Digital System Design

Lecture 13

Combinational Logic Design

Binary Multipliers and Decoders

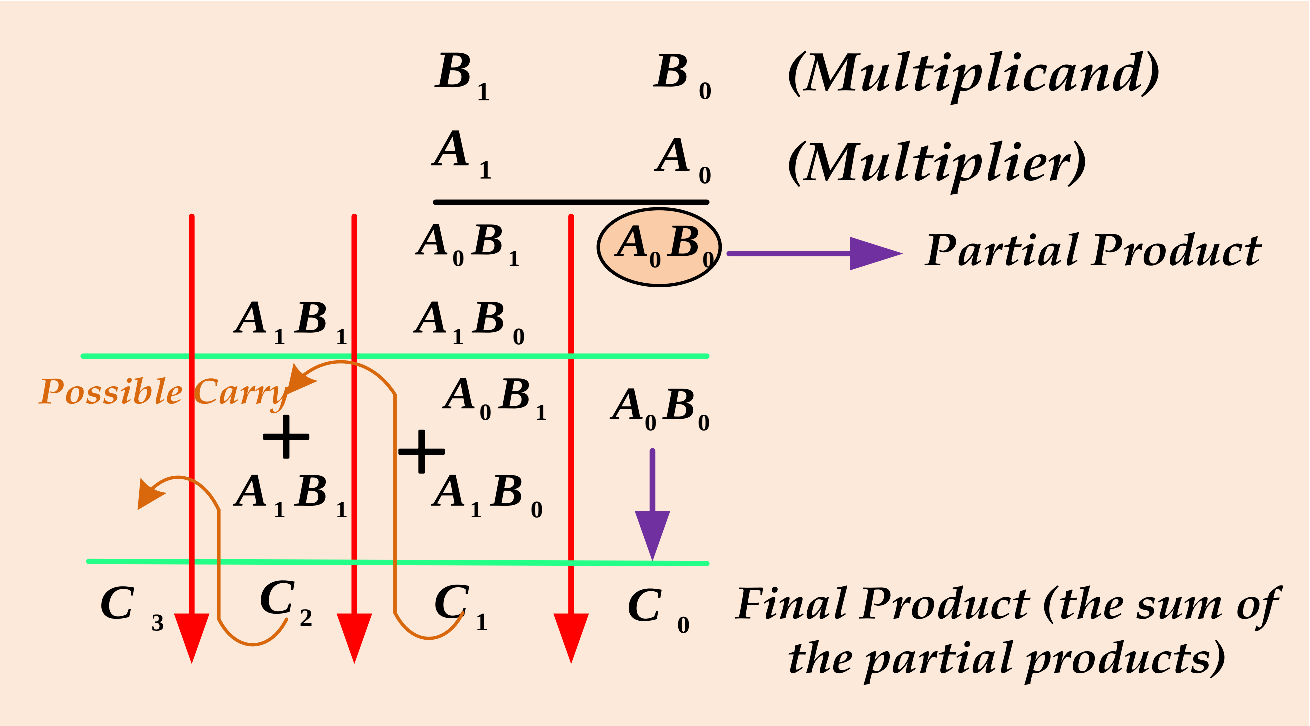
Objectives:

1. **Multipliers:**
   - **1.1 Multiplication of two 2-bit numbers.**
   - **1.2 Combinational circuit of binary multiplier with more bits.**
2. **Decoders:**
   - **2.1) 2x4 decoder (Active-high).**
   - **2.2) Active-low decoders.**
   - **2.3) Decoders with enable inputs.**
   - **2.4) Three-to-eight-line decoder circuit.**
   - **2.5) Larger decoder circuit.**
   - **2.6) Combinational logic implementation.**

1) Multipliers:

