagram

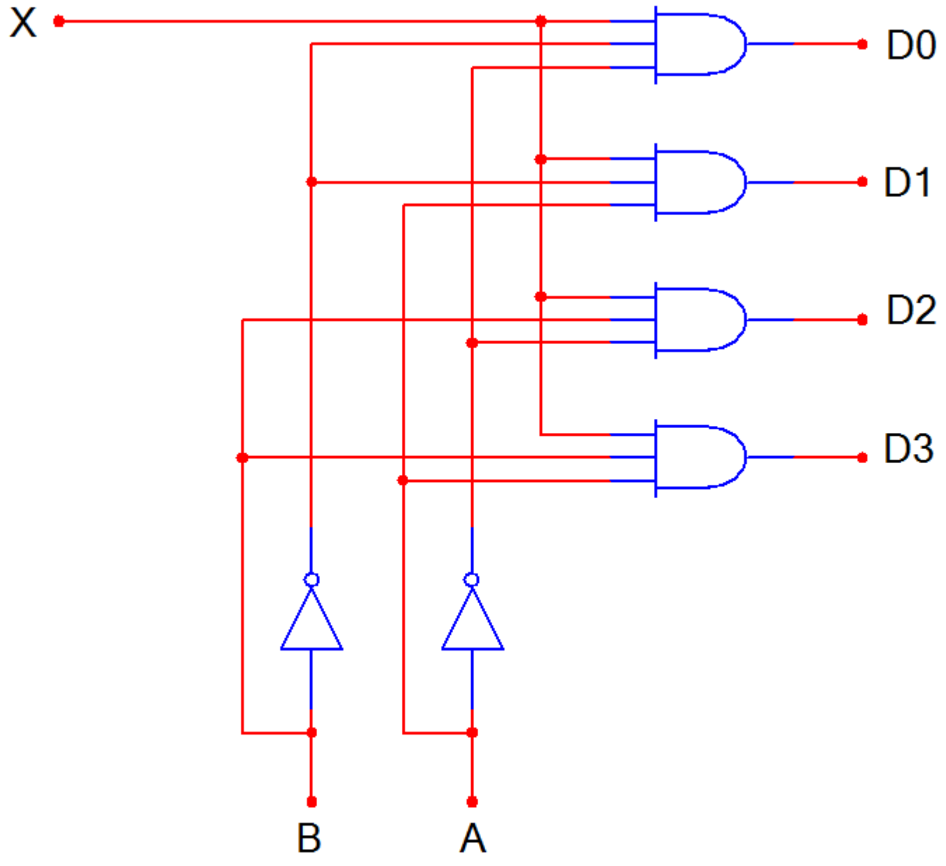


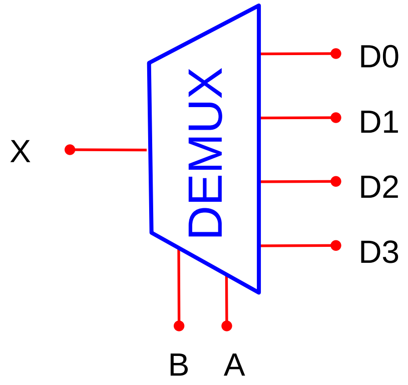
Typical Application of a DEMUX



B	A	Selected Destination
0	0	B/W Laser Printer
0	1	Fax Machine
1	0	Color Inkjet Printer
1	1	Pen Plotter

1-to-4 De-Multiplexer (DEMUX)





B	A	D0	D1	D2	D3
0	0	X	0	0	0
0	1	0	X	0	0
1	0	0	0	X	0
1	1	0	0	0	X

Decoders

- A decoder has
  - N inputs
  - 2^N outputs
- A decoder selects one of 2^N outputs by decoding the binary value on the N inputs.
- The decoder generates all of the minterms of the N input variables.
  - Exactly one output will be active for each combination of the inputs.

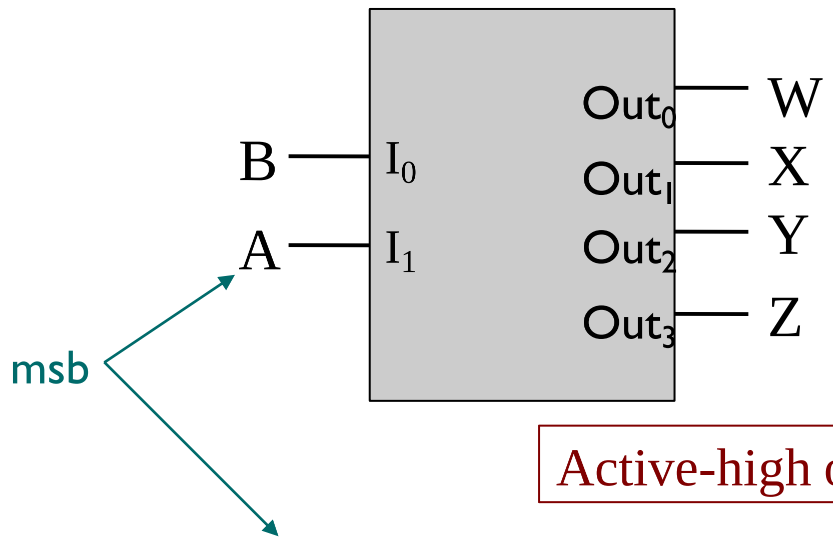
Decoders

- Difference between Multiplexer and Encoder:-

A multiplexer or MUX is a combination circuit that contains more than one input line, one output line and more than one selection line.

Whereas, an encoder is also considered a type of multiplexer but without a single output line. It is a combinational logic function that has 2^n (or fewer) input lines and n output lines.

Decoders



W = A'.B'

X = A.B'

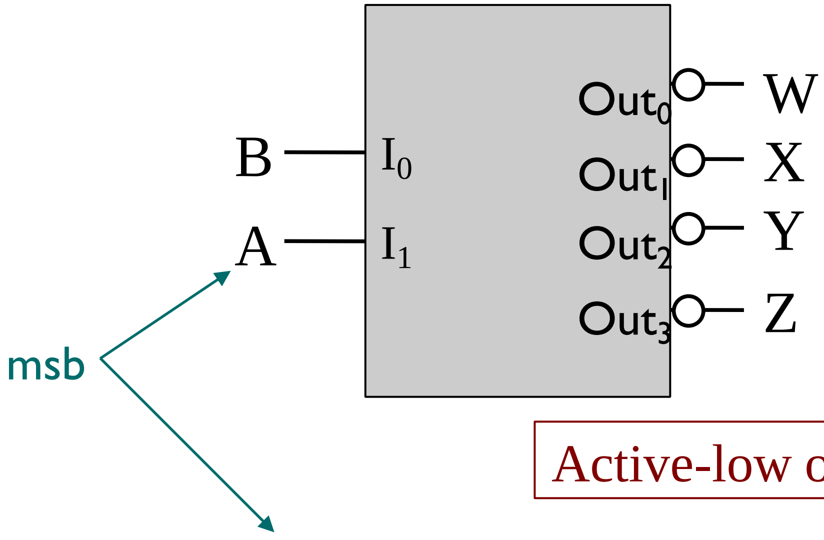
Y = A'.B

Z = A.B

Active-high outputs

A	B	**W**	**X**	**Y**	**Z**
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Decoders



W = (A'.B')'

X = (A.B')'

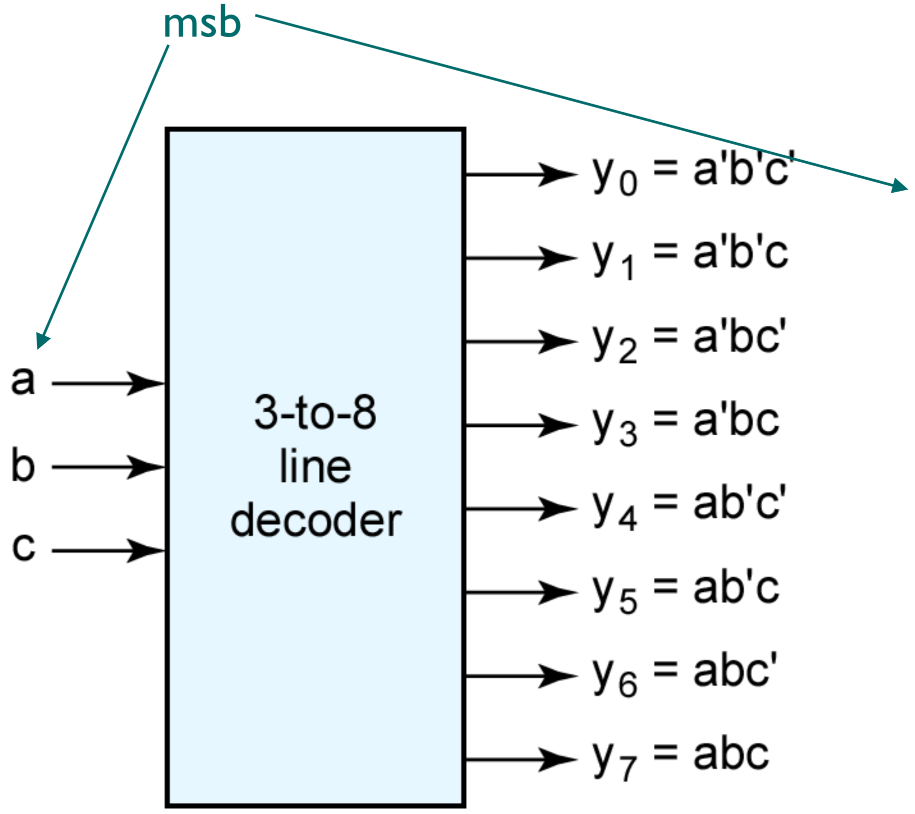
Y = (A'.B)'

Z = (A.B)'

