



Revision

Long Tan Le

HCMC University of Technology

tanlong.ce@gmail.com



- **Arithmetic Circuits**
 - Binary {Addition, Subtraction, Multiplication, Division}
 - Number Representation
 - Adders, Subtractors, BCD Adder, Carry Look-Ahead Adder
- **Counters & Registers**
 - Shift Register, Integrated-Circuit Registers
 - Asynchronous Counter, Synchronous Counter
 - Ring Counter, Johnson Counter
 - ICs for Counters
- **MSI Logic Circuits**
 - Multiplexer , Demultiplexer
 - Decoder, Encoder
 - Comparator
 - Logic Implementation using MSI circuits
 - ICs for MSI Circuits

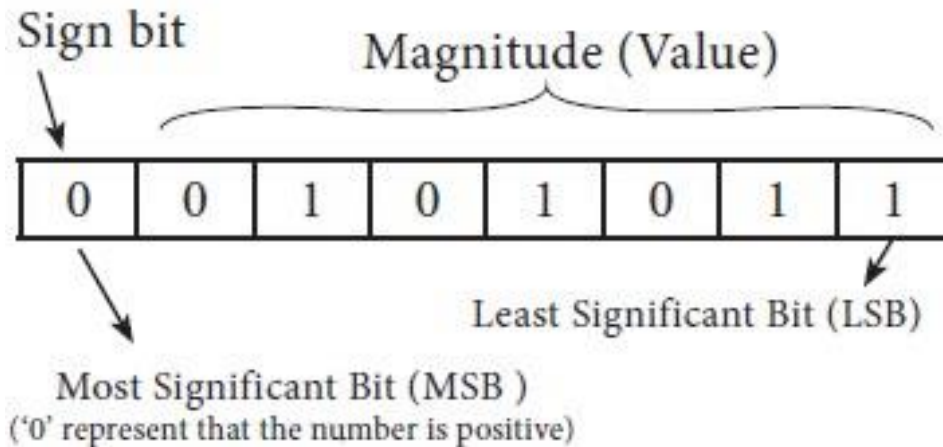
Chapter 1,2,3,4
Self-revision!

Topic 1

ARITHMETIC CIRCUITS

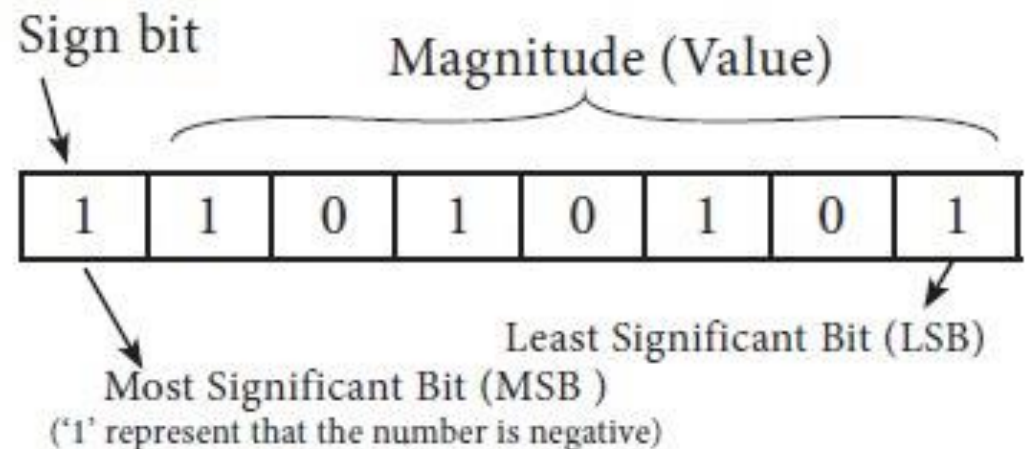
Representation for Binary Number

- Signed Magnitude



- Only (n-1) bit used for storing value
- the left most bit is considered as **SIGN BIT**

0 – Positive
1 – Negative



Representation for Binary Number

- 1's Complement
 - Invert all bits (i.e. Change 1 as 0 and 0 as 1)
- 2's Complement
 - Invert all bits (i.e. Change 1 as 0 and 0 as 1)
 - Add 1 to the result to the Least Significant Bit (LSB)

Binary equivalent of +24:	11000
8bit format:	00011000
1's complement:	11100111
Add 1 to LSB:	+1
2's complement of -24:	11101000

Representation for Binary Number

1's complement

- ☐ Simple implementation which uses only NOT gates for each input bit.
- ☐ Can be used for signed binary number representation but not suitable as ambiguous representation for number 0.
- ☐ 0 has two different representation one is -0 (e.g., 1 1111 in five bit register) and second is +0 (e.g., 0 0000 in five bit register).
- ☐ For k bits register, positive largest number that can be stored is $(2^{(k-1)}-1)$ and negative lowest number that can be stored is $-(2^{(k-1)}-1)$.
- ☐ end-around-carry-bit addition occurs in 1's complement arithmetic operations. It added to the LSB of result.
- ☐ 1's complement arithmetic operations are not easier than 2's complement because of addition of end-around-carry-bit.
- ☐ Sign extension is used for converting a signed integer from one size to another.

2's complement

- ☐ Uses NOT gate along with full adder for each input bit.
- ☐ Can be used for signed binary number representation and most suitable as unambiguous representation for all numbers.
- ☐ 0 has only one representation for -0 and +0 (e.g., 0 0000 in five bit register). Zero (0) is considered as always positive (sign bit is 0)
- ☐ For k bits register, positive largest number that can be stored is $(2^{(k-1)}-1)$ and negative lowest number that can be stored is $-(2^{(k-1)})$.
- ☐ end-around-carry-bit addition does not occur in 2's complement arithmetic operations. It is ignored.
- ☐ 2's complement arithmetic operations are much easier than 1's complement because of there is no addition of end-around-carry-bit.
- ☐ Sign extension is used for converting a signed integer from one size to another.

Signed Binary Numbers

Table 1.3
Signed Binary Numbers

Decimal	Signed-2's Complement	Signed-1's Complement	Signed Magnitude
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
-0	—	1111	1000
-1	1111	1110	1001
-2	1110	1101	1010
-3	1101	1100	1011
-4	1100	1011	1100
-5	1011	1010	1101
-6	1010	1001	1110
-7	1001	1000	1111
-8	1000	—	—

Representation for Binary Number

1-1. The 1's complement form of the number 01001100_2 ?

A. 10110100

C. 10110011

B. 00110010

D. None of above

1-2. Using 5 bits to represent each number, the representations of 13 and -13 in signed 2's complement integers:

A. 01101 & 11101

C. 01101 & 10011

B. 01101 & 10010

D. None of above

1-3. Sign extension from 5 bits to 8 bits represent -13 in signed 2's complement integers of the above question:

A. 00011101

C. 00010011

B. 11110010

D. None of above

Adding Two 1-bit Numbers

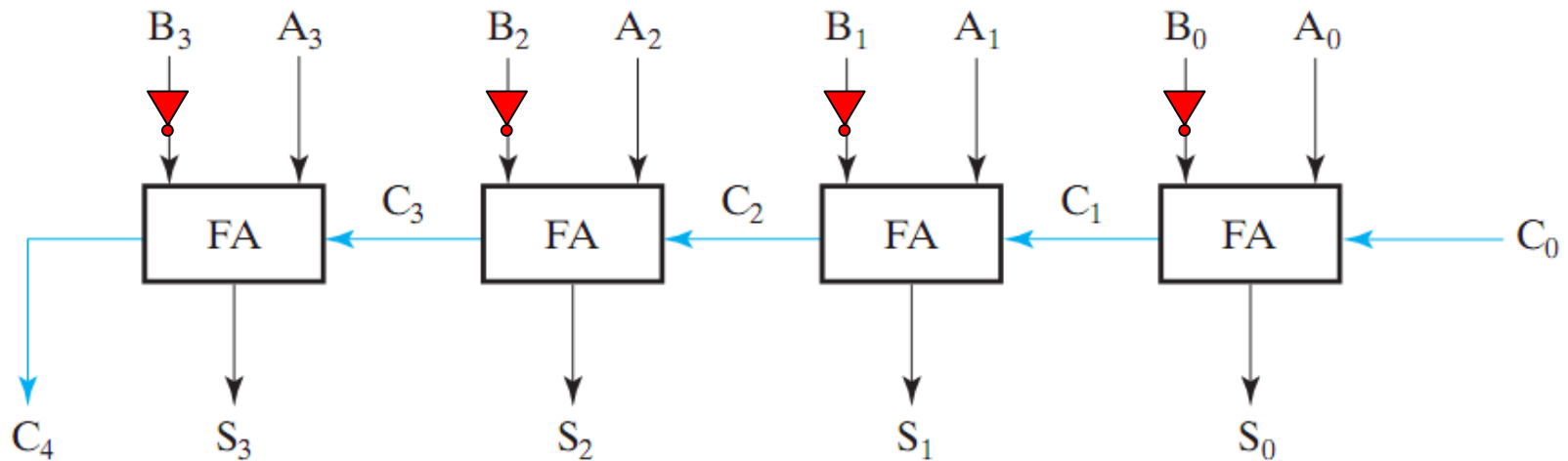
- Performs the addition of two binary bits.
- Four possible operations:
 - $0+0=0$
 - $0+1=1$
 - $1+0=1$
 - $1+1=10$
- Circuit implementation requires 2 outputs; one to indicate the *sum* and another to indicate the *carry*.

Addition of Two n -bit numbers

$$\begin{array}{r}
 \begin{array}{|c|c|c|c|} \hline 1 & 1 & 1 & 1 \\ \hline \end{array} \\
 \swarrow \quad \searrow \quad \swarrow \quad \searrow \\
 + \quad 1 \ 0 \ 1 \ 1 \\
 \quad 0 \ 1 \ 0 \ 1 \\
 \hline
 1 \ 0 \ 0 \ 0 \ 0
 \end{array}$$

- * We start from the **lsb**
- * Add the corresponding pair of bits and the **carry in**
- * Produce a **sum bit** and a **carry out**

Subtraction (2's Complement)



$$S = A + (-B)$$

If we represent signed numbers in 2's complement form, subtraction is the same as addition to negative (2's complemented) number.

Addition in the 2's Complement System

- Perform normal binary addition of magnitudes.

$$\begin{array}{r}
 +9 \rightarrow \begin{array}{|c|} \hline 0 \\ \hline \end{array} \begin{array}{c} 1001 \\ \hline 00100 \\ \hline 01101 \end{array} \quad (+13) \\
 +4 \rightarrow \begin{array}{|c|} \hline 0 \\ \hline \end{array} \begin{array}{c} 1001 \\ \hline 00100 \\ \hline 01101 \end{array} \\
 \hline
 \end{array}$$

↑ Sign bit

$$\begin{array}{r}
 +9 \rightarrow \begin{array}{|c|} \hline 0 \\ \hline \end{array} \begin{array}{c} 1001 \\ \hline 1100 \\ \hline 00101 \end{array} \quad (+5) \\
 -4 \rightarrow \begin{array}{|c|} \hline 1 \\ \hline \end{array} \begin{array}{c} 1001 \\ \hline 1100 \\ \hline 00101 \end{array} \\
 \hline
 \end{array}$$

↑ Sign bit

Don't care carry

$$\begin{array}{r}
 -9 \rightarrow \begin{array}{|c|} \hline 1 \\ \hline \end{array} \begin{array}{c} 1001 \\ \hline 00100 \\ \hline 11011 \end{array} \quad (-5) \\
 +4 \rightarrow \begin{array}{|c|} \hline 0 \\ \hline \end{array} \begin{array}{c} 1001 \\ \hline 00100 \\ \hline 11011 \end{array} \\
 \hline
 \end{array}$$

↑ Sign bit

$$\begin{array}{r}
 -9 \rightarrow \begin{array}{|c|} \hline 1 \\ \hline \end{array} \begin{array}{c} 1001 \\ \hline 1100 \\ \hline 10011 \end{array} \quad (-13) \\
 -4 \rightarrow \begin{array}{|c|} \hline 1 \\ \hline \end{array} \begin{array}{c} 1001 \\ \hline 1100 \\ \hline 10011 \end{array} \\
 \hline
 \end{array}$$

↑ Sign bit

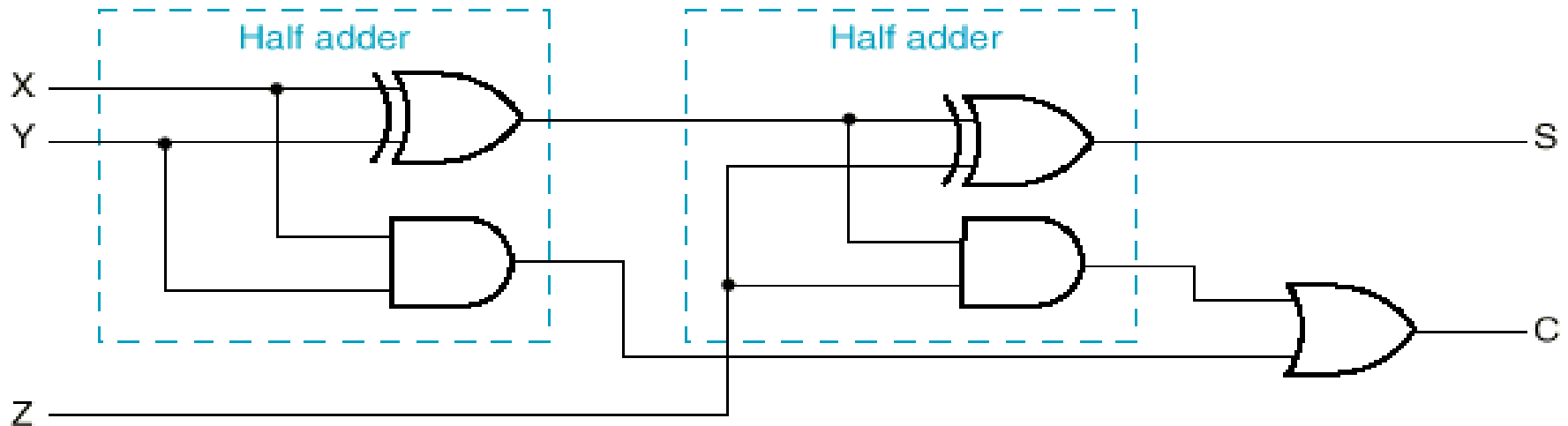
Don't care carry

$$\begin{array}{r}
 +9 \rightarrow \begin{array}{|c|} \hline 0 \\ \hline \end{array} \begin{array}{c} 1001 \\ \hline 0111 \\ \hline 00000 \end{array} \quad (0) \\
 -9 \rightarrow \begin{array}{|c|} \hline 1 \\ \hline \end{array} \begin{array}{c} 1001 \\ \hline 0111 \\ \hline 00000 \end{array} \\
 \hline
 \end{array}$$

↑ Sign bit

Don't care carry

Full Adder from Half Adders



1-4. How many AND, OR and EX-OR gates are requires for the configuration of full adder?

A. 1, 2, 2

B. 2, 1, 2

C. 3, 1, 2

D. 4, 0 ,1

Adders & Subtractors

1-5. Half-adders have a major limitation in that they cannot:

- A. Accept a carry from a present stage
- B. Accept a carry bit from a next stage
- C. Accept a carry bit from a previous stage**
- D. Accept a carry bit from the following stages

1-6. Full subtractor is used to perform subtraction of?

- A. 2 bits**
- B. 3 bits
- C. 4 bits
- D. 8 bits

1-7. The output of a full subtractor is same as:

- A. Half Adder
- B. Full Adder**
- C. Half Subtractor

Multiplication & Division

1-8. Perform multiplication of the binary number: 01001 x 01011

A. 001100011

C. 010100110

B. 110011100

D. 101010111

1-9. The quotient of 111101 ÷ 1001 is

A. 0010

C. 1100

B. 1010

D. 0011

1-10. In binary division process, a series of performed operations is said to be of

A. Additions

C. Multiplications

B. Subtractions

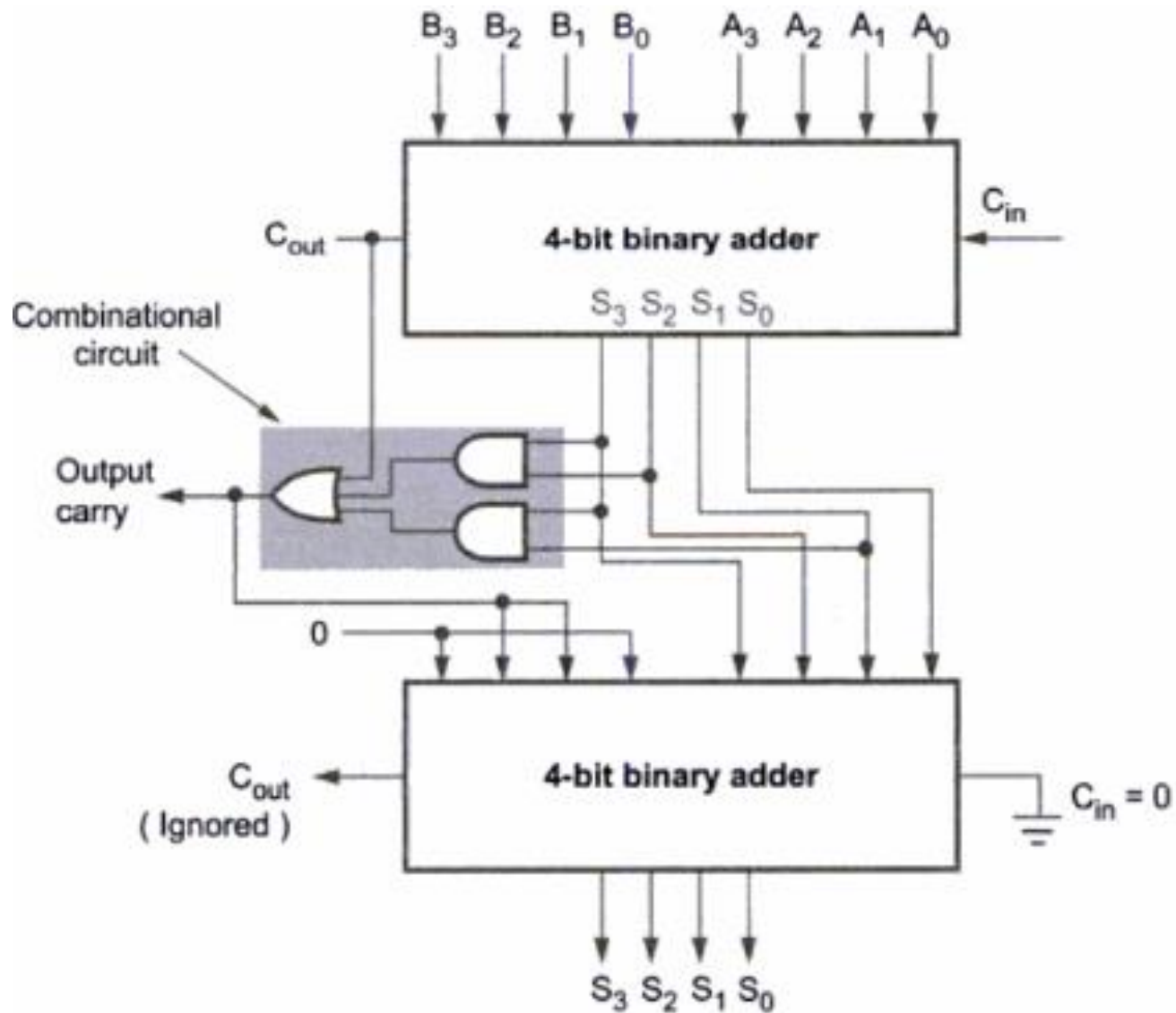
D. Complement

BCD Adder

BCD Adder

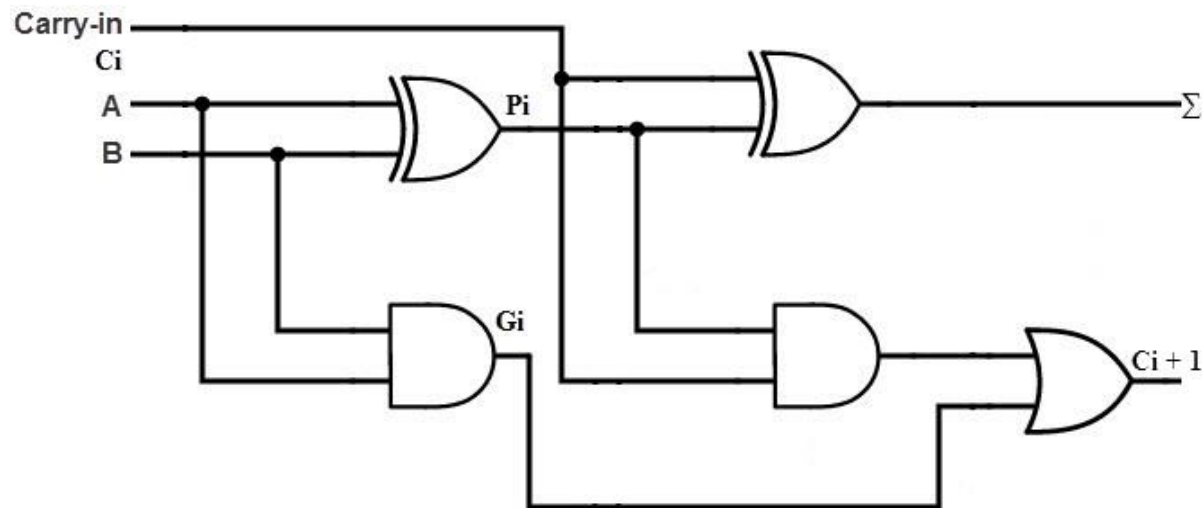
- When the sum of two digits is less than or equal to 9 then the ordinary 4-bit adder can be used
- But if the sum of two digits is greater than 9 then a correction must be added “**i.e. adding 0110**”
- We need to design a circuit that is capable of doing the correct addition

BCD Adder

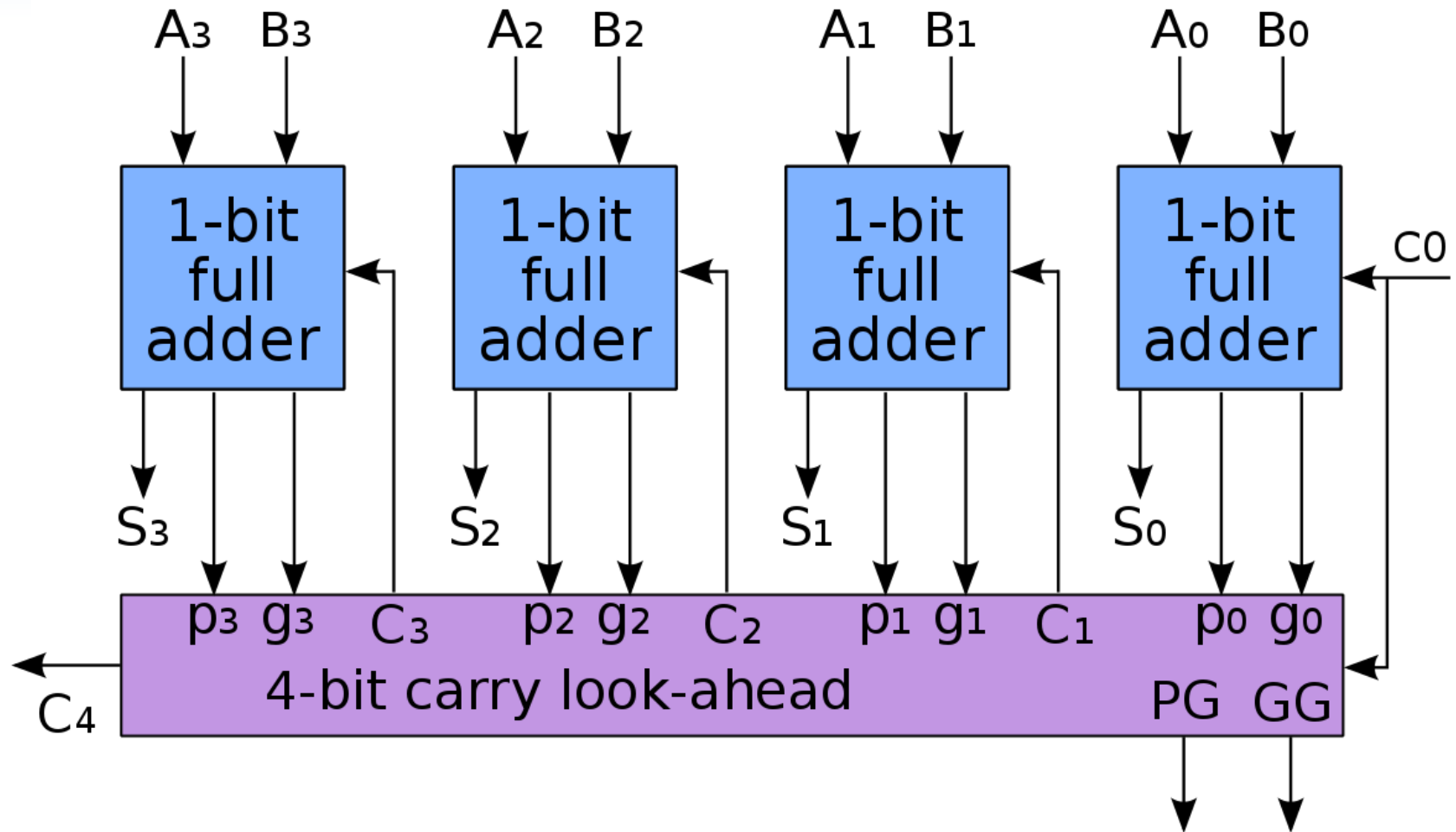


Carry Look Ahead (CLA) Adder

- Used in most ALU designs
- **Faster** than ripple carry logic adders or full adders
 - Especially when adding a large number of bits.
- Be able to **generate carries before the sum**
 - Using the **Carry Propagation (P_i)** and **Carry Generation (G_i)** logic to make addition much faster.



Carry Look Ahead (CLA) Adder



Other Adders

1-11. One of the advantages of the fast carry or look-ahead carry circuits found in most 4-bit parallel-adder circuits

- A. Add a 1 to complemented inputs
- B. Increase ripple delay
- C. Determine sign and magnitude
- D. Reduce propagation delay**

1-12. The **carry generate (CG) and **carry propagate (CP)** function of the Carry Look-Ahead adder is**

- A. $CG = A \text{ or } B$, $CP = A \text{ xor } B$
- B. $CG = A \text{ xor } B$, $CP = A \text{ or } B$
- C. $CG = A \text{ and } B$, $CP = A \text{ xor } B$**
- D. $CG = A \text{ and } B$, $CP = A \text{ or } B$

Topic 2

COUNTERS AND REGISTERS

Registers

2-1. A register is defined as

- A. The group of latches for storing one bit of information
- B. The group of latches for storing n-bit of information
- C. The group of flip-flops suitable for storing one bit of information
- D.** The group of flip-flops suitable for storing binary information

2-2. A shift register is defined as

- A. The register capable of shifting information to another register
- B.** The register capable of shifting information either to the right or to the left
- C. The register capable of shifting information to the right only
- D. The register capable of shifting information to the left only

Registers

2-3. The full form of SIPO is

A. Serial-in Parallel-out

C. Serial-in Serial-out

B. Parallel-in Serial-out

D. Serial-in Peripheral-out

2-4. The **group of bits 11001 is serially shifted (right-most bit first) into a 5-bit parallel output shift register with an initial state 01110. After three clock pulses, the register contains?**

A. 01110

C. 00101

B. 00001

D. 00110

2-5. With a 200 KHz clock frequency, eight bits can be serially entered into a shift register in ?

A. 4 μ s

C. 400 μ s

B. 40 μ s

D. 40 ms

Asynchronous (Ripple) Counter

This type of counter is easiest to design, and requires the least amount of hardware

Called Asynchronous

- The flip-flops are not driven by the same clock signal

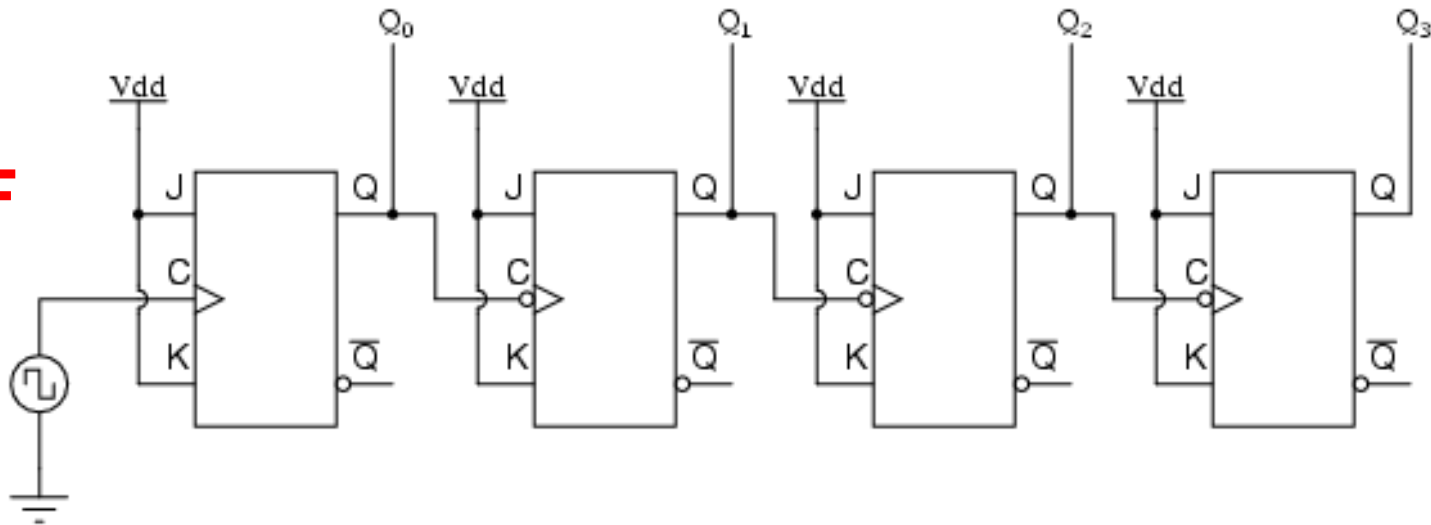
Called **Ripple**

- Propagation Delay causes Ripple Effect
- Can lead to unwanted transitions
- Glitches are possible

Design a Ripple Counter

A four-bit "up" counter

Up Counter
using
JK FF



J & K at HIGH level

Use Negative Edge Triggered (NGT) FFs:

- Connect **Q output** of the previous to **CLK input** of the next


Use Positive Edge Triggered (PGT) FFs:

- Connect **Q' output** of the previous to **CLK input** of the next

Design a Ripple Counter

- Suppose we want to design a **module-M counter**
 - Counter will count up from **0** to **M-1**, and then back 0
- Basic idea:

We first design a full-modulus ripple counter with $\lceil \log_2 M \rceil$ (ceiling) flip-flops (e.g.: 4 FF \rightarrow MOD-16)



Then we design a gating circuit, which takes inputs from the counter outputs, and generate a 0 whenever the count value reaches M

Connect the output of the gating circuit to the **CLR' inputs** of the flip-flops