Active-low outputs

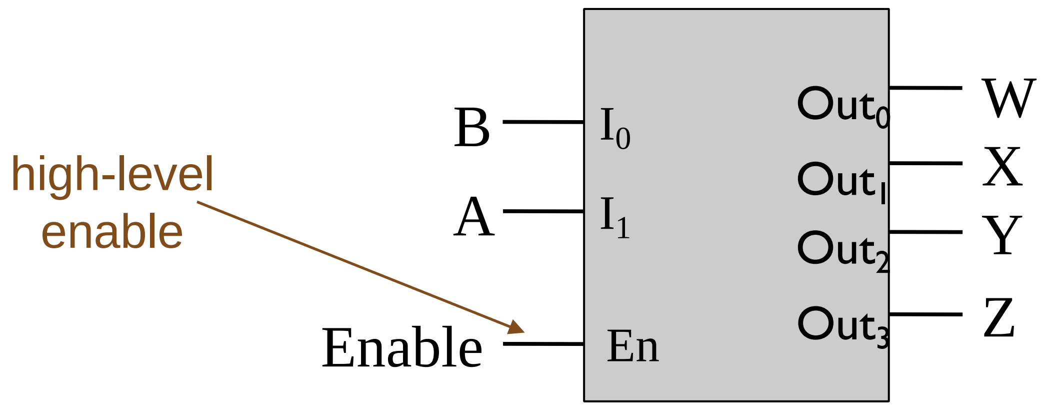
A	B	**W**	**X**	**Y**	**Z**
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

Decoders



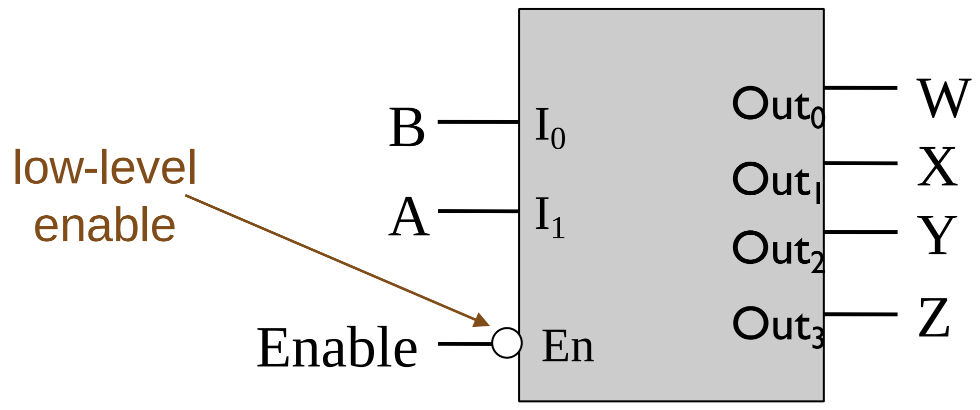
a	b	c	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Decoder with Enable



	En	A	B	**W**	**X**	**Y**	**Z**
enabled	1	0	0	1	0	0	0
	1	0	1	0	1	0	0
	1	1	0	0	0	1	0
	1	1	1	0	0	0	1
disabled	0	x	x	0	0	0	0

Decoder with Enable

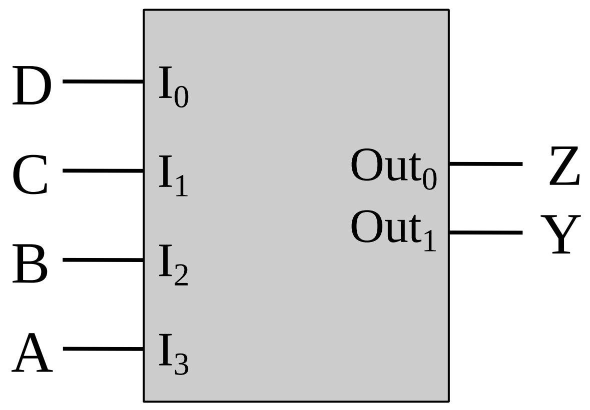


	En	A	B	**W**	**X**	**Y**	**Z**
enabled	0	0	0	1	0	0	0
	0	0	1	0	1	0	0
	0	1	0	0	0	1	0
	0	1	1	0	0	0	1
disabled	1	x	x	0	0	0	0

Encoders

- An encoder has
  - 2^N inputs
  - N outputs
- An encoder outputs the binary value of the selected (or active) input.
- An encoder performs the inverse operation of a decoder.

Encoders

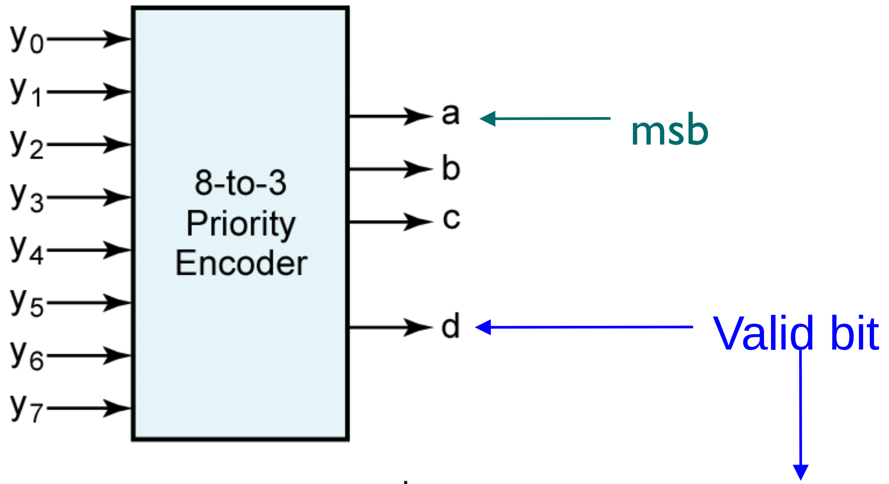


A	B	C	D	**Y**	**Z**
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Priority Encoders

- If more than one input is active, the higher-order input has priority over the lower-order input.
  - The higher value is encoded on the output
- A valid indicator, d, is included to indicate whether or not the output is valid.
  - Output is invalid when no inputs are active
    - d = 0
  - Output is valid when at least one input is active
    - d = 1

Priority Encoders



y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7	a	b	c	d
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
X	1	0	0	0	0	0	0	0	0	1	1
X	X	1	0	0	0	0	0	0	1	0	1
X	X	X	1	0	0	0	0	0	1	1	1
X	X	X	X	1	0	0	0	1	0	0	1
X	X	X	X	X	1	0	0	1	0	1	1
X	X	X	X	X	X	1	0	1	1	0	1
X	X	X	X	X	X	X	1	1	1	1	1

Quiz

Q. 1. Explain half adder with necessary truth table & circuitry.

Q. 2. Explain full subtractor with necessary truth table & circuitry.

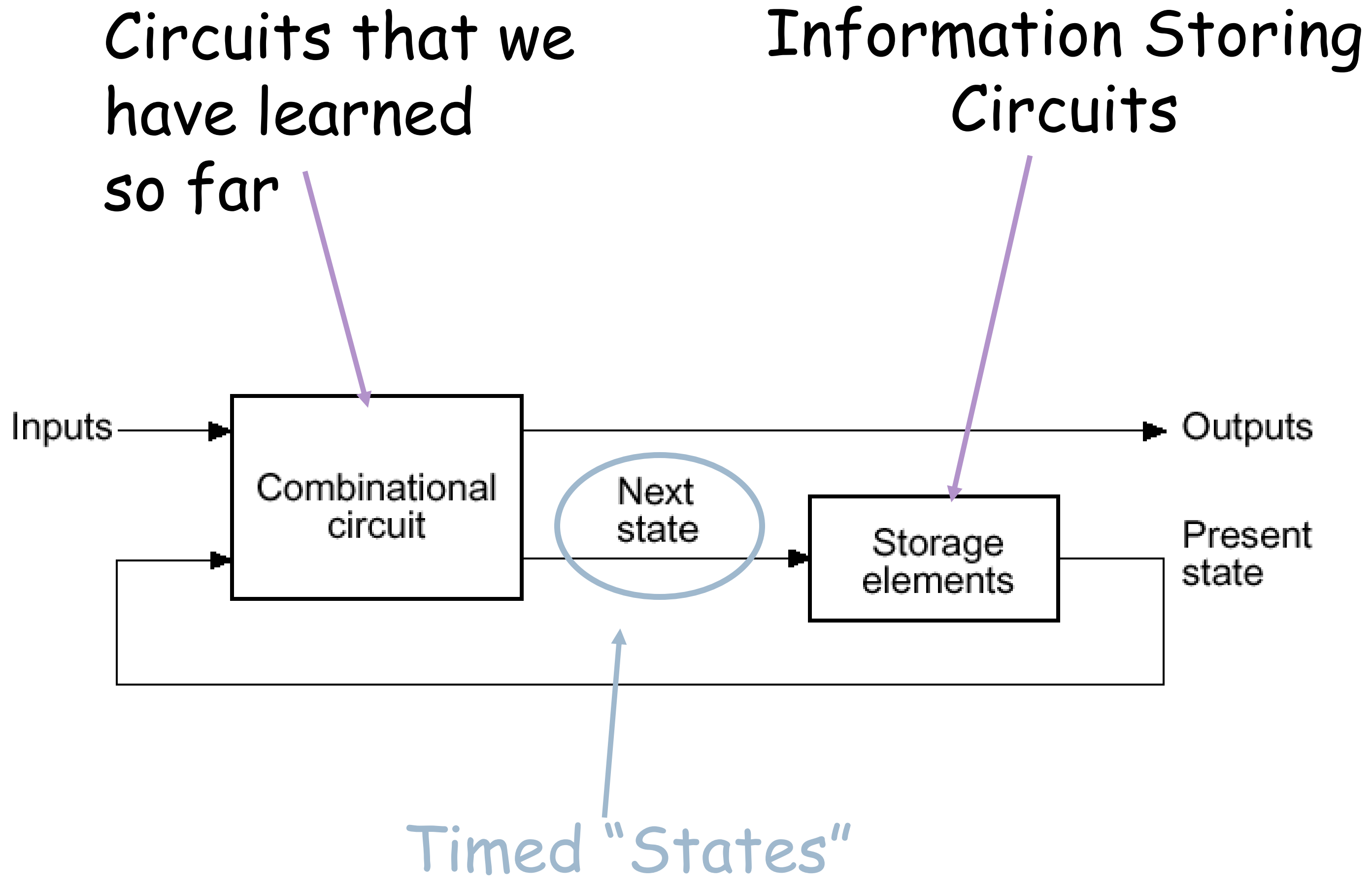
Q. 3. Express half subtractor using NAND gates.

Q. 4. What is single bit magnitude comparator? Explain in details.

Sequential Logic

- Sequential Logic:
  - Output depends not only on current input but also on past input values, e.g., design a counter
  - Need some type of memory to remember the past input values

Sequential Circuits



Sequential Logic: Concept

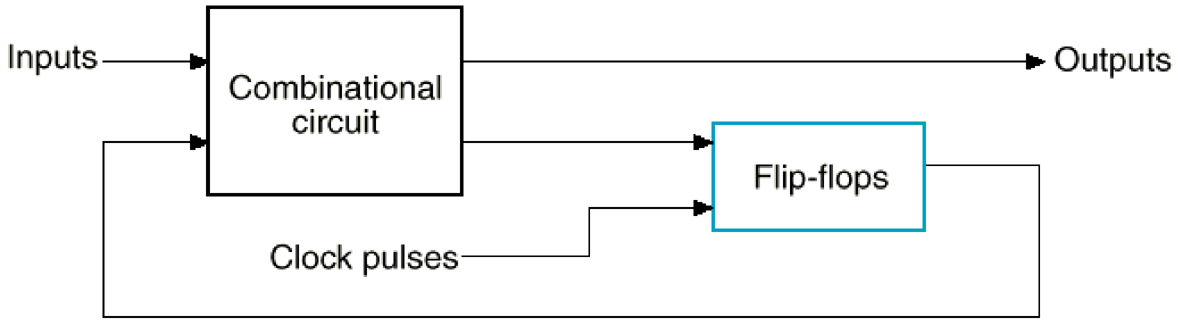
- Sequential Logic circuits remember past inputs and past circuit state.
- Outputs from the system are "fed back" as new inputs
  - With gate delay and wire delay
- The storage elements are circuits that are capable of storing binary information: memory.

Synchronous *vs.* Asynchronous

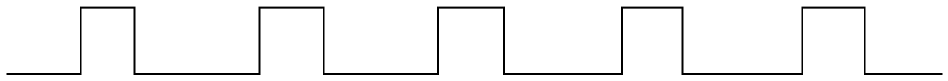
There are two types of sequential circuits:

- **Synchronous** sequential circuit: circuit output changes only at some discrete instants of time. This type of circuits achieves synchronization by using a timing signal called the *clock*.
- **Asynchronous** sequential circuit: circuit output can change at **any** time (clockless).

Synchronous Sequential Circuits: Flip flops as state memory



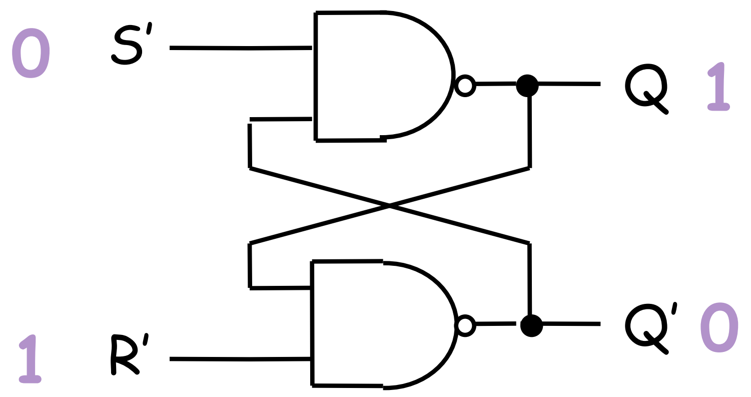
(a) Block diagram



(b) Timing diagram of clock pulses

- **The flip-flops receive their inputs from the combinational circuit and also from a clock signal with pulses that occur at fixed intervals of time, as shown in the timing diagram.**

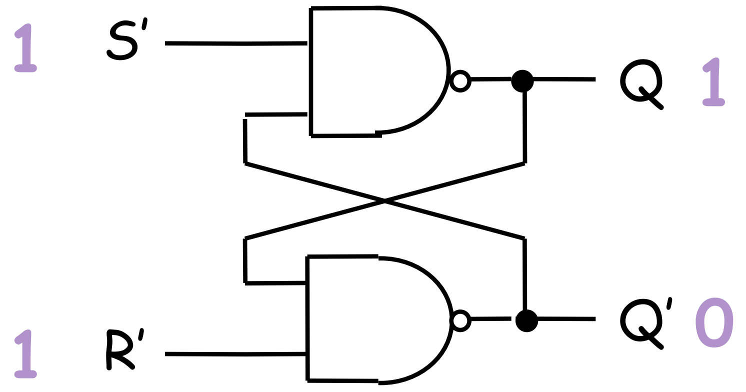
SR Latch (NAND version)



S′	R′	Q	Q′	
0	0			
0	1	1	0	Set
1	0			
1	1			

X	Y	NAND
0	0	1
0	1	1
1	0	1
1	1	0

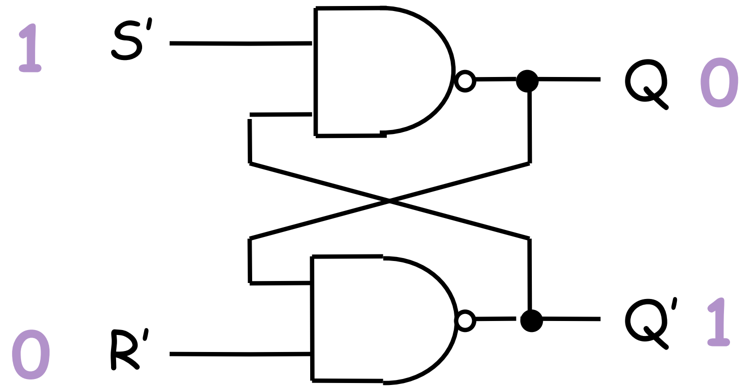
SR Latch (NAND version)



S′	R′	Q	Q′	
0	0			
0	1	1	0	Set
1	0			
1	1	1	0	Hold

X	Y	NAND
0	0	1
0	1	1
1	0	1
1	1	0

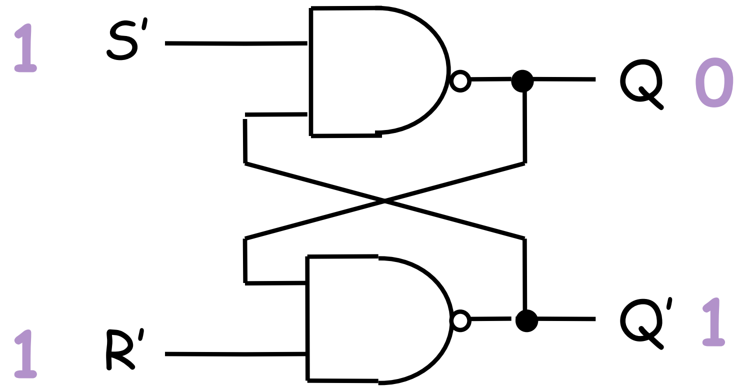
SR Latch (NAND version)



S'	R'	Q	Q'	
0	0			
0	1	1	0	Set
1	0	0	1	Reset
1	1	1	0	Hold

X	Y	NAND
0	0	1
0	1	1
1	0	1
1	1	0

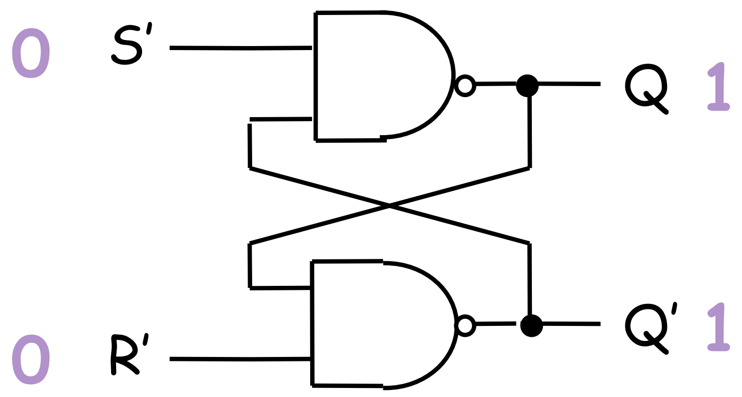
SR Latch (NAND version)



S'	R'	Q	Q'	
0	0			
0	1	1	0	Set
1	0	0	1	Reset
1	1	1	0	Hold
		0	1	Hold

X	Y	NAND
0	0	1
0	1	1
1	0	1
1	1	0

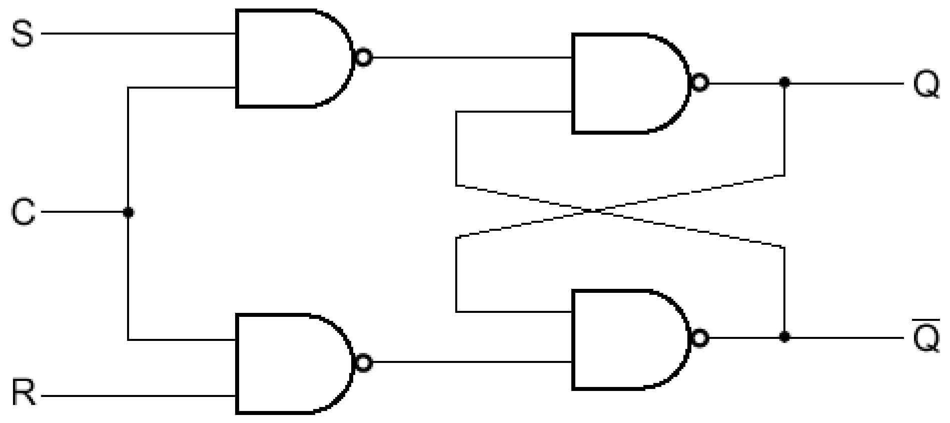
SR Latch (NAND version)