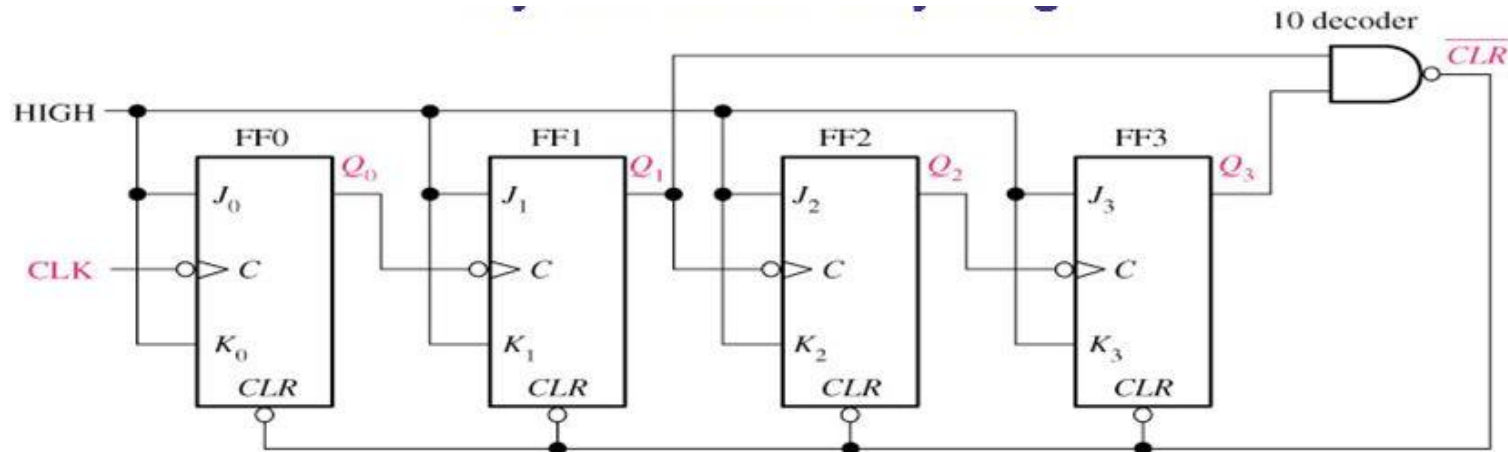
Design a Ripple Counter

- Example: Design a MOD – 10 (Decade) Up Ripple Counter using JK Flip-Flop

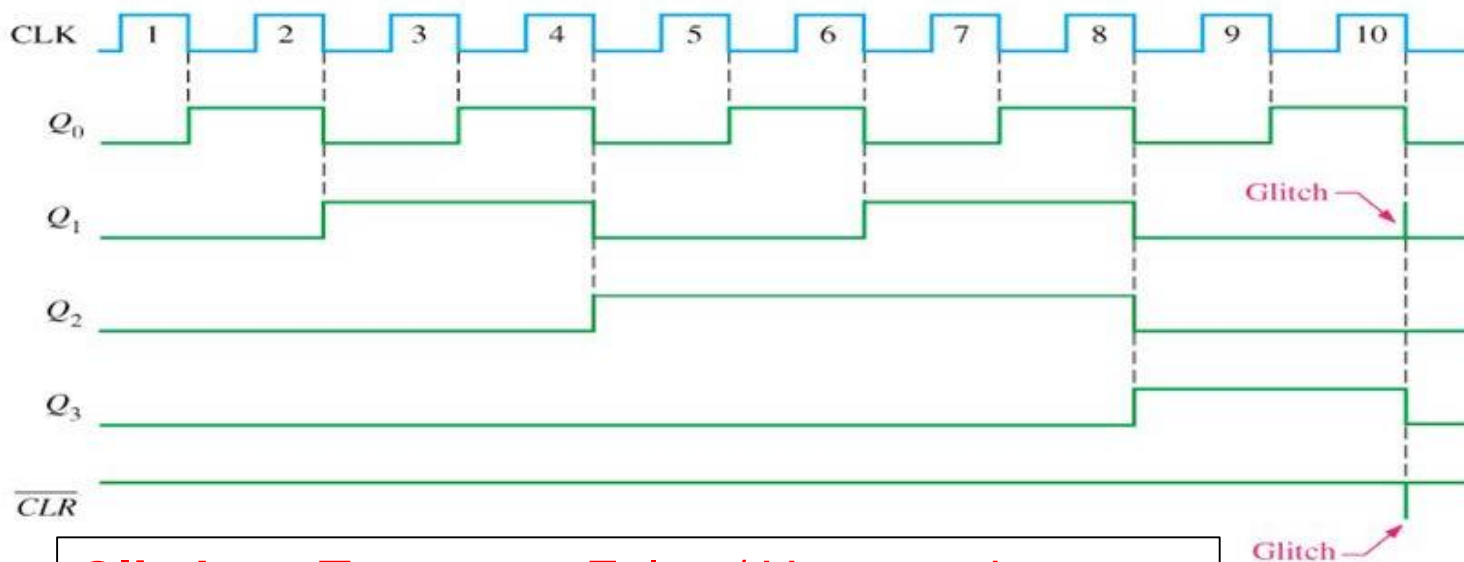
Input Pulses	D	C	B	A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
0	0	0	0	0 (resets)

**Reset
state**

Design a Ripple Counter



(a)

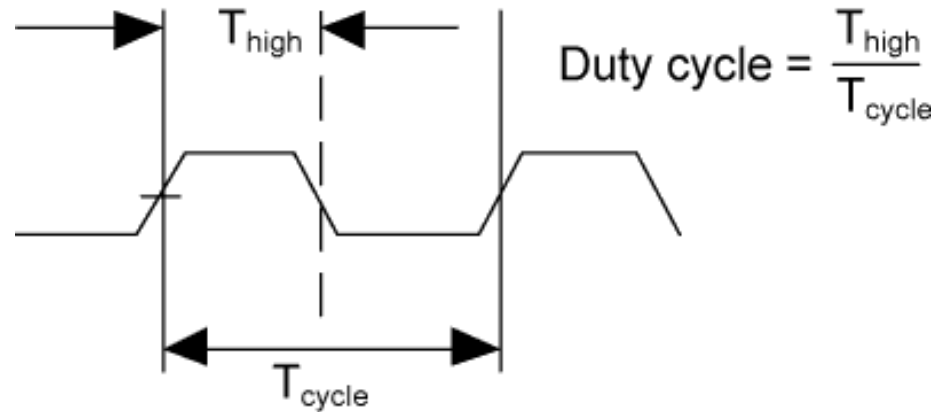


(b)

Glitches: Temporary False / Unwanted outputs

Duty Cycle

- The percentage of time a digital signal is ON over an interval or period of time



50% duty cycle



75% duty cycle



25% duty cycle



Ripple Counters

2-6. Ripple counter's speed is limited by the propagation delay of

- A.** Each flip-flop
- B. All flip-flops and gates
- C. The flip-flops only with gates
- D. Only circuit gates

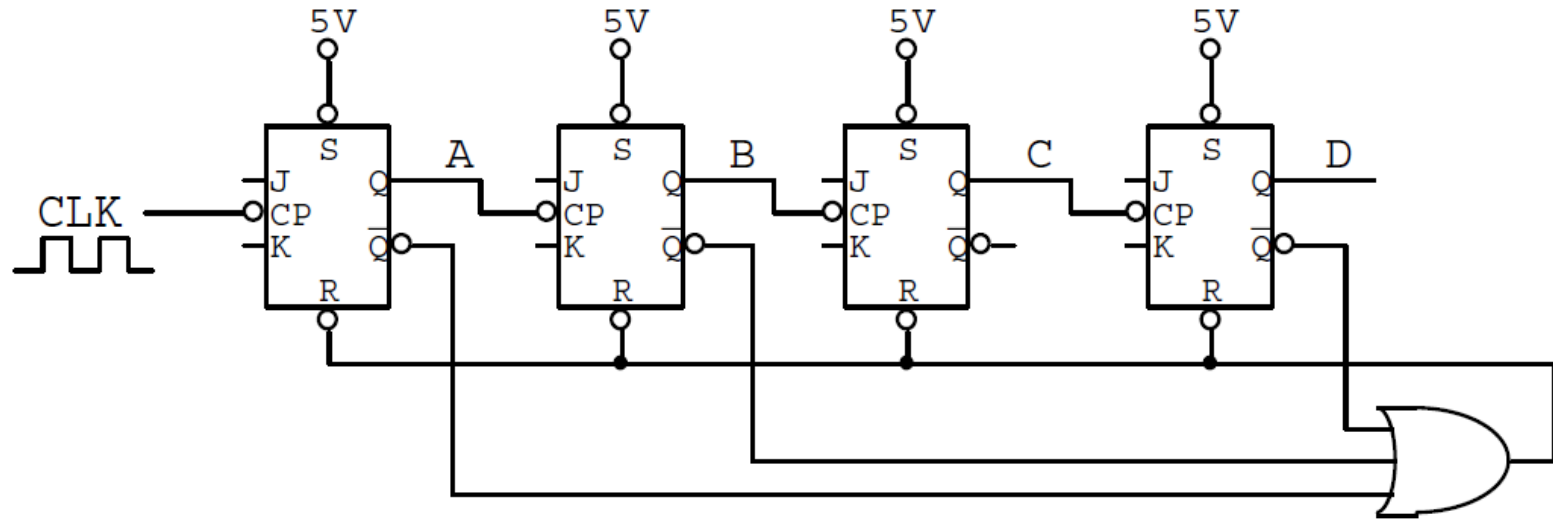
2-7. The duty cycle of the most significant bit from a 4-bit (0–9) BCD counter is

- A.** 20%
- B. 50%
- C. 10%
- D. 50%

2-8. A 5-bit asynchronous binary counter is made up of five flip-flops, each with a 12 ns propagation delay

- A. 12 ms
- B. 24 ns
- C. 48 ns
- D.** 60 ns

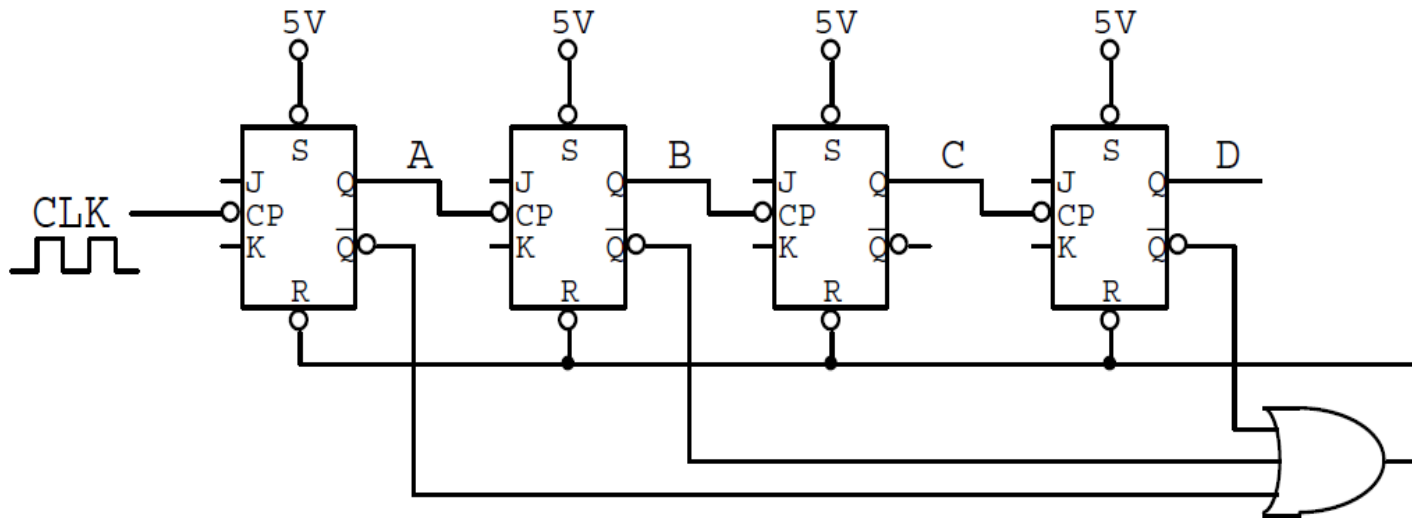
Ripple Counter



2-9. Chose a TRUE statement

- A. The up counter with MOD-12
- B. The down counter with MOD-9
- C. The down counter with MOD-13
- D. The up counter with MOD-11**

Ripple Counter



2-10. Determine the frequency of output C

A. CLK/8

B. CLK/11

C. CLK/12

D. CLK/13

2-11. Determine Duty Cycle (HIGH) of output C.