S′	R′	Q	Q′	
0	0	1	1	Disallowed
0	1	1	0	Set
1	0	0	1	Reset
1	1	1	0	Hold
		0	1	Hold

X	Y	NAND
0	0	1
0	1	1
1	0	1
1	1	0

SR Latch with Clock signal



(a) Logic diagram

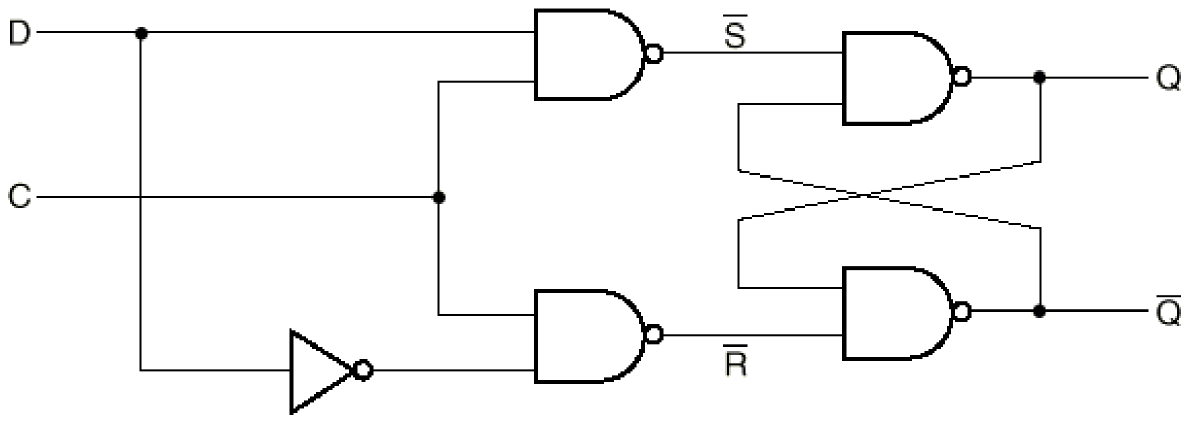
C	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	Q = 0; Reset state
1	1	0	Q = 1; Set state
1	1	1	Undefined

(b) Function table

Latch is sensitive to input changes ONLY when C=1

D Latch

- One way to eliminate the undesirable indeterminate state in the RS flip flop is to ensure that inputs S and R are never 1 simultaneously. This is done in the *D latch:*



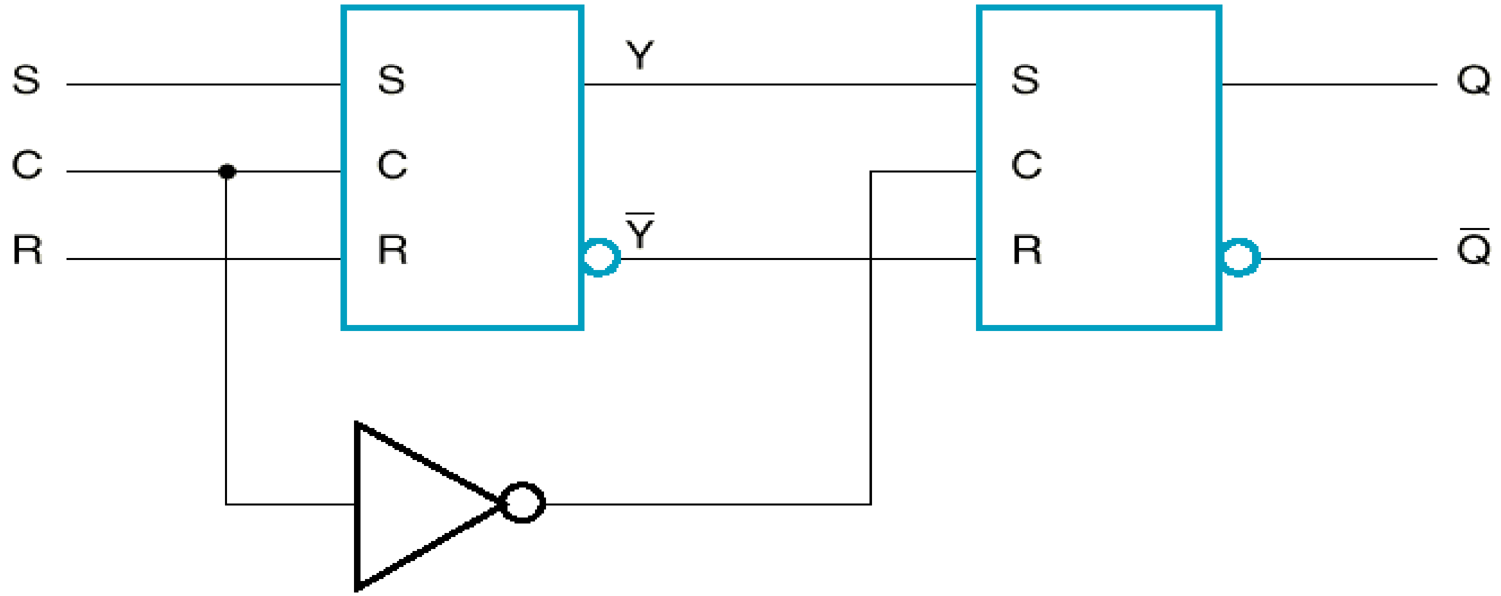
C	D	Next state of Q
0	X	No change
1	0	Q = 0; Reset state
1	1	Q = 1; Set state

Flip-Flops

- Latches are "transparent" (= any change on the inputs is seen at the outputs immediately when C=1).
- This causes synchronization problems.
- Solution: use latches to create flip-flops that can respond (update) only on specific times (instead of any time).
- Types: RS flip-flop and D flip-flop

Master-Slave FF configuration using SR latches

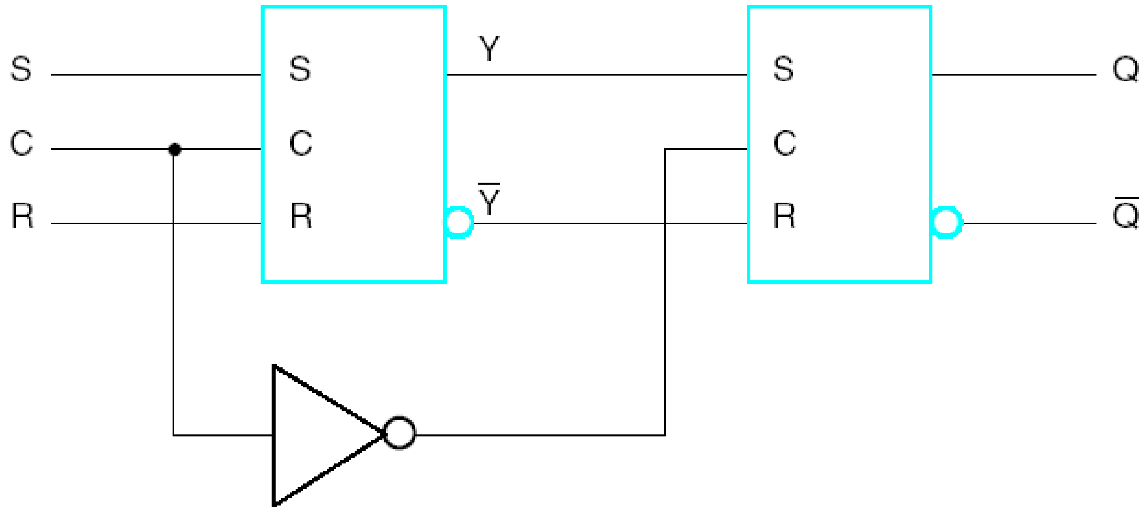
The **Master-Slave Flip-Flop** is basically two gated SR flip-flops connected together in a series configuration with the slave having an inverted clock pulse.
The outputs from Q and Q' from the "Slave" flip-flop are fed back to the inputs of the "Master" with the outputs of the "Master" flip flop being connected to the two inputs of the "Slave" flip flop. This feedback configuration from the slave's output to the master's input gives the characteristic toggle of the JK flip flop as shown below.



Master-Slave FF configuration using SR latches (cont.)

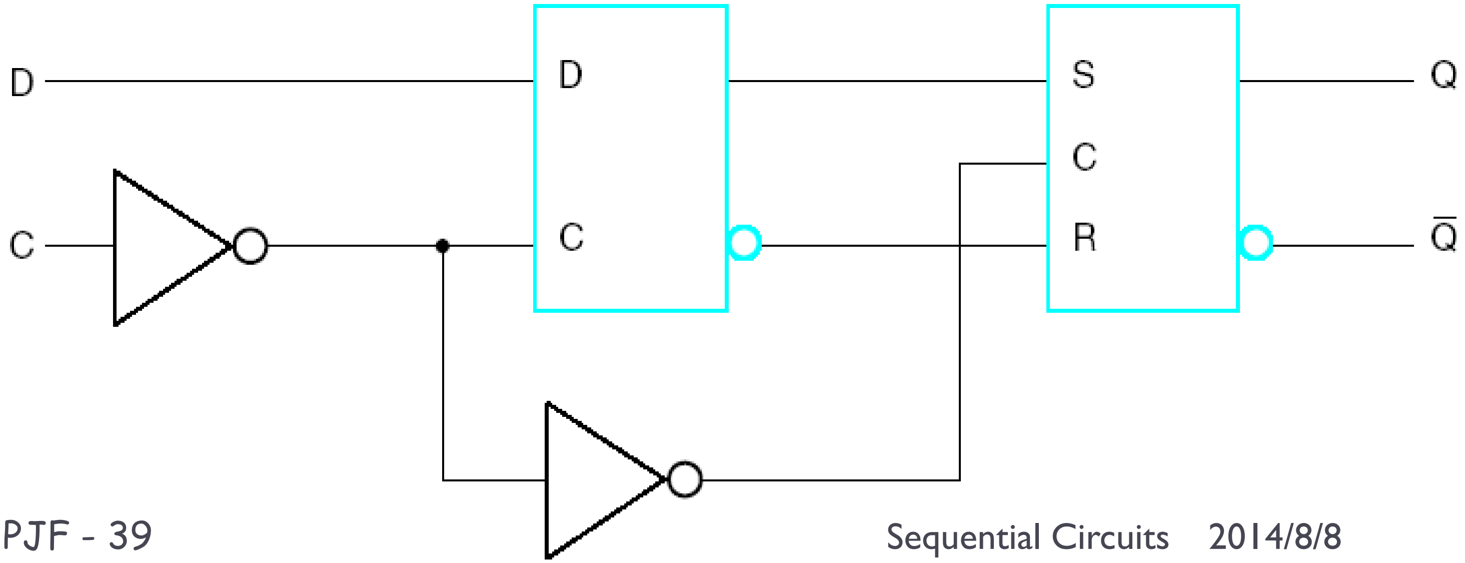
S	R	CLK	Q	Q'	
0	0	1	Q_0	Q_0'	Store
0	1	1	0	1	Reset
1	0	1	1	0	Set
1	1	1	1	1	Disallowed
X	X	0	Q_0	Q_0'	Store

- When C=1, master is enabled and stores *new* data, slave stores *old* data.
- When C=0, master's state passes to enabled slave, master not sensitive to new data (disabled).