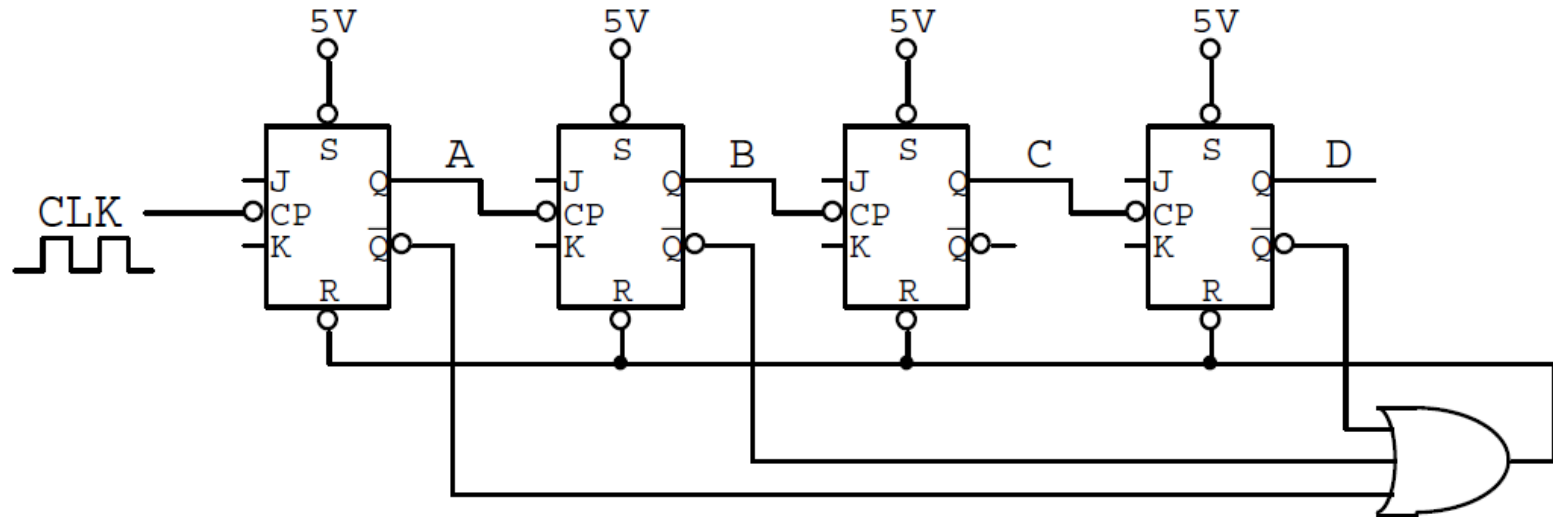
A. 50%

B. 33.33%

C. 36.36%

D. 18.18%

Ripple Counter

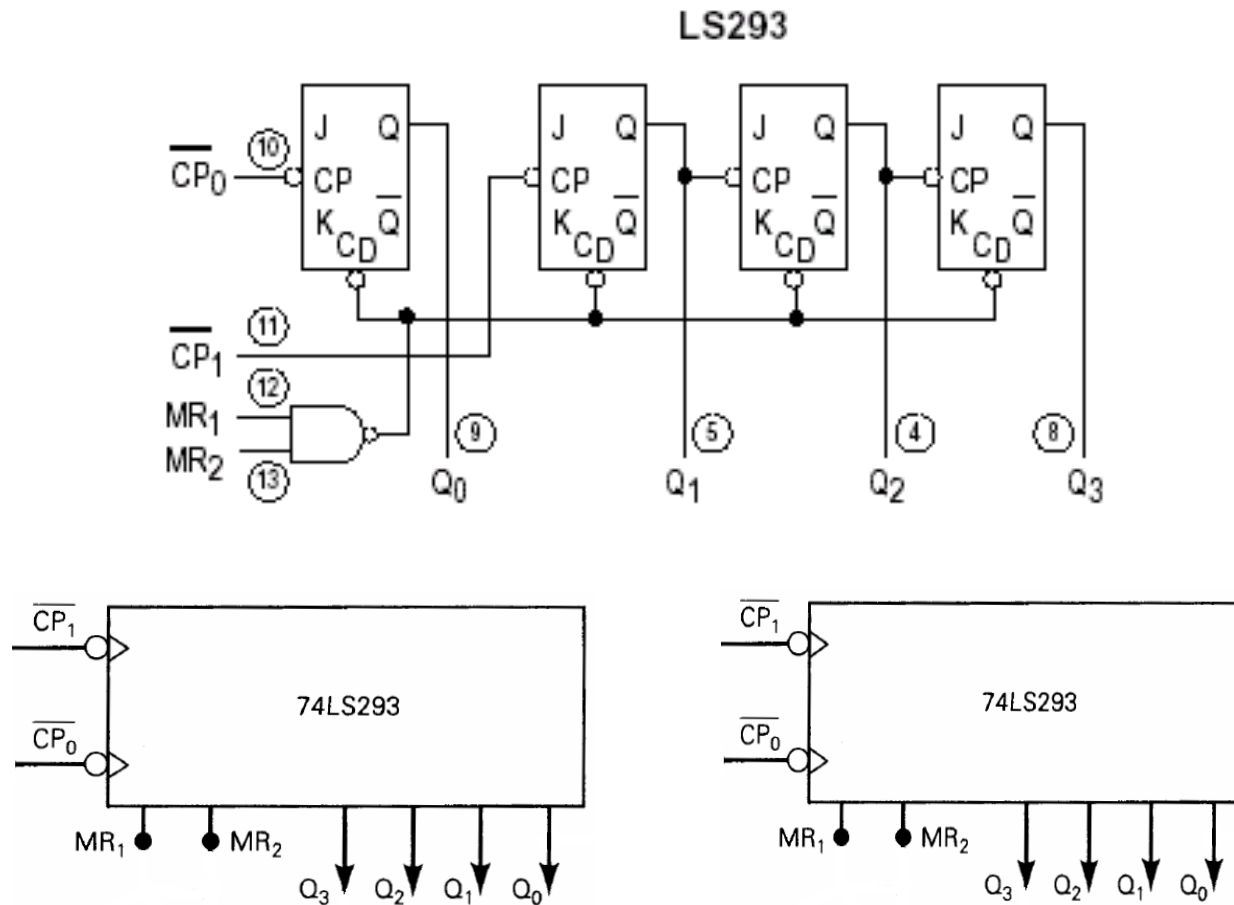


2-12. Determine the output that has glitches.

- A.** Output A
- B. Output B
- C. Output C
- D. Output D

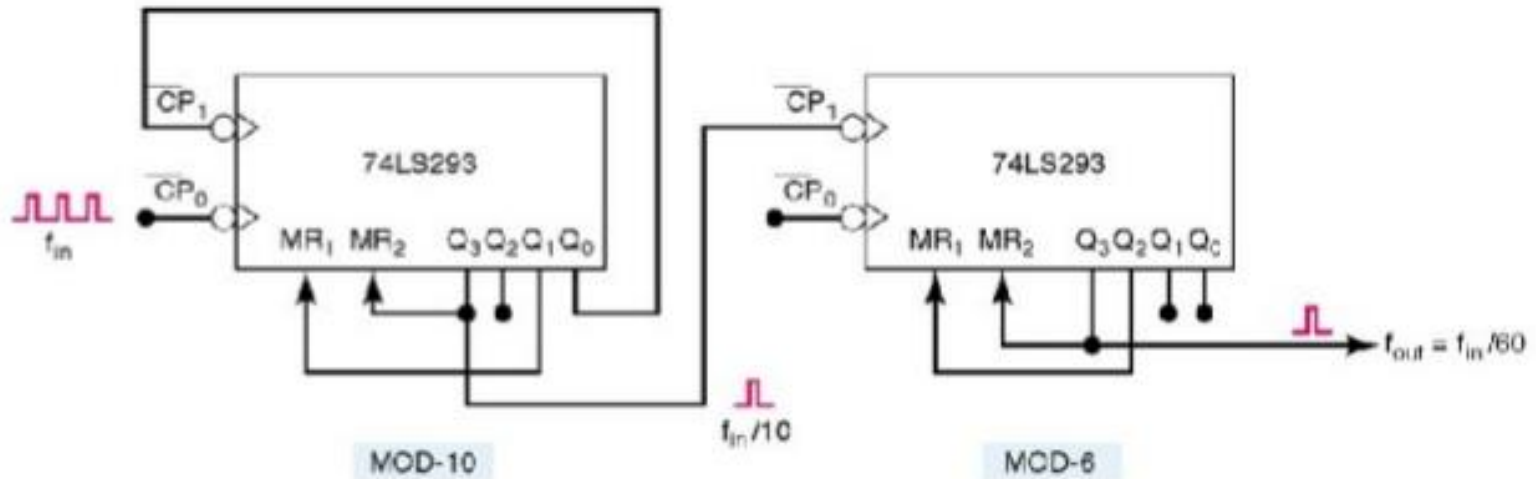
Cascading Ripple Counter

2-13. List which pins need to be connected together on two **74293** to make a MOD-60 counter



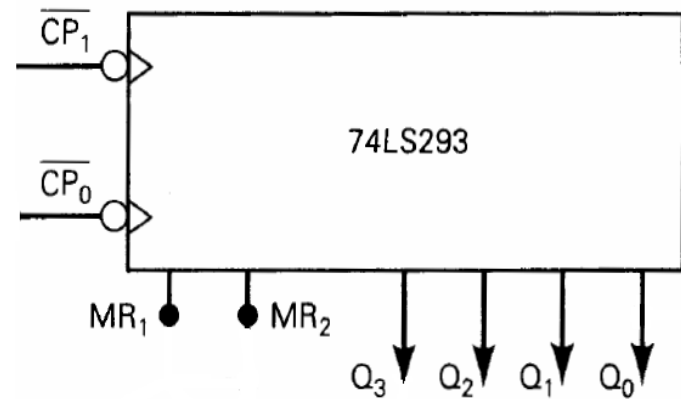
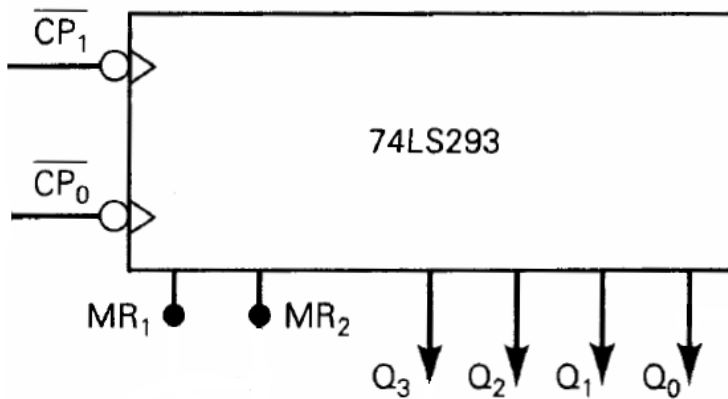
Cascading Ripple Counter

2-13. List which pins need to be connected together on two 74293 to make a MOD-60 counter



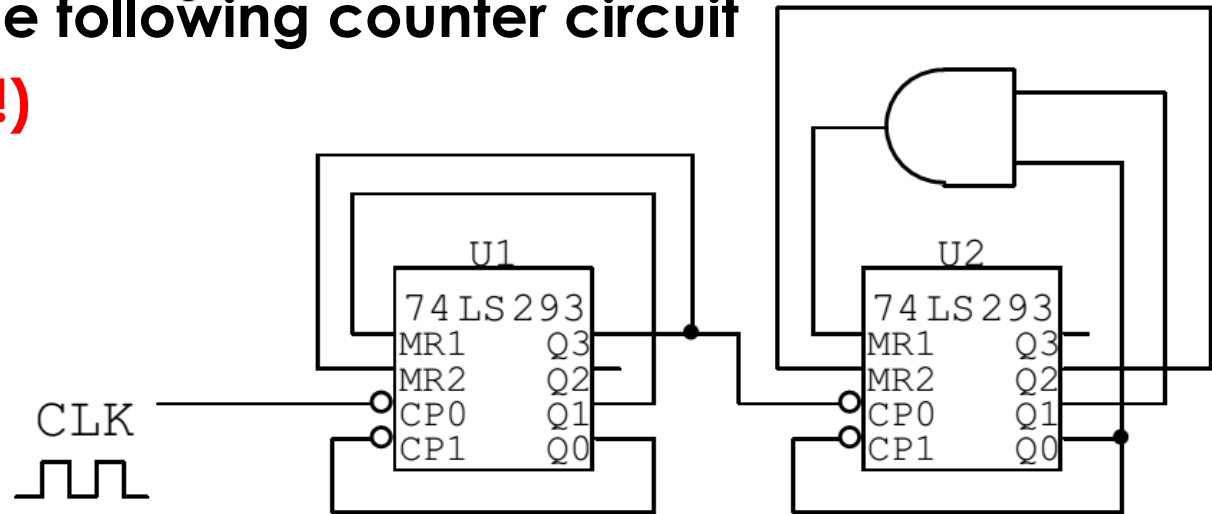
Cascading Ripple Counter

2-14. Design an asynchronous counter MOD-40 (4x10) using only 74LS293 ICs. (Do it yourself!)



Cascading Ripple Counter

**2-15. Given the following counter circuit
(Do it yourself!)**



With the frequency of the clock signal $f_{CLK} = 35 \text{ KHz}$


- What is the MOD of the counter ?
- Determine the frequency of Q3 of U1?
- Determine the frequency of Q2 of U2?
- In the Q3, Q2, Q1, Q0 signals of U1 and U2, which signals are **glitches**?
- Determine **duty cycle** of Q2 of U2?

Synchronous Counter

- All flip-flops are **simultaneously clocked** by an external clock.
- Synchronous counters are **faster than asynchronous counters** because of the simultaneous clocking.
- Synchronous counters are an example of a state machines design.

Design a Synchronous Counter

1. Determine the # of FFs needed to support the counting sequence's highest #.




2. Build a State Transition Diagram. Be sure to include all states.



3. Build a State/Excitation Truth Table.



4. Simplify expressions for J and K inputs for each F/F on K-Maps.



5. Implement the Synchronous Counter/State Machine Circuit. .

Design a Synchronous Counter

Flip-Flop Transition Tables

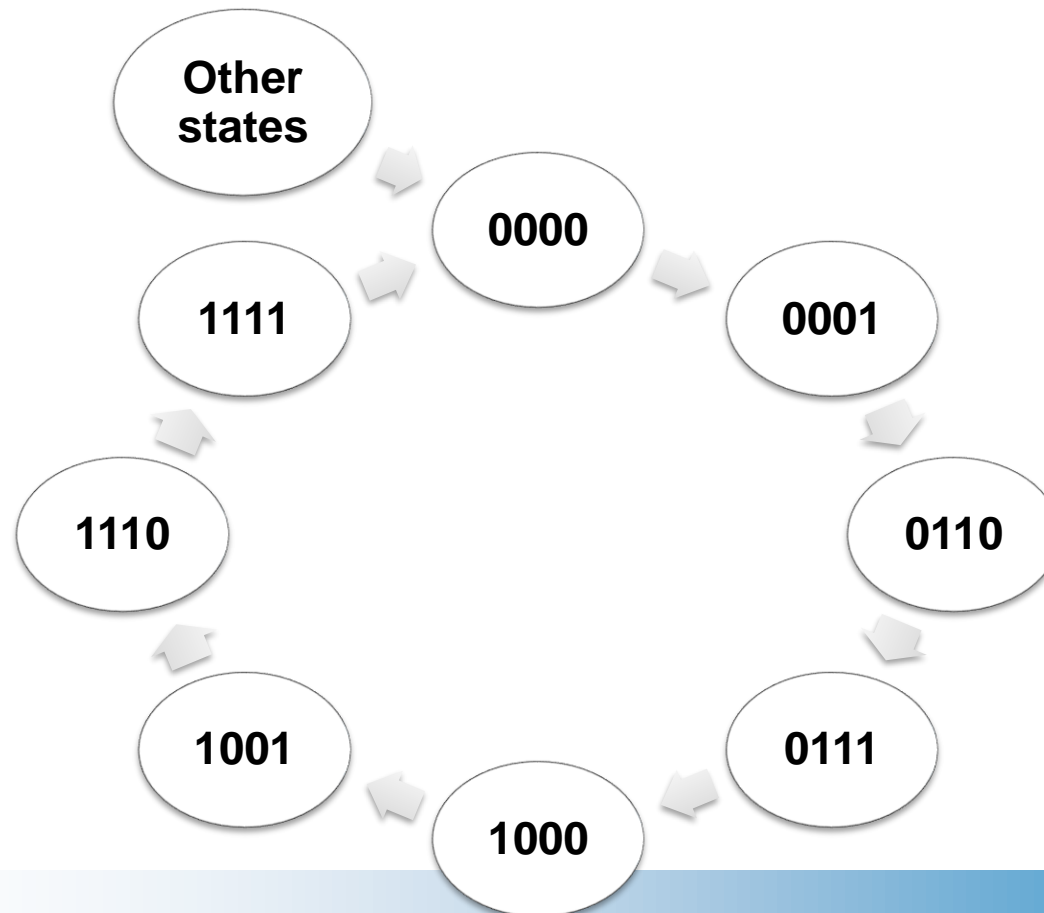
SR Flip-flop				D Flip-flop		
Q(t)	Q(t+1)	S	R	Q(t)	Q(t+1)	DR
0	0	0	X	0	0	0
0	1	1	0	0	1	1
1	0	0	1	1	0	0
1	1	X	0	1	1	1

JK flip-flop				T flip-flop		
Q(t)	Q(t+1)	J	K	Q(t)	Q(t+1)	DR
0	0	0	x	0	0	0
0	1	1	x	0	1	1
1	0	x	1	1	0	1
1	1	x	0	1	1	0

Design a Synchronous Counter

1. For the following stage diagram, design a synchronous counter using **D Flip-Flops**

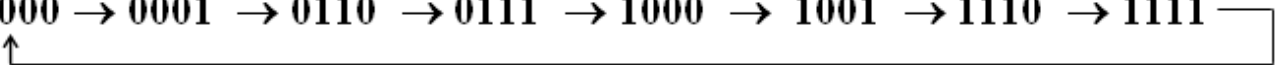
(A: LSB, D: MSB)



Design a Synchronous Counter

- For the following stage diagram, design a synchronous counter using **D Flip-Flops** (A: LSB, D: MSB)

Other states → 0000 → 0001 → 0110 → 0111 → 1000 → 1001 → 1110 → 1111 →



State table (Transition Table)

Present State				Next State				Output			
D	C	B	A	D	C	B	A	D _D	D _C	D _B	D _A

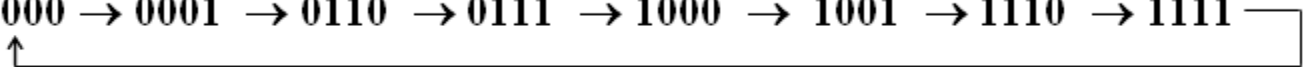
Other states → 0000 → 0001 → 0110 → 0111 → 1000 → 1001 → 1110 → 1111 —
↑

Present State				Next State				Control Signal			
D	C	B	A	D	C	B	A	D _D	D _C	D _B	D _A
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	1	0	1	1	0	0	1	1	0
0	0	1	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	0
0	1	1	0	0	1	1	1	0	1	1	1
0	1	1	1	1	0	0	0	1	0	0	0
1	0	0	0	1	0	0	1	1	0	0	1
1	0	0	1	1	1	1	0	1	1	1	0
1	0	1	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0	0
1	1	1	0	1	1	1	1	1	1	1	1
1	1	1	1	0	0	0	0	0	0	0	0