D Flip-Flop

The working of D flip flop is similar to the D latch except that the output of D Flip Flop takes the state of the D input at the moment of a positive edge at the clock pin (or negative edge if the clock input is active low) and delays it by one clock cycle. That's why, it is commonly known as a delay flip flop. The D FlipFlop can be interpreted as a delay line or zero order hold. The advantage of the D flip-flop over the D-type "transparent latch" is that the signal on the D input pin is captured the moment the flip-flop is clocked, and subsequent changes on the D input will be ignored until the next clock event.



D Flip-Flop

Characteristics and applications of D latch and D Flip Flop :

1. D-latch is a level Triggering device while D Flip Flop is an Edge triggering device.
2. The disadvantage of the D FF is its circuit size, which is about twice as large as that of a D latch. That's why, delay and power consumption in Flip flop is more as compared to D latch.
3. Latches are used as temporary buffers whereas flip flops are used as registers.
4. Flip flop can be considered as a basic memory cell because it stores the value on the data line with the advantage of the output being synchronized to a clock.
5. Many logic synthesis tool use only D flip flop or D latch.
6. D flip flops are also used in finite state machines.

D Flip-Flop

Edge Triggering vs. Level Clocking

1. When a circuit is edge triggered the output can change only on the rising or falling edge of the clock. But in the case of level-clocked, the output can change when the clock is high (or low).

2. In edge triggering output can change only at one instant during the lock cycle; with level clocking output can change during an entire half cycle of the clock.

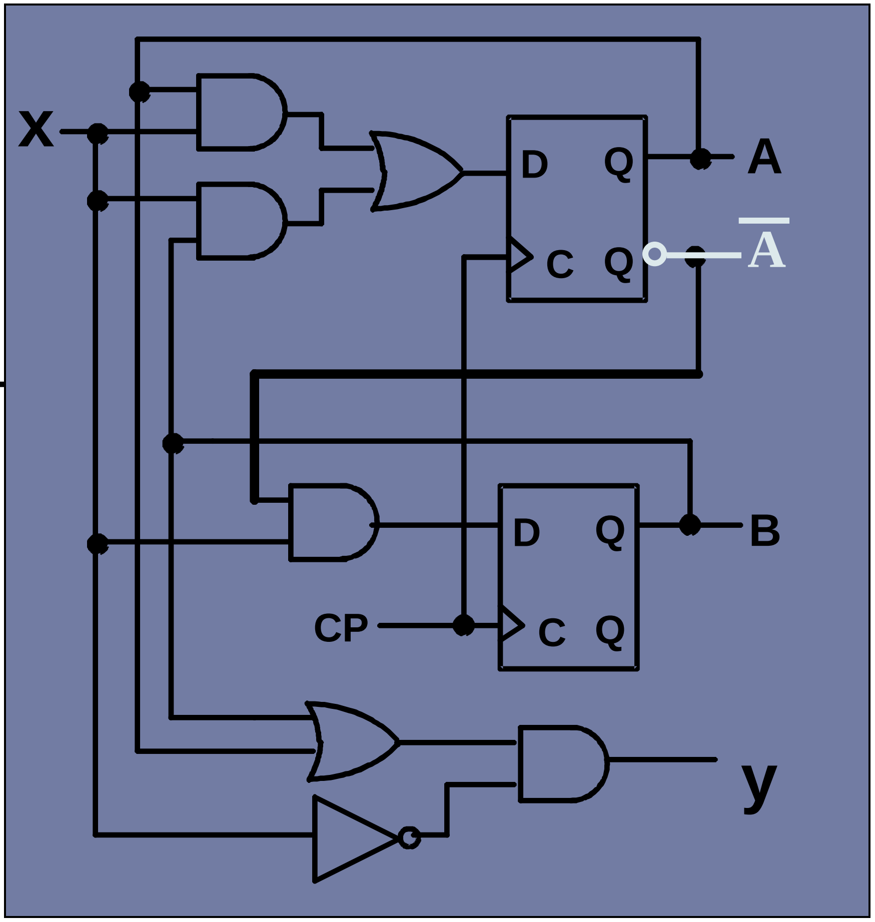
Sequential Circuit Analysis

- ***Analysis***: Consists of obtaining a suitable description that demonstrates the time sequence of inputs, outputs, and states.
- Logic diagram: Boolean gates, flip-flops (of any kind), and appropriate interconnections.
- The logic diagram is derived from any of the following:
  - Boolean Equations (FF-Inputs, Outputs)
  - State Table
  - State Diagram

Example

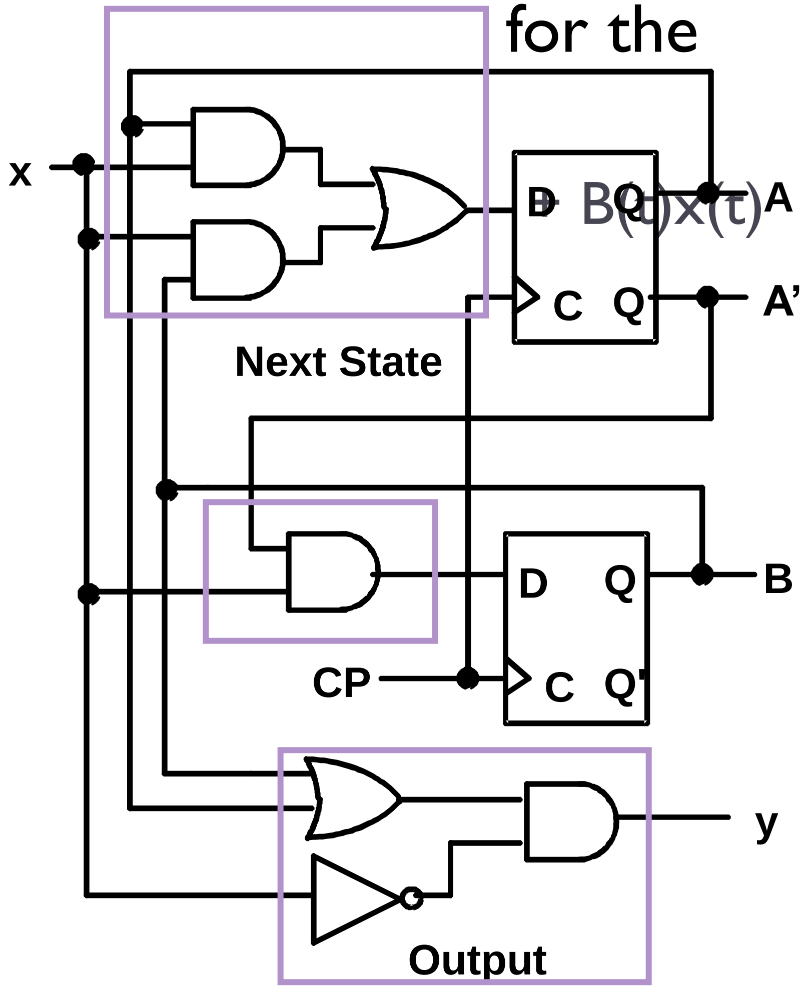
- Input: x(t)
- Output: y(t)
- State: (A(t), B(t))
- What is the Output

- What is the Next State Function?



Example (continued)

- Boolean equations for the functions:
  - A(t+1) = A(t)x(t) + B(t)x(t)
  - B(t+1) = A’(t)x(t)
  - y(t) = x’(t)(B(t) + A(t))



State Table Characteristics

- *State table* – a multiple variable table with the following four sections:
  - *Present State* – the values of the state variables for each allowed state.
  - *Input* – the input combinations allowed.
  - *Next-state* – the value of the state at time (t+1) based on the present state and the input.
  - *Output* – the value of the output as a function of the present state and (sometimes) the input.
- From the viewpoint of a truth table:
  - the inputs are Input, Present State
  - and the outputs are Output, Next State

Example: State Table

- The state table can be filled in using the next state and output equations:
  - $A(t+1) = A(t)x(t) + B(t)x(t)$
  - $B(t+1) = \bar{A}(t)x(t);$
  - $y(t) = \bar{x}(t)(B(t) + A(t))$

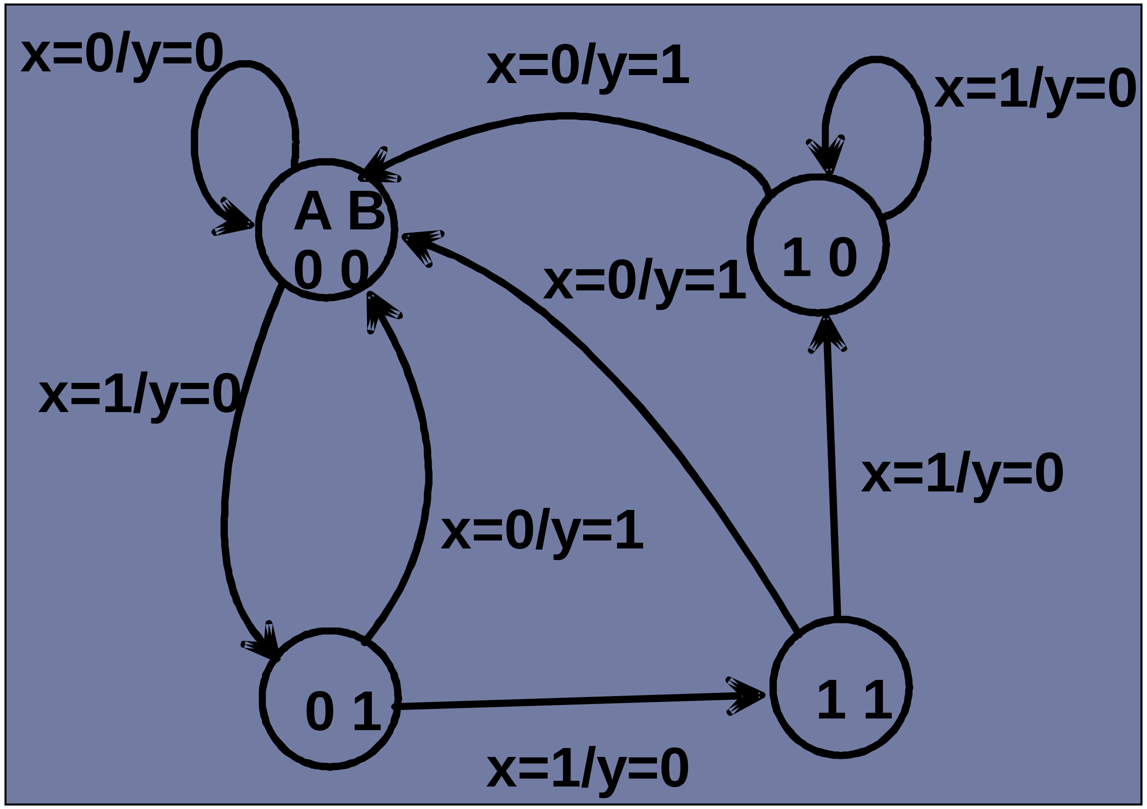
Present State		Input	Next State		Output
A(t)	B(t)	x(t)	A(t+1)	B(t+1)	y(t)
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

State Diagrams

- The sequential circuit function can be represented in graphical form as a state diagram with the following components:
  - A circle with the state name in it for each state
  - A directed arc from the Present State to the Next State for each state transition
  - A label on each directed arc with the Input values which causes the state transition, and
  - A label:
    - On each circle with the output value produced, or
    - On each directed arc with the output value produced.

Example: State Diagram

- Diagram gets confusing for large circuits
- For small circuits, usually easier to understand than the state table



Summary

- Sequential circuit timing analysis
- Flip-Flop
  - Transmission gate based flip-flop design
  - Setup time

Sequential Circuits: Models

Moore Machine

– Outputs are a function of the present state.

– Outputs are independent of the inputs.

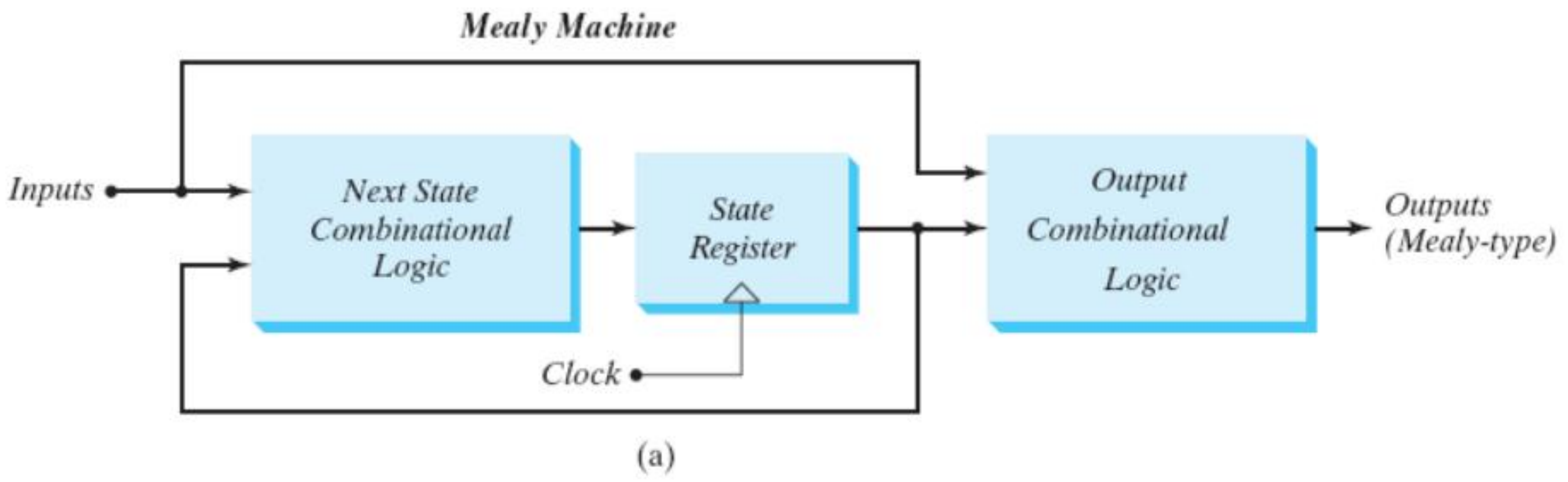
– State diagram includes an output value for each state.

Mealy Machine

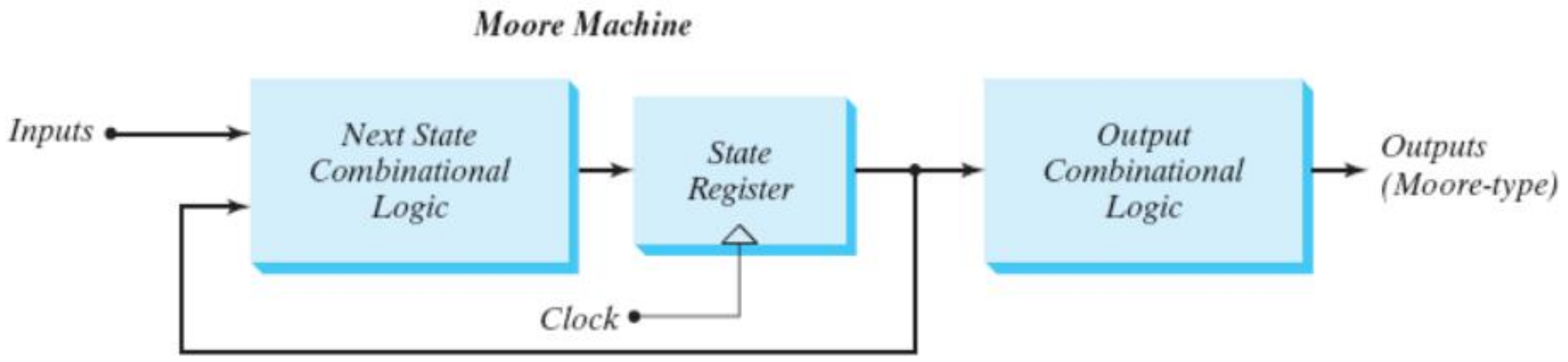
– Outputs are a function of the present state and the present input.

– State diagram includes an input and output value for each transition (between states).

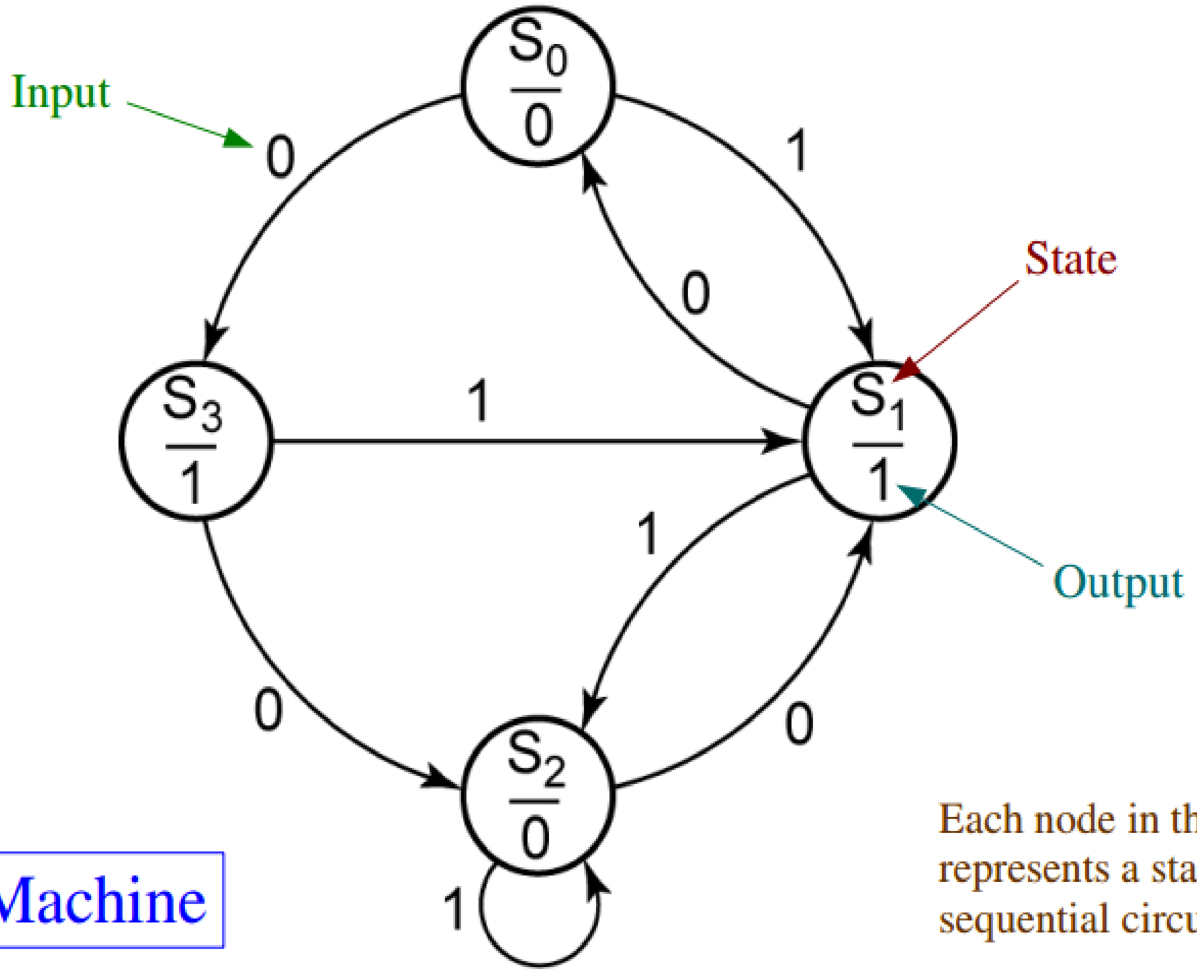
Sequential Circuits: Models



(a)