Design a Synchronous Counter

- For the following stage diagram, design a synchronous counter using **D Flip-Flops** (A: LSB, D: MSB)

Other states → 0000 → 0001 → 0110 → 0111 → 1000 → 1001 → 1110 → 1111 →



K-maps

$D_D =$

$D_C =$

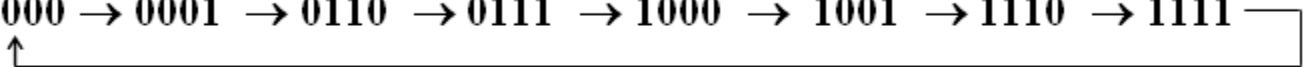
$D_B =$

$D_A =$

Design a Synchronous Counter

1. For the following state diagram, design a synchronous counter using **D Flip-Flops** (A: LSB, D: MSB)

Other states → 0000 → 0001 → 0110 → 0111 → 1000 → 1001 → 1110 → 1111 →



Implement the logic circuit using D Flip-Flops

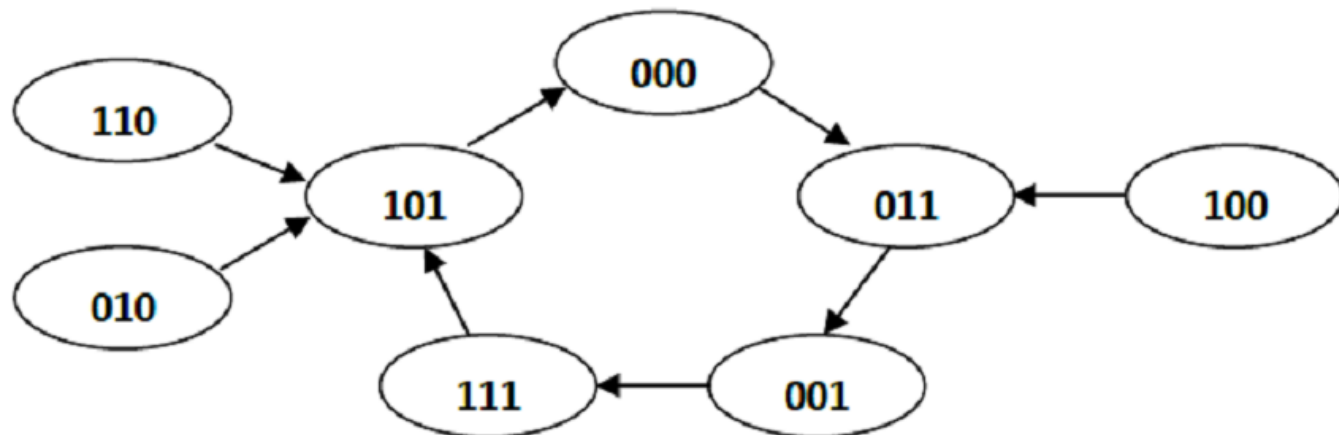
Design a Synchronous Counter

2-16. Design a ripple counter, MOD-12, up counter, using **J-K Flip-Flop** and **D Flip-Flop**

2-17. Use J-K Flip-Flop and D Flip-Flop to design an asynchronous down counter as follows

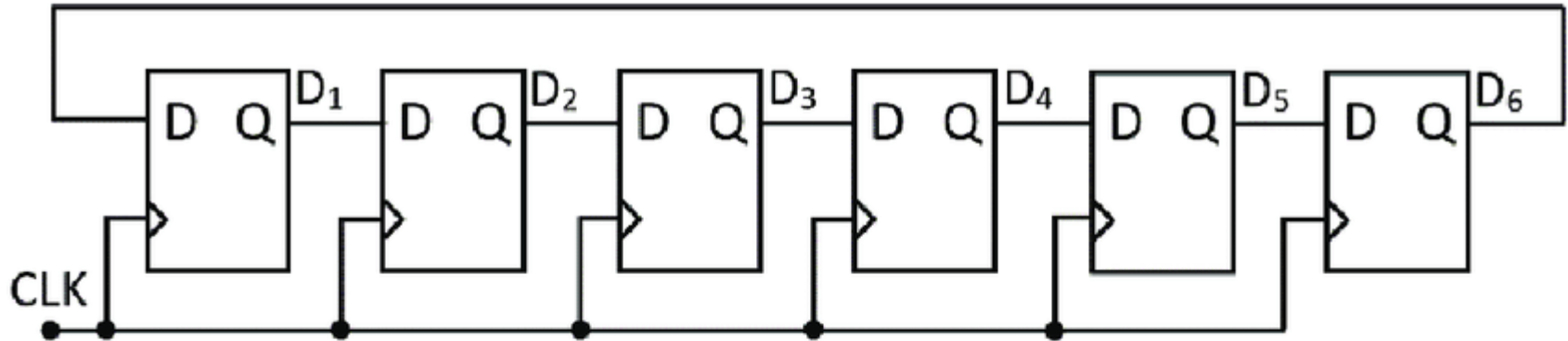
$$6 \rightarrow 5 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 1 \rightarrow 6 \rightarrow \dots$$

2-18. Use J-K Flip-Flop to design a synchronous counter circuits according to the following transition diagram

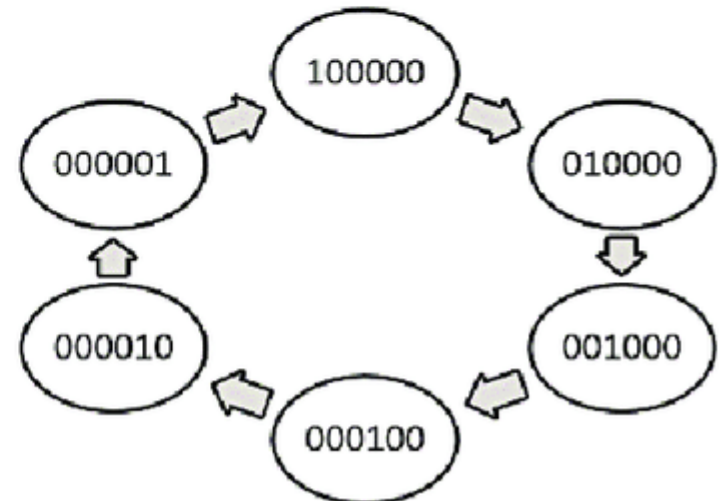


Ring & Johnson Counter

Ring Counter



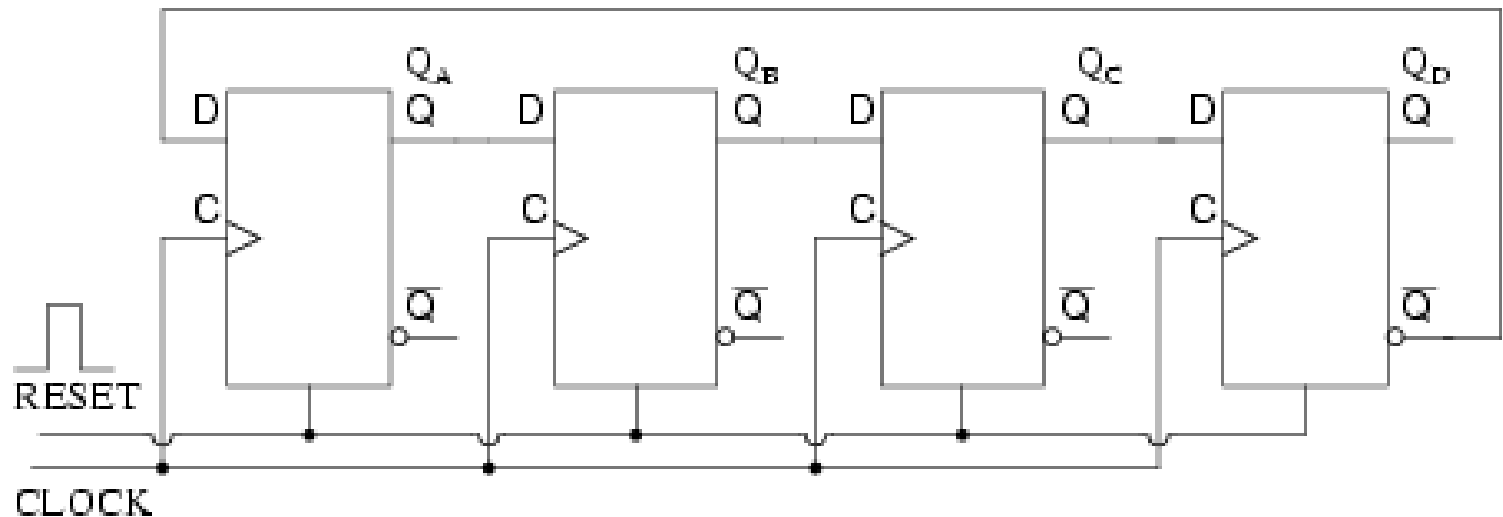
CLK	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆
1	1	0	0	0	0	0
2	0	1	0	0	0	0
3	0	0	1	0	0	0
4	0	0	0	1	0	0
5	0	0	0	0	1	0
6	0	0	0	0	0	1



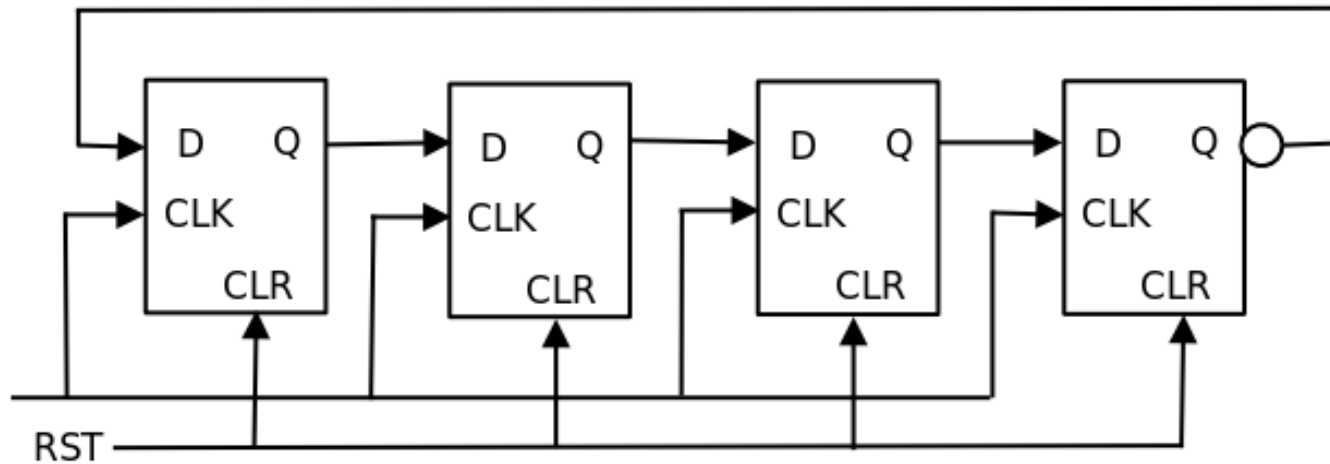
Ring & Johnson Counter

Johnson Counter

Q_A	Q_B	Q_C	Q_D
0	0	0	0
1	0	0	0
1	1	0	0
1	1	1	0
1	1	1	1
0	1	1	1
0	0	1	1
0	0	0	1
repeat			



Ring & Johnson Counter



2-19. On the fifth clock pulse, a 4-bit Johnson sequence is $Q_0 = 0$, $Q_1 = 1$, $Q_2 = 1$, and $Q_3 = 1$. On the sixth clock pulse, the sequence is....

- A. $Q_0 = 1$, $Q_1 = 0$, $Q_2 = 0$, and $Q_3 = 0$
- B. $Q_0 = 1$, $Q_1 = 1$, $Q_2 = 1$, and $Q_3 = 0$
- C. $Q_0 = 0$, $Q_1 = 0$, $Q_2 = 1$, and $Q_3 = 1$**
- D. $Q_0 = 0$, $Q_1 = 0$, $Q_2 = 0$, and $Q_3 = 1$

Ring & Johnson Counter

2-20. What is the difference between a ring shift counter and a Johnson shift counter?

A. There is no difference

C. The feedback is reversed

B. A ring is faster

D. The Johnson is faster

2-21. In a 6-bit Johnson counter sequence there are a total of how many states, or bit patterns ?

A. 2

C. 12

B. 6

D. 24

2-22. A modulus – 12 ring counter requires a minimum of

A. 10 Flip-flops

C. 6 Flip-flops

B. 12 Flip-flops

D. 2 Flip-flops

2-23. A MOD-16 ripple counter is holding the count 1001_2 . What will the count be after 31 clock pulses?

A. 1000_2

C. 1011_2

B. 1010_2

D. 1101_2

2-24. A MOD-12 and a MOD-10 counter are cascaded. Determine the output frequency if the input clock frequency is 60 MHz

A. 500 kHz

C. 6 MHz

B. 1500 kHz

D. 5 MHz

2-25. A BCD counter is a

A. Binary Counter

C. Decade Counter

B. Full-modulus Counter

D. Divide-by-10 Counter

Topic 3

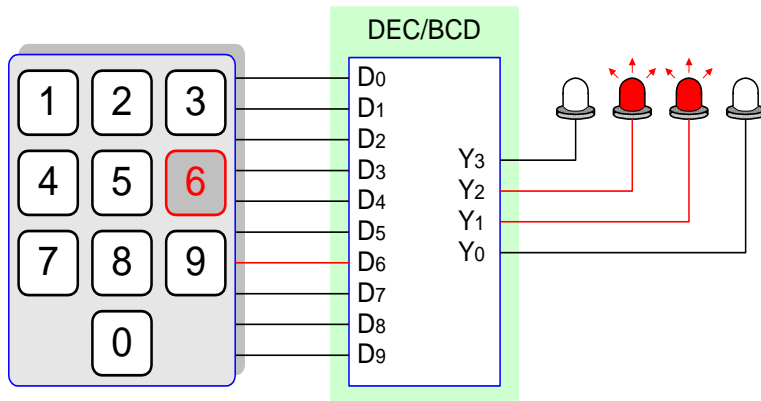
MSI LOGIC CIRCUITS

MSI Devices

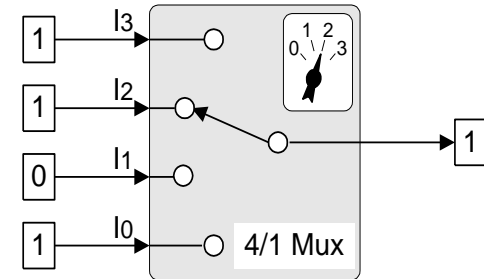
- **Medium Scale Integration (MSI)**
 - Using a **few tens** to **hundreds** of logic gates.
- Used as discrete devices packed in a single Integrated Circuit (IC),
- Building blocks for more complex devices such as memory devices or microprocessors.
- Some typical MSI devices are the following:
 - **Encoders and Decoders**
 - **Multiplexers and Demultiplexers**
 - **Comparator**