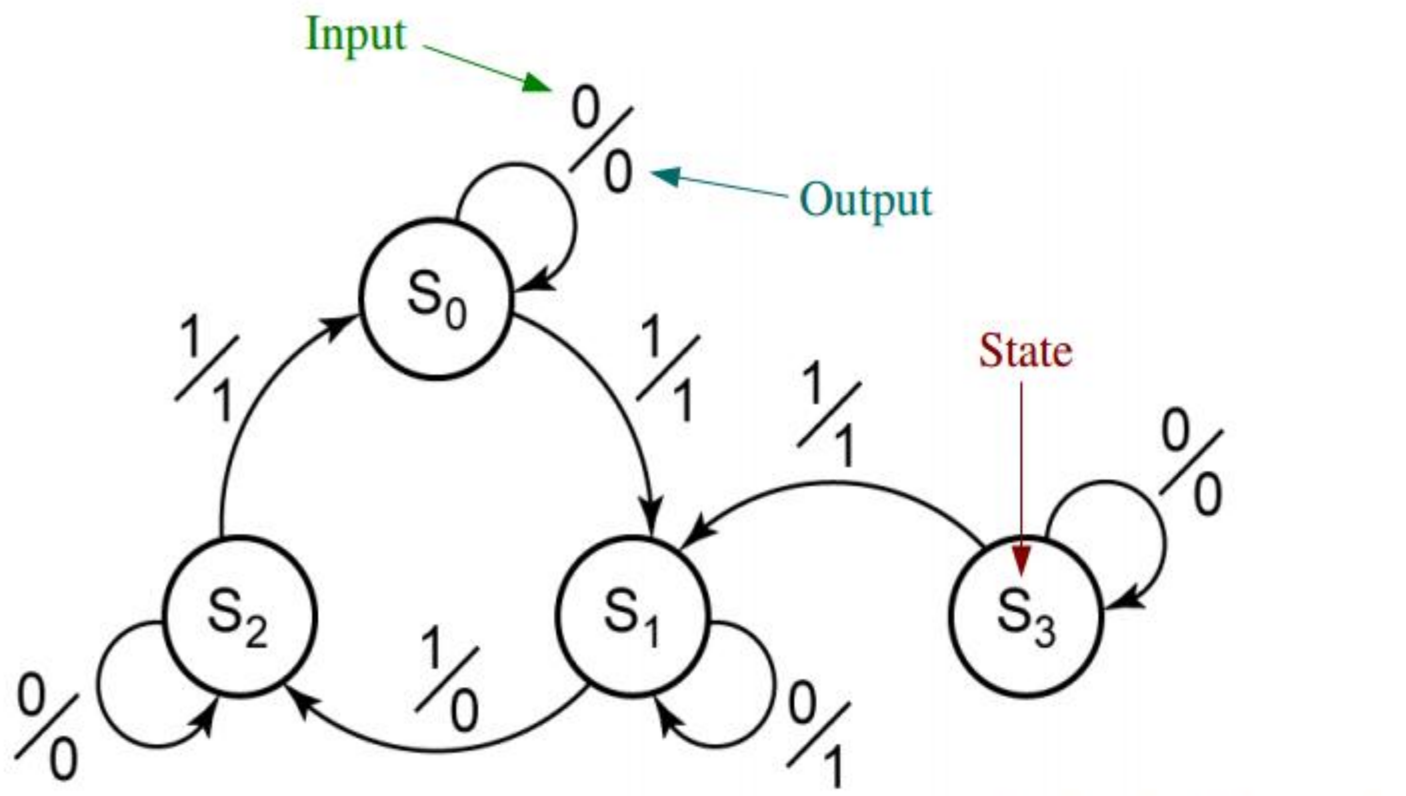
Sequential Circuits: State Diagram



Moore Machine

Each node in the graph represents a state in the sequential circuit.

Sequential Circuits: State Diagram



Each node in the graph represents a state in the sequential circuit.

Mealy Machine

Terms Reviewed

- Latch

Two NANDs (or NORs) used to store one bit

- Gated latch

Latch with an control enable, called Clk

Two basic types: SR and D, both level sensitive

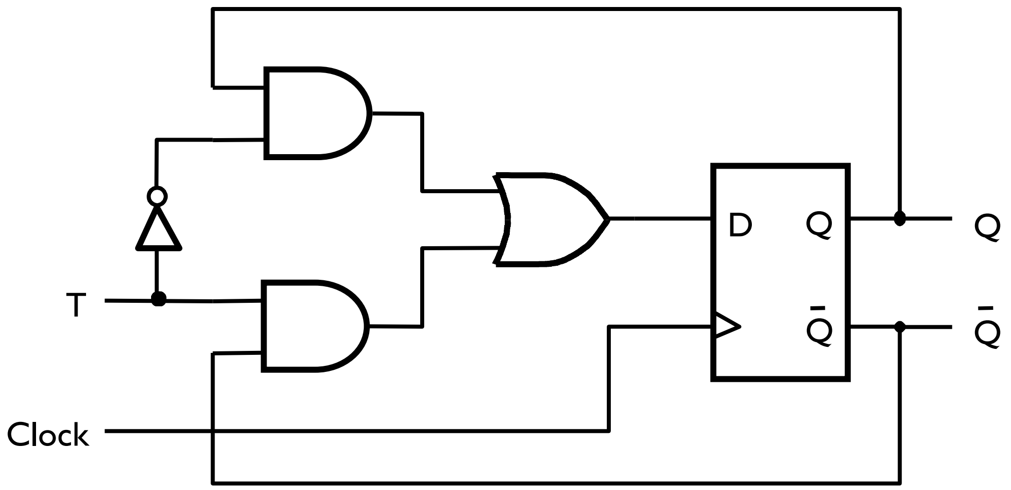
- Master-slave flip-flop

    State changes only on clock edge; made from two gated D latches

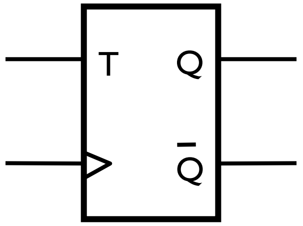
- Edge-triggered flip-flop

    Same as MS FF with fewer transistors

- T Flip-Flop
  - Toggle flip-flop
    - Output toggles on clock edge
    - D = T'Q + TQ'

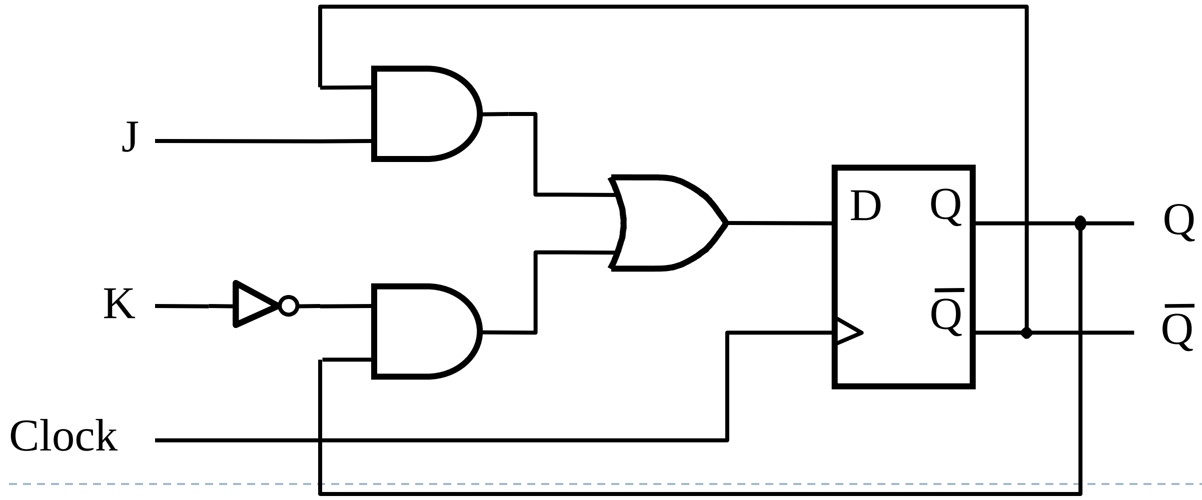


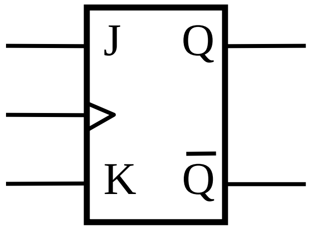
T	Q(t+1)
0	Q(t)
1	Q(t)'



- JK Flip-Flop
  - JK behaves just like SR but removes the S=R=1 problem
    - Output toggles on clock edge when J = K = 1
    - D = JQ' + K'Q

J	K	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	Q(t)'





State Diagrams: D

- The D flip-flop has the following *state table*
  - Note that changes on clock edge are always assumed

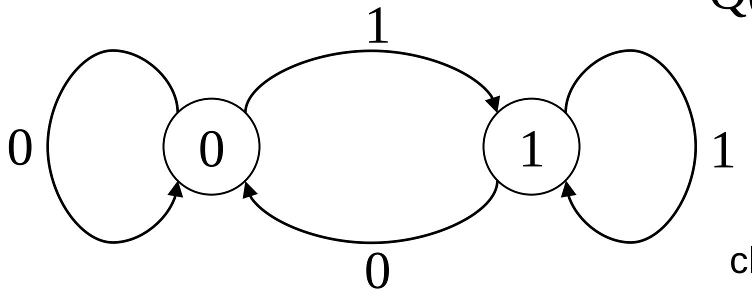
D	Q(t+1)
0	0
1	1

Q \ D	0	1
0	0	1
1	0	1

$Q(t+1) = D$

characteristic equation

- The corresponding *state diagram* is
  - Again, transitions occurs only on a clock edge



State Diagrams: T

- The T flip-flop state table

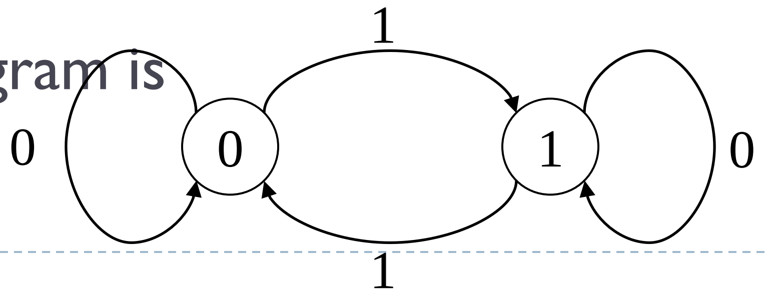
T	Q(t+1)
0	Q(t)
1	Q(t)'

$$\begin{array}{c|cc} Q \backslash T & 0 & 1 \\ \hline 0 & 0 & \boxed{1} \\ 1 & \boxed{1} & 0 \end{array}$$

$$Q(t+1) = TQ(t)' + T'Q(t) = T \oplus Q(t)$$

characteristic equation

- The state diagram is



State Diagrams: SR

- The SR flip-flop state table

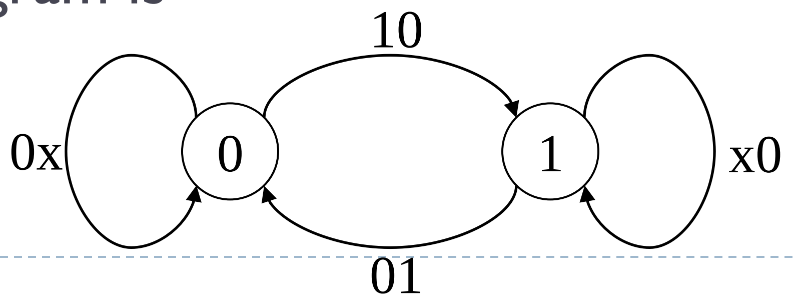
S	R	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	x

Q \ S R	00	01	11	10
0	0	0	x	1
1	1	0	x	1

$$Q(t+1) = S + R'Q(t)$$

characteristic equation

- The state diagram is



State Diagrams: JK

- The JK flip-flop state table

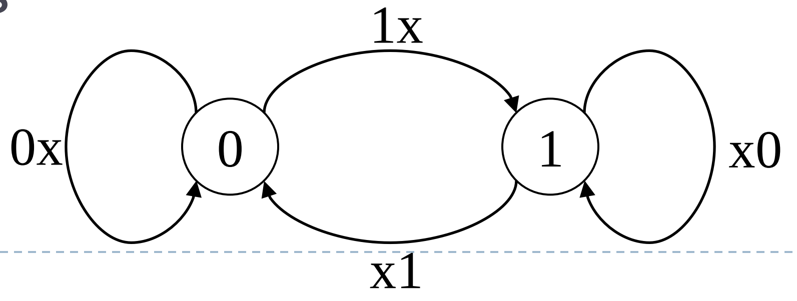
J	K	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	Q(t)'

Q \ J K	00	01	11	10
0	0	0	1	1
1	1	0	0	1

Q(t+1) = J Q(t)' + K' Q(t), or
Q(t+1) = J Q(t)' + K' Q(t) + J K'

characteristic equation

- The state diagram is



Characteristic Equations

- Summary of the characteristic equations
  - How is the next state determined from the inputs and current state?

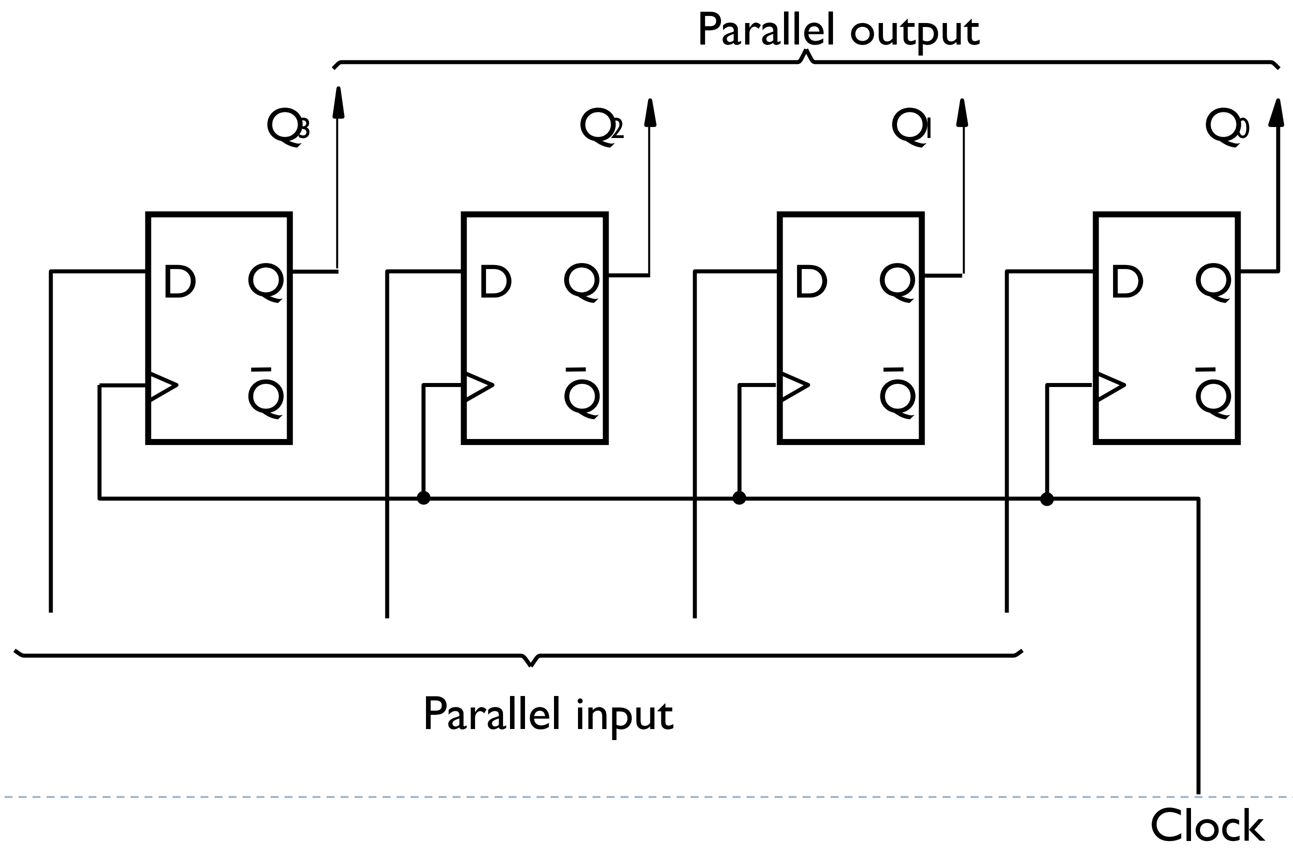
Flip-flop	Characteristic Equation
D	$Q(t+1) = D$
T	$Q(t+1) = T \oplus Q(t)$
SR	$Q(t+1) = S + R' Q(t)$
JK	$Q(t+1) = J Q(t)' + K' Q(t)$

Registers

- A flip-flop stores one bit of information
- When you want to store n bits → register
  - n flip-flops used
  - Clock is shared by all so action is synchronous with clock edge
- Some common register types
  - Simple register
  - Shift register
  - Counters: up counter, down counter, BCD counter, ring counter, Johnson counter

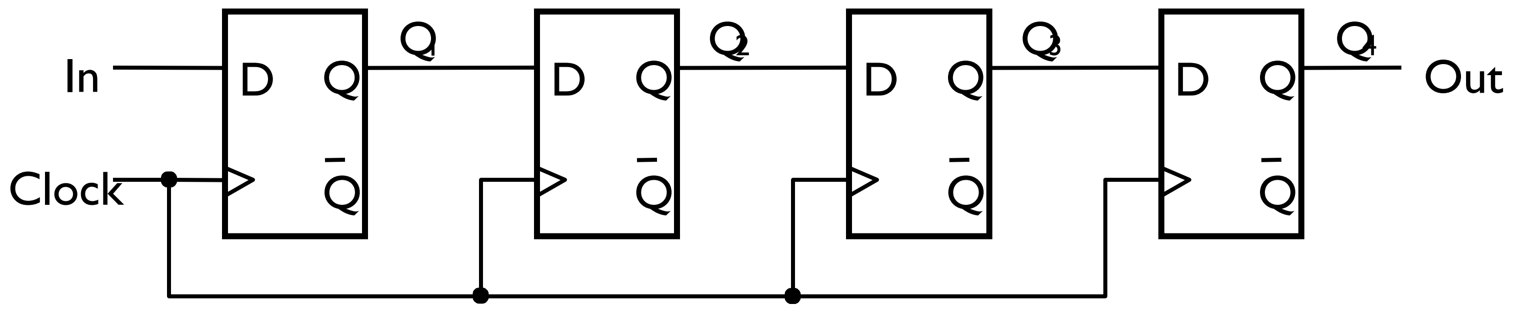
Simple 4 Bit Register

- A standard 4 bit register using D flip flops



Simple Shift Register

- Provide only serial in/out access



- Action of Shift Register

	In	Q_1	Q_2	Q_3	Q_4 = Out
t_0	1	0	0	0	0
t_1	0	1	0	0	0
t_2	1	0	1	0	0
t_3	1	1	0	1	0
t_4	1	1	1	0	1
t_5	0	1	1	1	0
t_6	0	0	1	1	1
t_7	0	0	0	1	1

- Can you use a level sensitive gated latch instead of a flip-flop?
  - No! → The values would propagate during Clock = 1