Examples of MSI Devices

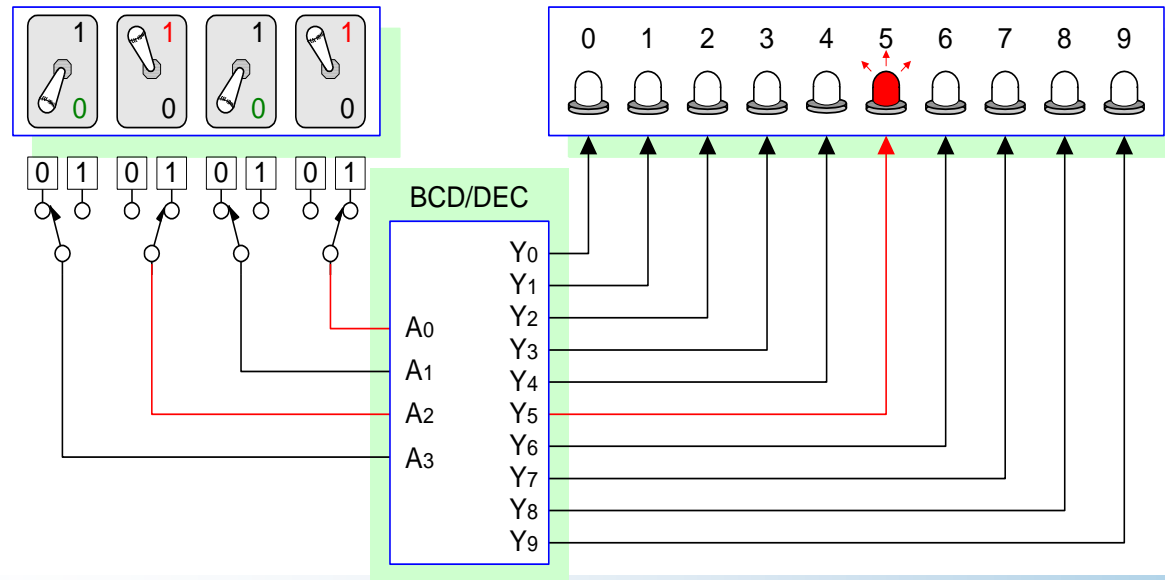
Decimal to BCD Encoder



4-to-1 Multiplexer



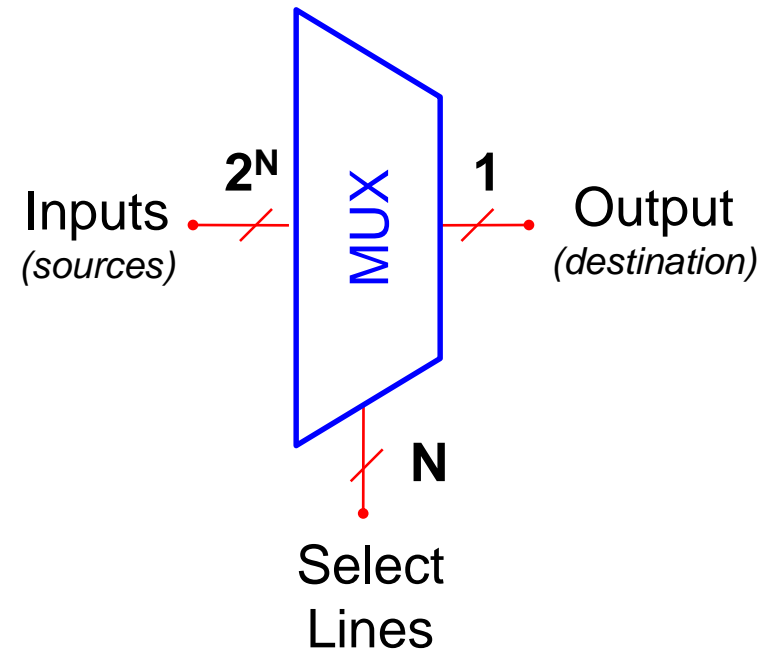
BCD to Decimal Decoder



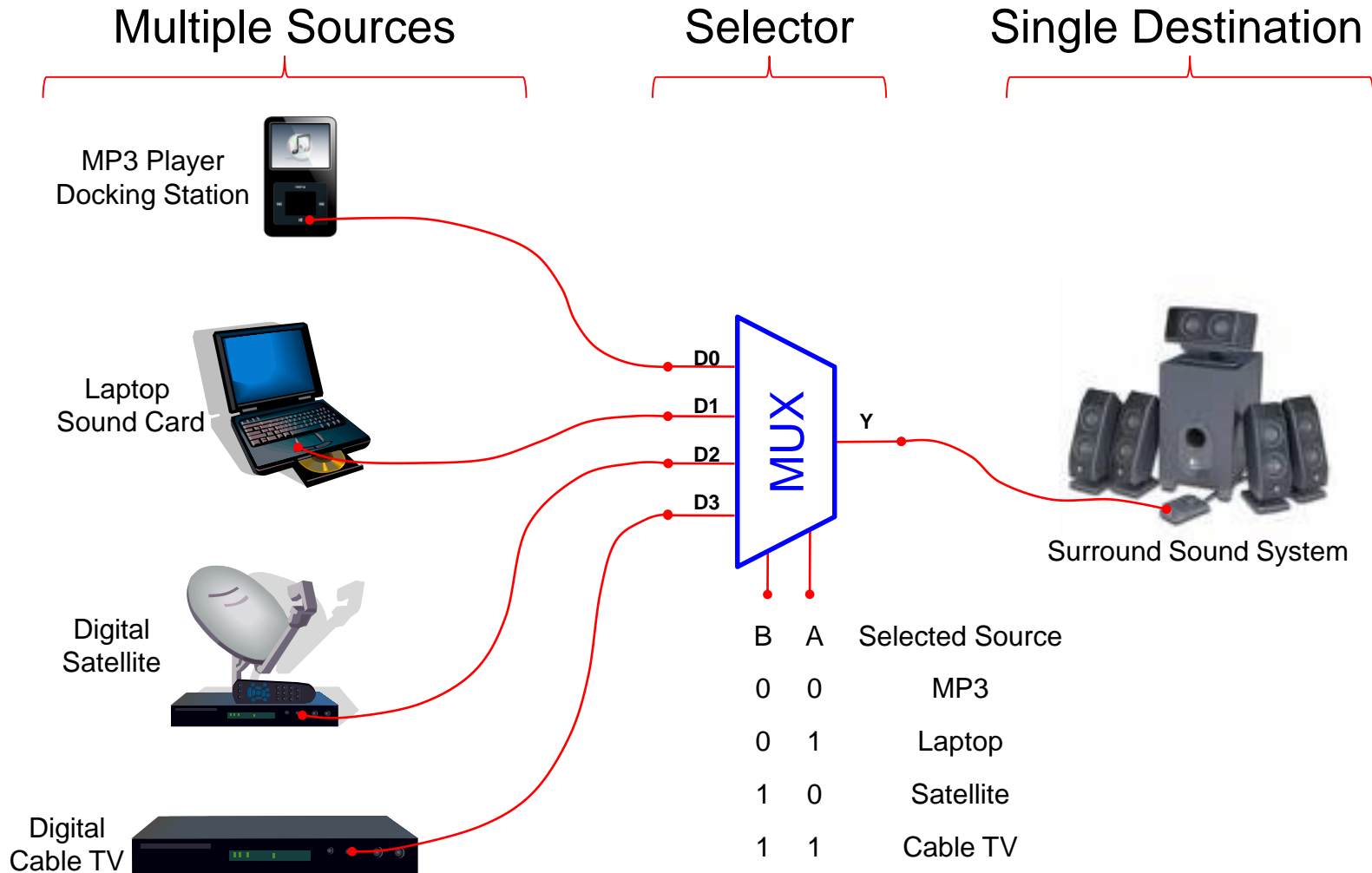
Multiplexer (MUX)

- A MUX has a
 - 2^N data inputs,
 - N control inputs
 - **One output**
- 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)

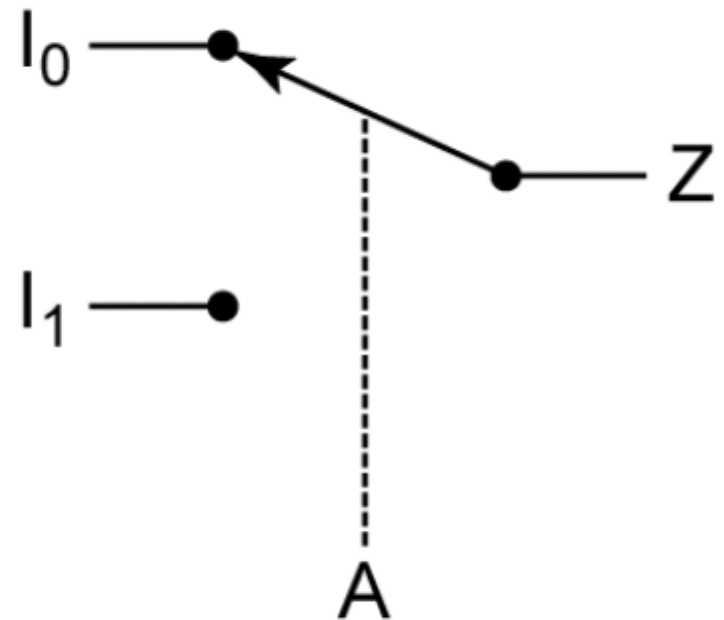
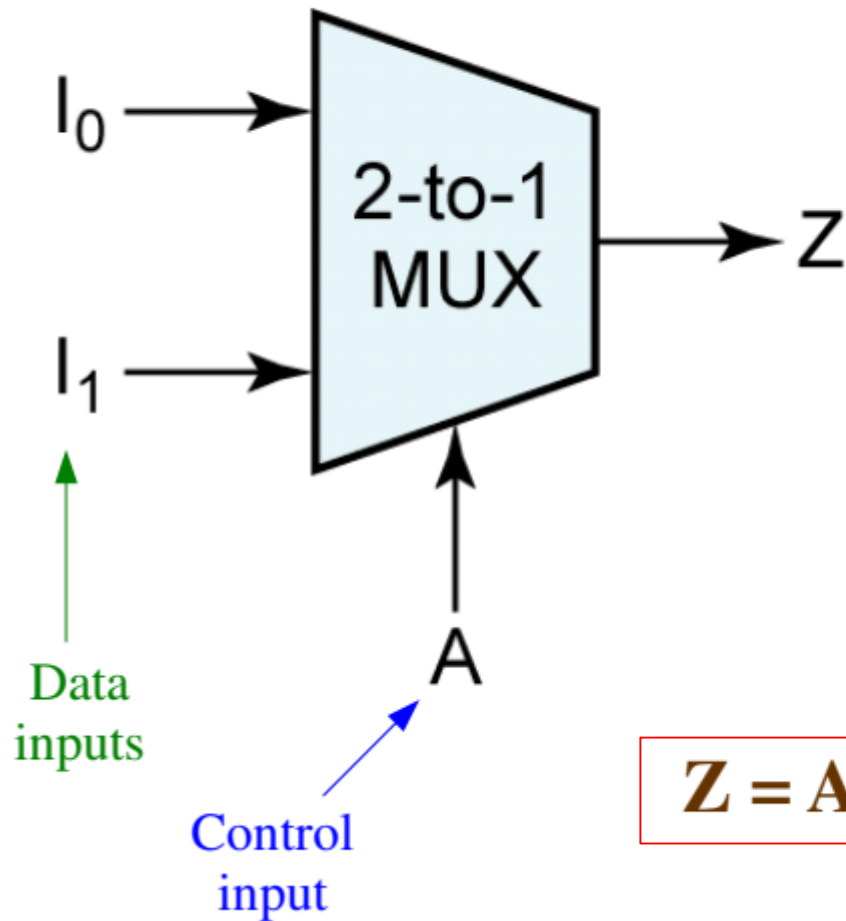
Multiplexer
Block Diagram



Typical Application of a MUX

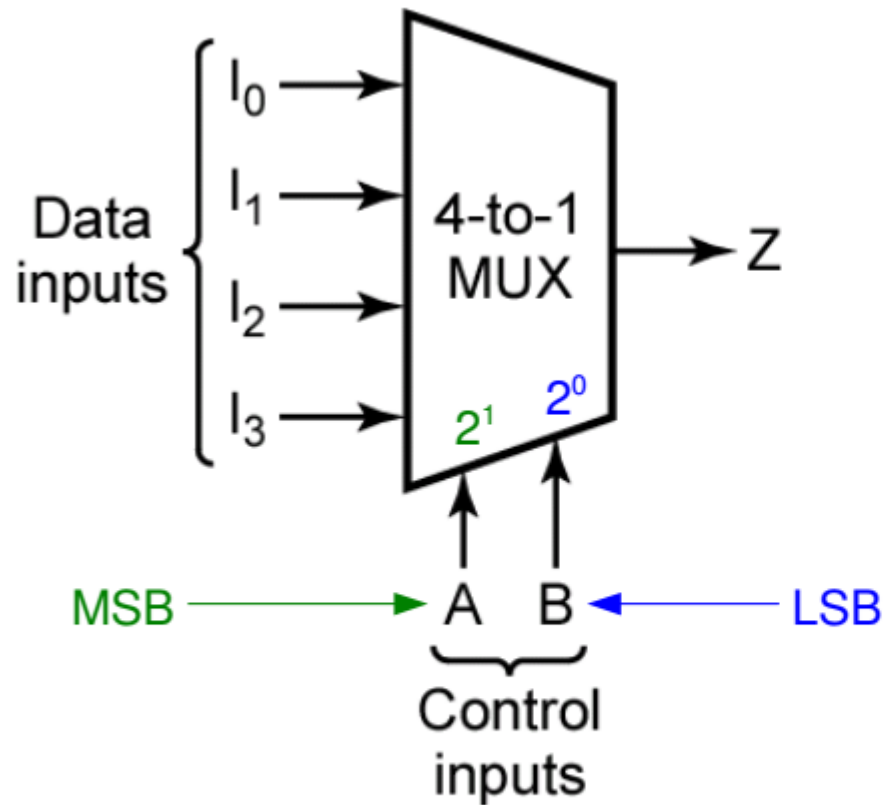


Multiplexer



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

Multiplexer



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

$$m_0 = A'.B'$$

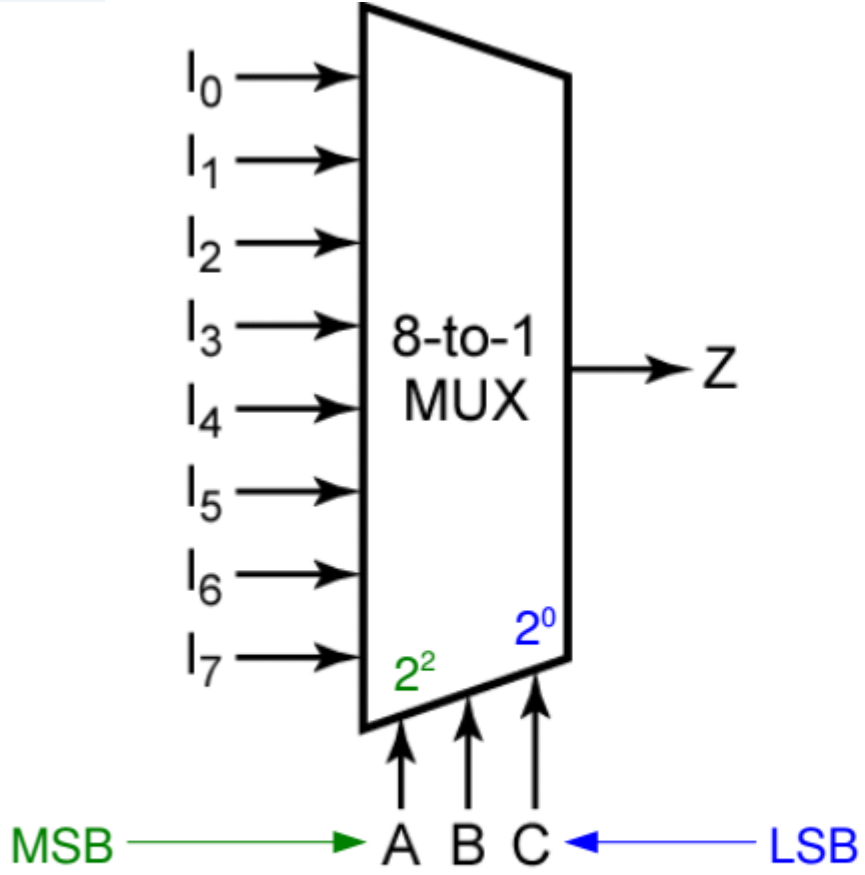
$$m_1 = A'.B$$

$$m_2 = A.B'$$

$$m_3 = A.B$$

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

Multiplexer

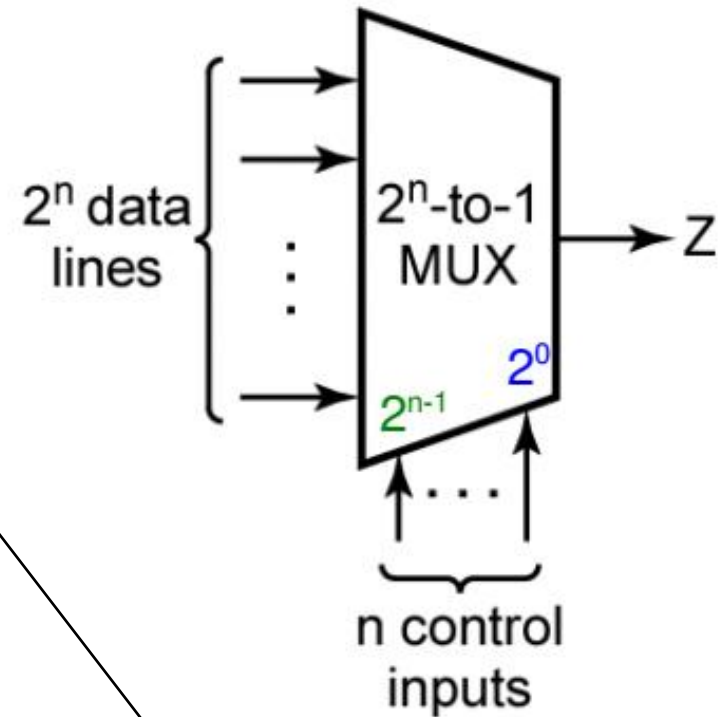


A	B	C	Z	
0	0	0	I_0	m_0
0	0	1	I_1	m_1
0	1	0	I_2	m_2
0	1	1	I_3	m_3
1	0	0	I_4	m_4
1	0	1	I_5	m_5
1	1	0	I_6	m_6
1	1	1	I_7	m_7

$$Z = A'.B'.C'.I_0 + A'.B'.C.I_1 + A'.B.C'.I_2 + A'.B.C.I_3 + A.B'.C'.I_4 + A.B'.C.I_5 + A.B.C'.I_6 + A.B.C.I_7$$

Logic Implementation using MUX

- A 2^n -to-1 MUX
 - Can be used to implement **ANY** function of **n variables**
 - Can be used to implement **SOME** functions of **$(n + 1)$ variables**



$$Z = \sum m_i \cdot I_i$$

Logic Implementation using MUX

- Example:

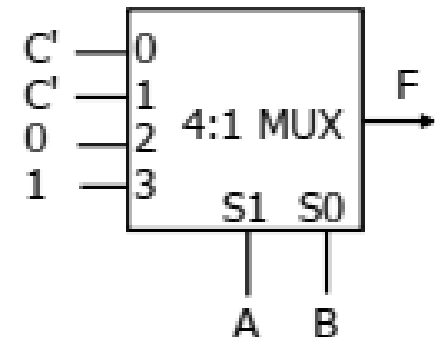
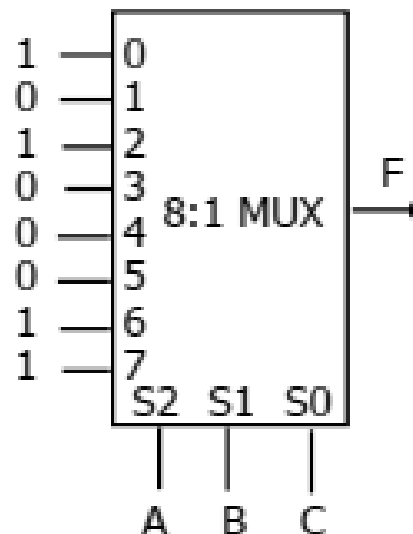
$$F(A, B, C) = \sum m(0, 2, 6, 7)$$

$$= m_0 + m_2 + m_6 + m_7$$

$$= A'B'C' + A'B'C + ABC' + ABC$$

$$= A'B'(C') + A'B'(C) + AB(0) + AB(1)$$

A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1



Logic Implementation using MUX

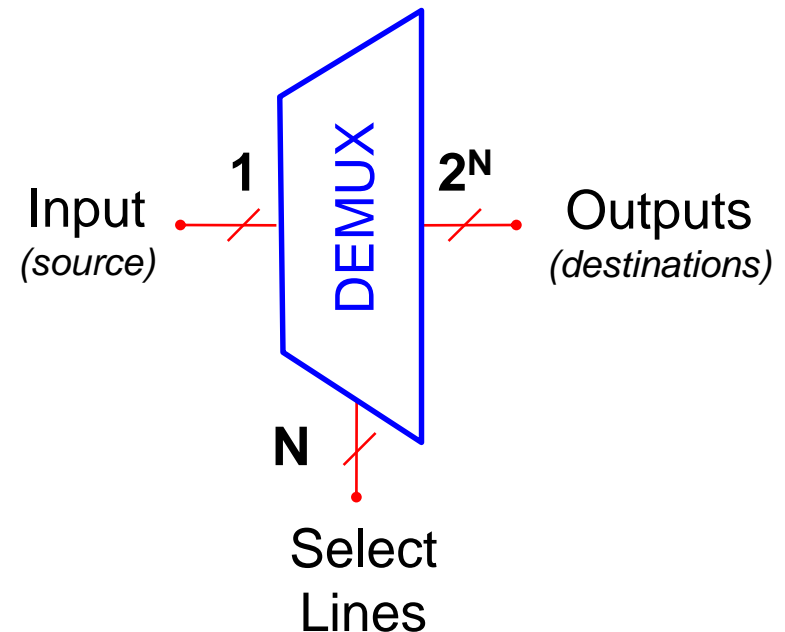
3-1. Implementing the following function as a 4-to-1 MUX

A	B	C	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Demultiplexer (DEMUX)

- A digital switch with
 - **One input (source)**
 - **N control inputs**
 - **2^N 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

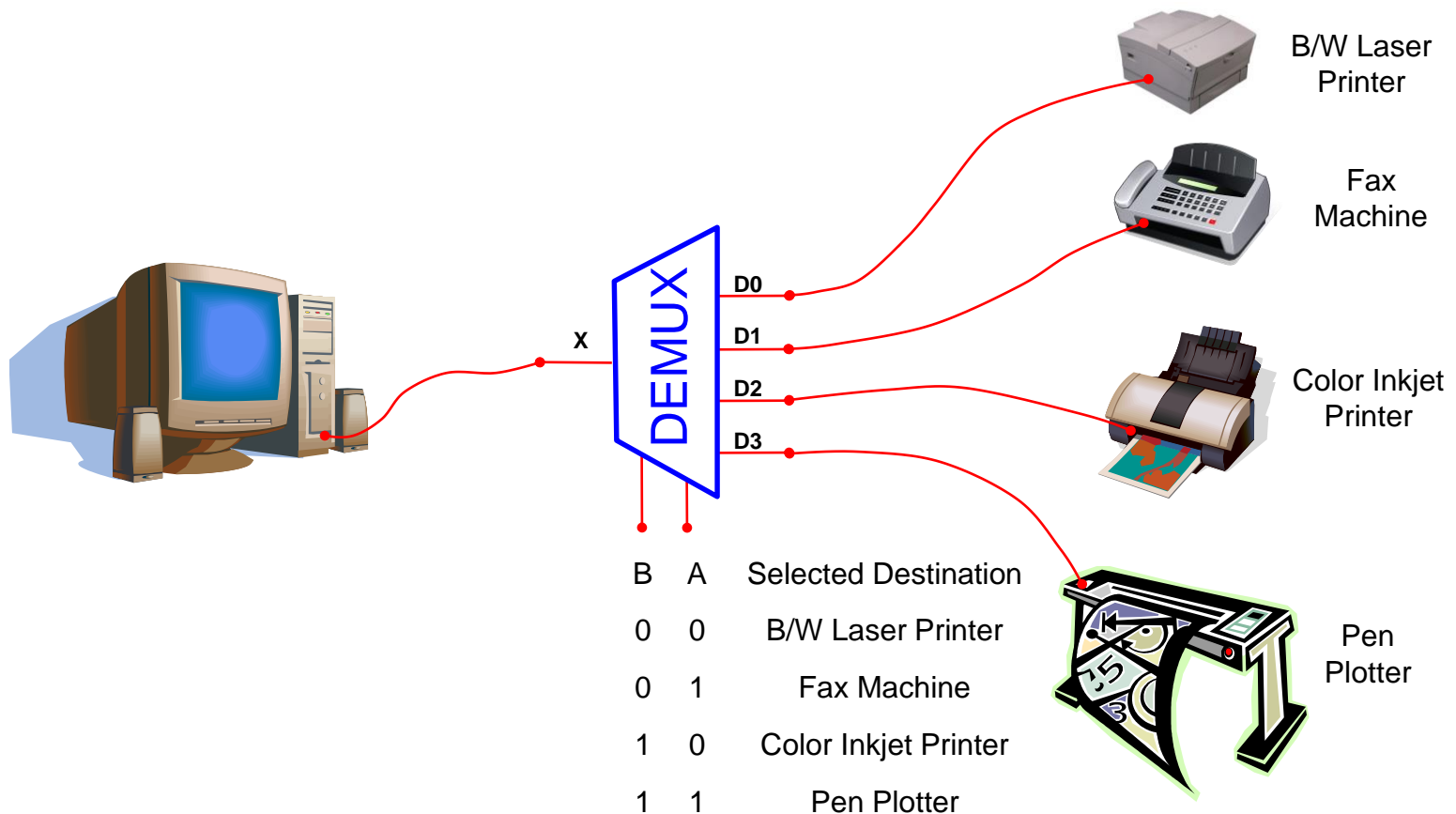


Typical Application of a DEMUX

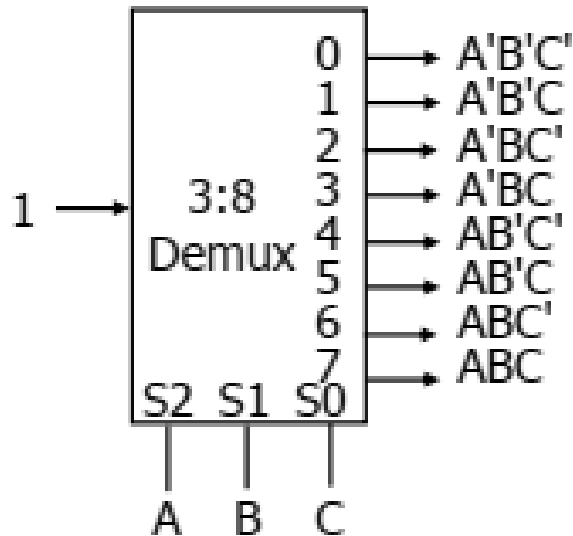
Single Source

Selector

Multiple Destinations



Logic Implementation using Demux



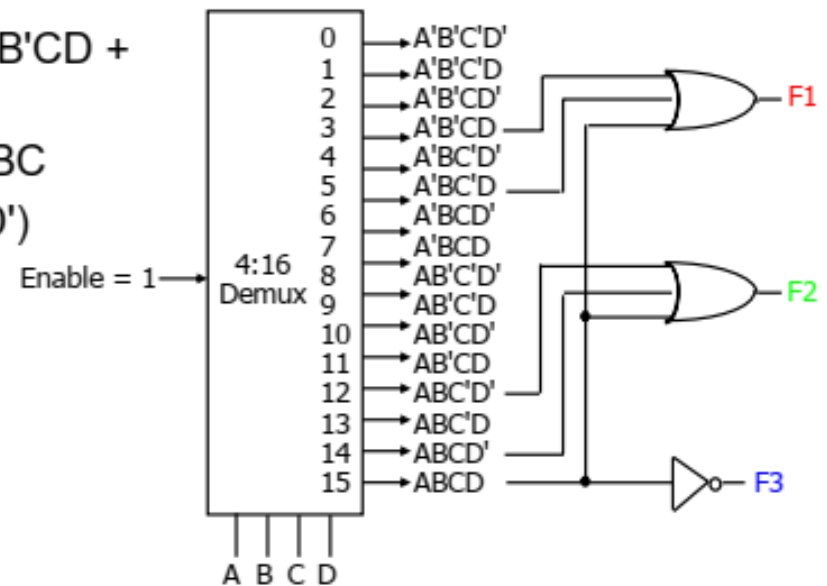
demultiplexer "decodes"
appropriate minterms
from the control signals

Example

$$F1 = A'BC'D + A'B'CD + ABCD$$

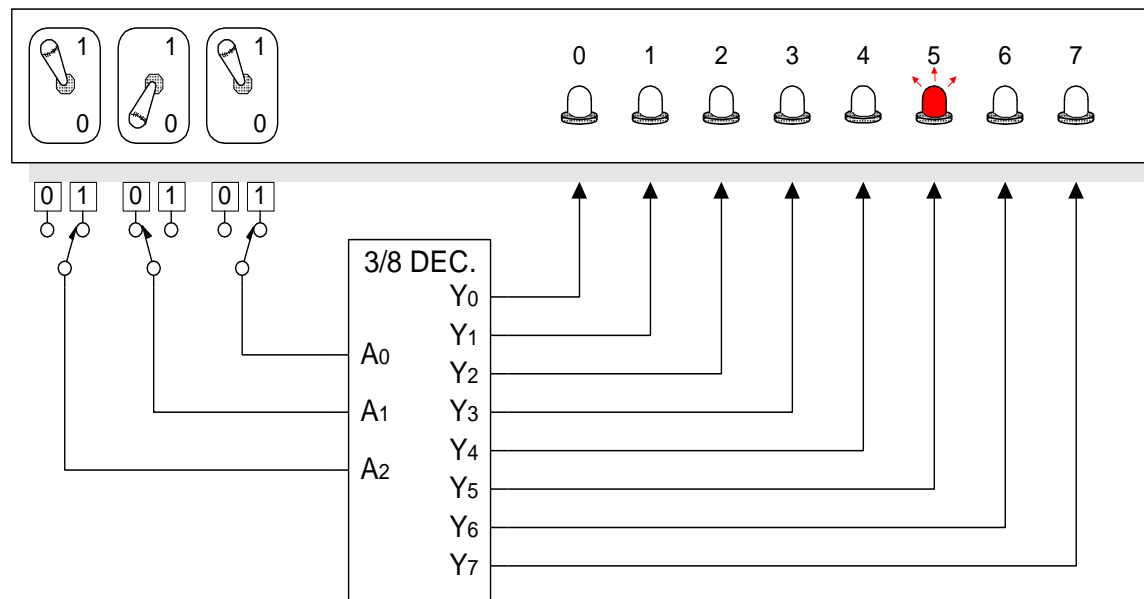
$$F2 = ABC'D' + ABC$$

$$F3 = (A'+B'+C'+D')$$

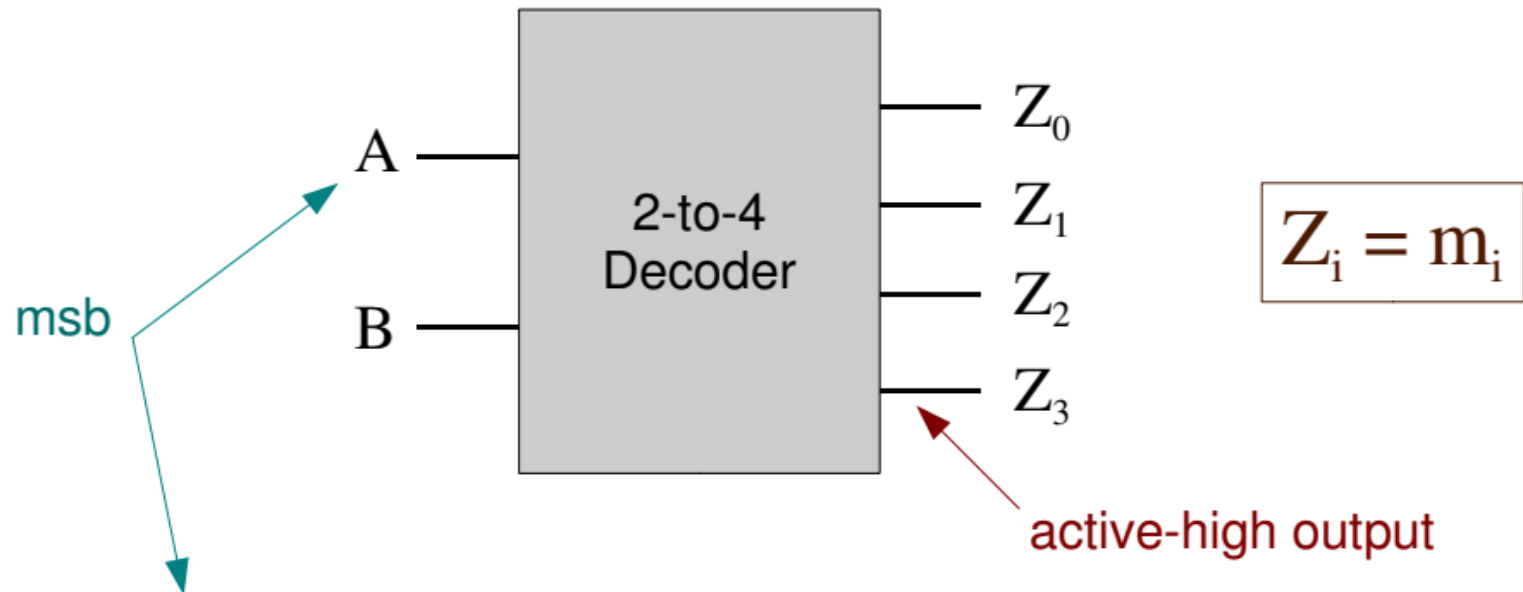


Decoders

- A decoder has
 - **N inputs**
 - **2^N outputs**
- **Only one of the outputs is enabled at a time.**
 - The output enabled is the one specified by the binary number formed at the inputs of the decoder.

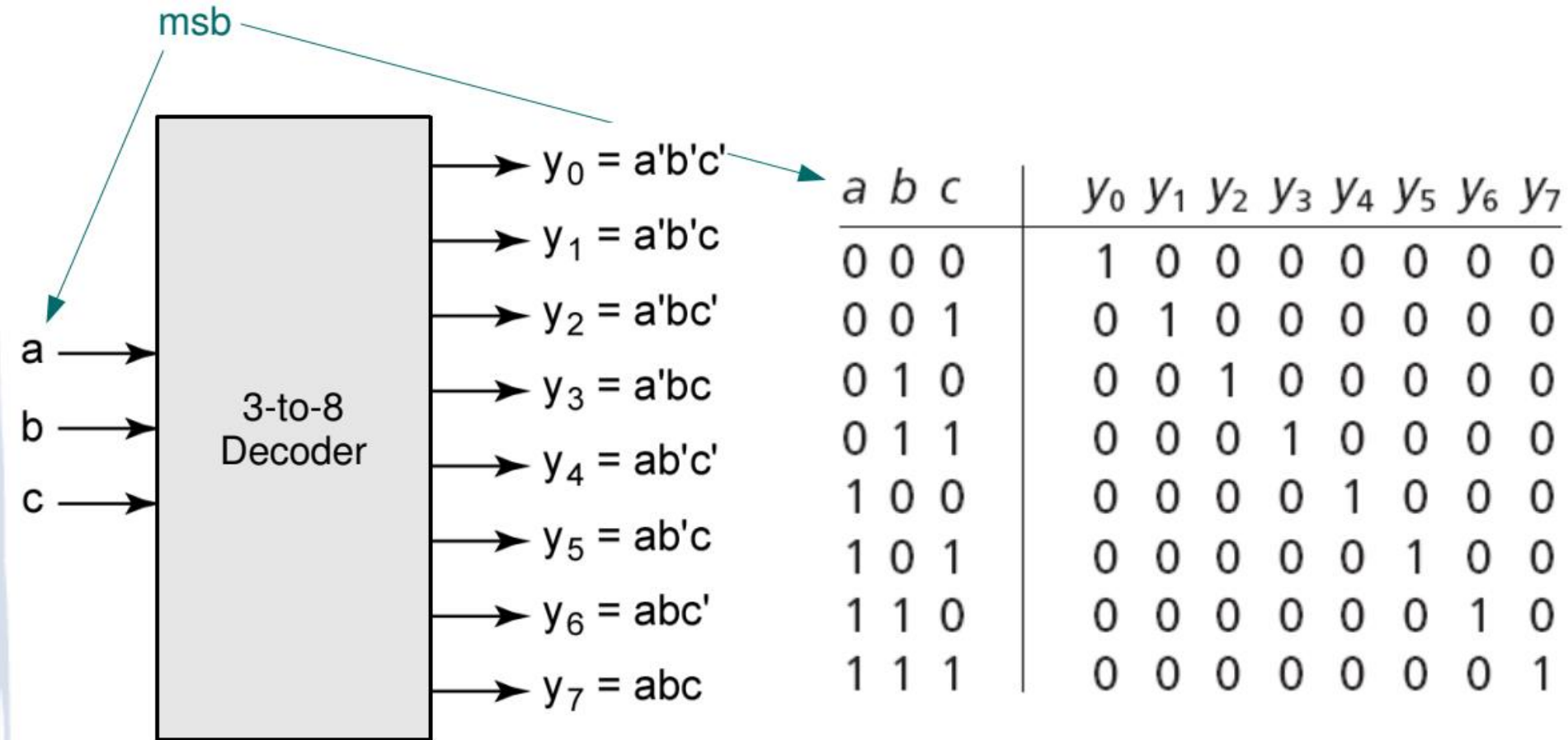


2 to 4 Line Decoder



A	B	Z_0	Z_1	Z_2	Z_3	
0	0	1	0	0	0	m_0
0	1	0	1	0	0	m_1
1	0	0	0	1	0	m_2
1	1	0	0	0	1	m_3

3 to 8 Line Decoder



Decoders

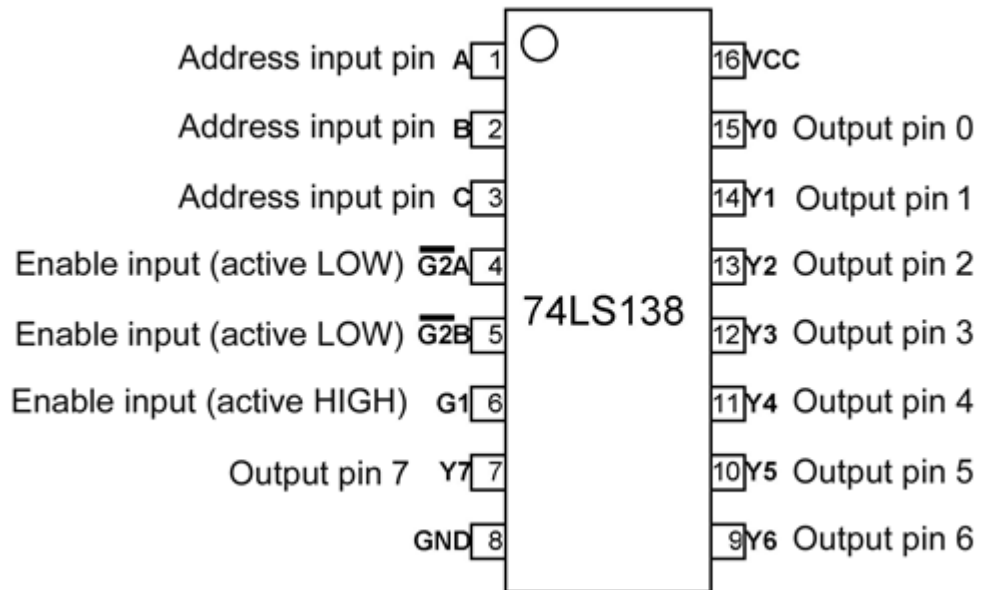
3-3. Which statement below best describes the function of a decoder?

- A. A decoder will convert a decimal number into the proper binary equivalent.
- B.** A decoder will convert a binary number into a specific output representing a particular character or digit.
- C. Decoders are used to prevent improper operation of digital systems.
- D. Decoders are special ICs that are used to make it possible for one brand of computer to talk to another.

Decoders

3-4. Output 5 of a 74138 octal decoder is selected when it is enabled by a data input of:

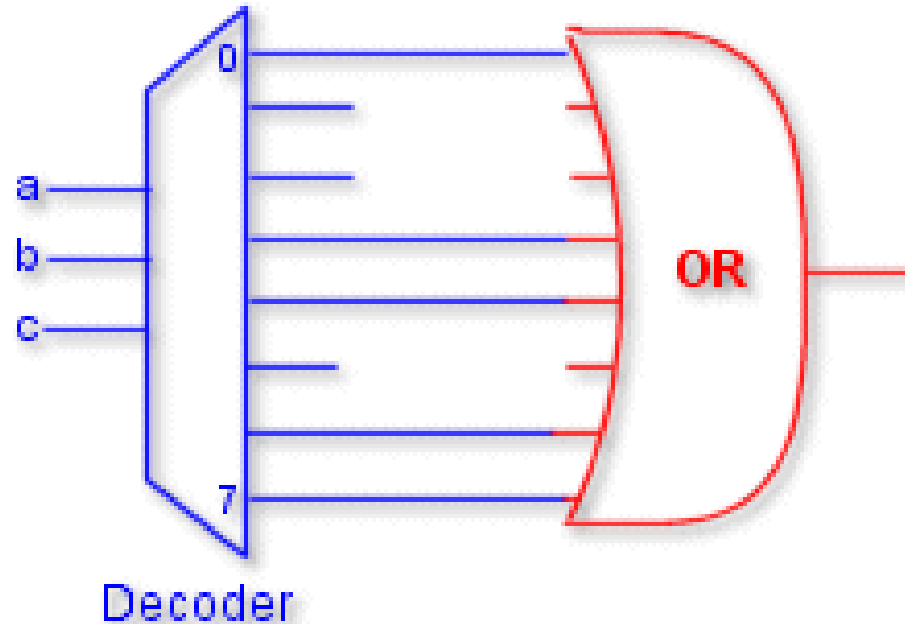
- A. $A_0 = 1, A_1 = 1, A_2 = 0$
- B. $A_0 = 1, A_1 = 0, A_2 = 1$**
- C. $A_0 = 0, A_1 = 1, A_2 = 0$
- D. $A_0 = 1, A_1 = 0, A_2 = 0$



Logic Implementation using Decoder

3-5. Implement following function with decoder:

$$f(a, b, c) = \sum m(0, 3, 4, 6, 7)$$



Implementing Logic
Using Decoders

Logic Implementation using Decoder

3-6. Using a 2-to-4 line decoder, design a logic circuit to realize the following Boolean function

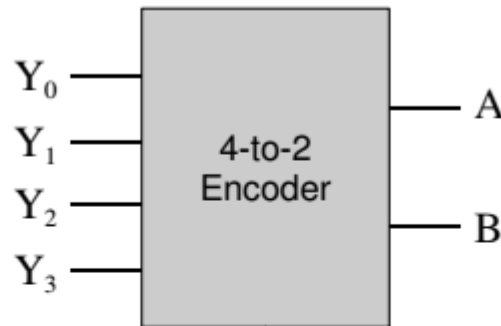
$$F(A, B, C) = \sum m(0, 1, 4, 6, 7)$$

3-7. Using a 3-to-8 line decoder, design a logic circuit to realize the following Boolean function

$$F(A, B, C) = \sum m(2, 3, 5, 6, 7)$$

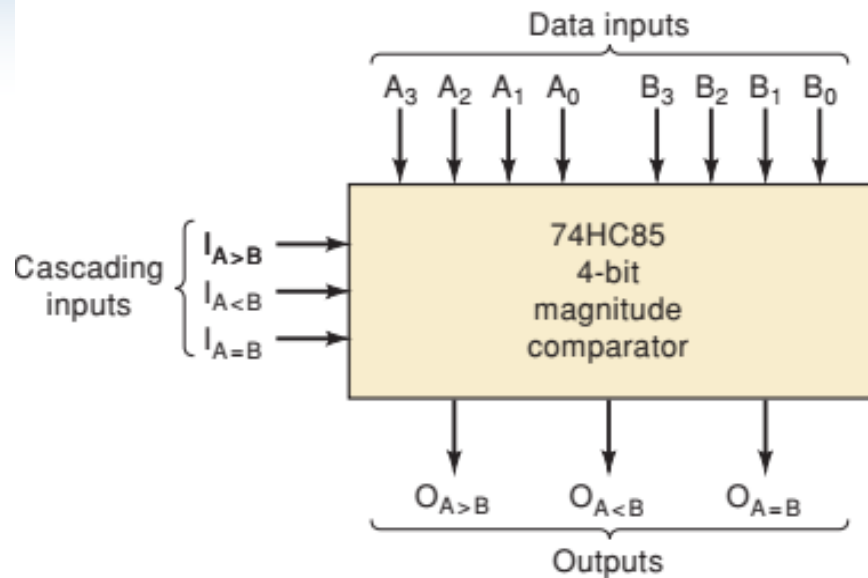
Encoders

- An encoder has
 - 2^N inputs
 - N outputs
- Performs the inverse operation of a decoder



Y_0	Y_1	Y_2	Y_3	A	B
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

Magnitude Comparator



- Compares two input binary quantities and generates outputs to indicate which one has the greater magnitude

COMPARING INPUTS				CASCADING INPUTS			OUTPUTS		
A ₃ , B ₃	A ₂ , B ₂	A ₁ , B ₁	A ₀ , B ₀	I _{A>B}	I _{A<B}	I _{A=B}	O _{A>B}	O _{A<B}	O _{A=B}
A ₃ >B ₃	X	X	X	X	X	X	H	L	L
A ₃ <B ₃	X	X	X	X	X	X	L	H	L
A ₃ =B ₃	A ₂ >B ₂	X	X	X	X	X	H	L	L
A ₃ =B ₃	A ₂ <B ₂	X	X	X	X	X	L	H	L
A ₃ =B ₃	A ₂ =B ₂	A ₁ >B ₁	X	X	X	X	H	L	L
A ₃ =B ₃	A ₂ =B ₂	A ₁ <B ₁	X	X	X	X	L	H	L
A ₃ =B ₃	A ₂ =B ₂	A ₁ =B ₁	A ₀ >B ₀	X	X	X	H	L	L
A ₃ =B ₃	A ₂ =B ₂	A ₁ =B ₁	A ₀ <B ₀	X	X	X	L	H	L
A ₃ =B ₃	A ₂ =B ₂	A ₁ =B ₁	A ₀ =B ₀	H	L	L	H	L	L
A ₃ =B ₃	A ₂ =B ₂	A ₁ =B ₁	A ₀ =B ₀	L	H	L	L	H	L
A ₃ =B ₃	A ₂ =B ₂	A ₁ =B ₁	A ₀ =B ₀	X	X	H	L	L	H
A ₃ =B ₃	A ₂ =B ₂	A ₁ =B ₁	A ₀ =B ₀	L	L	L	H	H	L
A ₃ =B ₃	A ₂ =B ₂	A ₁ =B ₁	A ₀ =B ₀	H	H	L	L	L	L

Magnitude Comparator

3-8. What are the outputs of a 7485 four-bit magnitude comparator when the inputs are $A = 1001$ and $B = 1010$?

- A. $A < B$ is 1; $A = B$ is 0; $A > B$ is 1
- B. $A < B$ is 0; $A = B$ is 1; $A > B$ is 0
- C. $A < B$ is 0; $A = B$ is 0; $A > B$ is 1
- D. $A < B$ is 1; $A = B$ is 0; $A > B$ is 0**

3-9. Which is the decimal number for the BCD number, 10110110 ?

- A. 182
- B. 36
- C. 116
- D. 10110110 is not a valid BCD number**