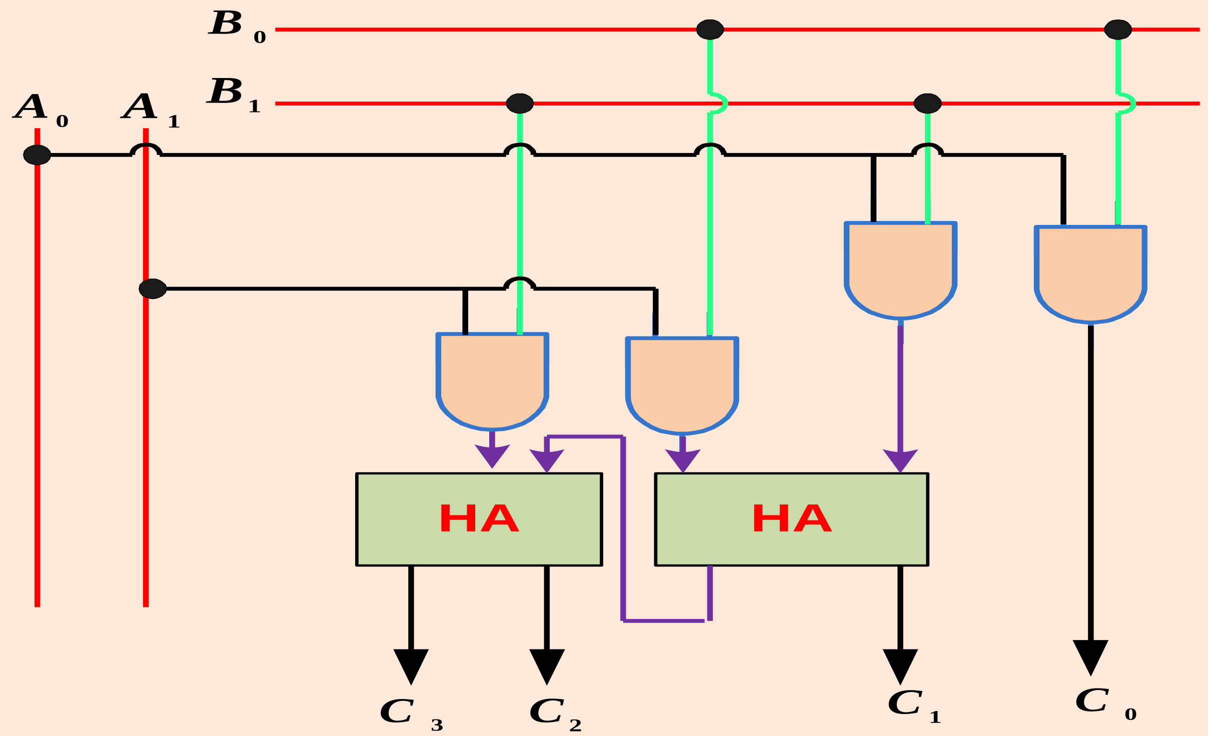
- Multiplication of binary numbers is performed in the same way as multiplication of decimal numbers:

1.1 Multiplication of two 2-bit numbers.



Combinational circuit

- The multiplication of two bits such as A_0 and B_0 produces a ***1*** if both bits are ***1***; otherwise, it produces a ***0***, this is identical to an ***AND*** operation.
- The two partial products are added with two half-adder (***HA***) circuits (if there are more than two bits, we must use full adder (***FA***)).



Two-bit by Two-bit binary multilplier

1.2 Combinational circuit of binary multiplier with more bits.

- For J bits ***multiplier*** and K bits ***multiplicand***, we need:

  $J \times K$ ***AND*** gates.

  $(J-1)K$-bits adders to produce a product of $J+K$ bits.

Example: - ***Create a logic circuit for***

$$\begin{array}{ccccccccc} & & & B_3 & B_2 & B_1 & B_0 & \times & \\ & & & & A_2 & A_1 & A_0 & & \\ \hline C_6 & C_5 & C_4 & C_3 & C_2 & C_1 & C_0 & & \textit{Answer} \end{array}$$

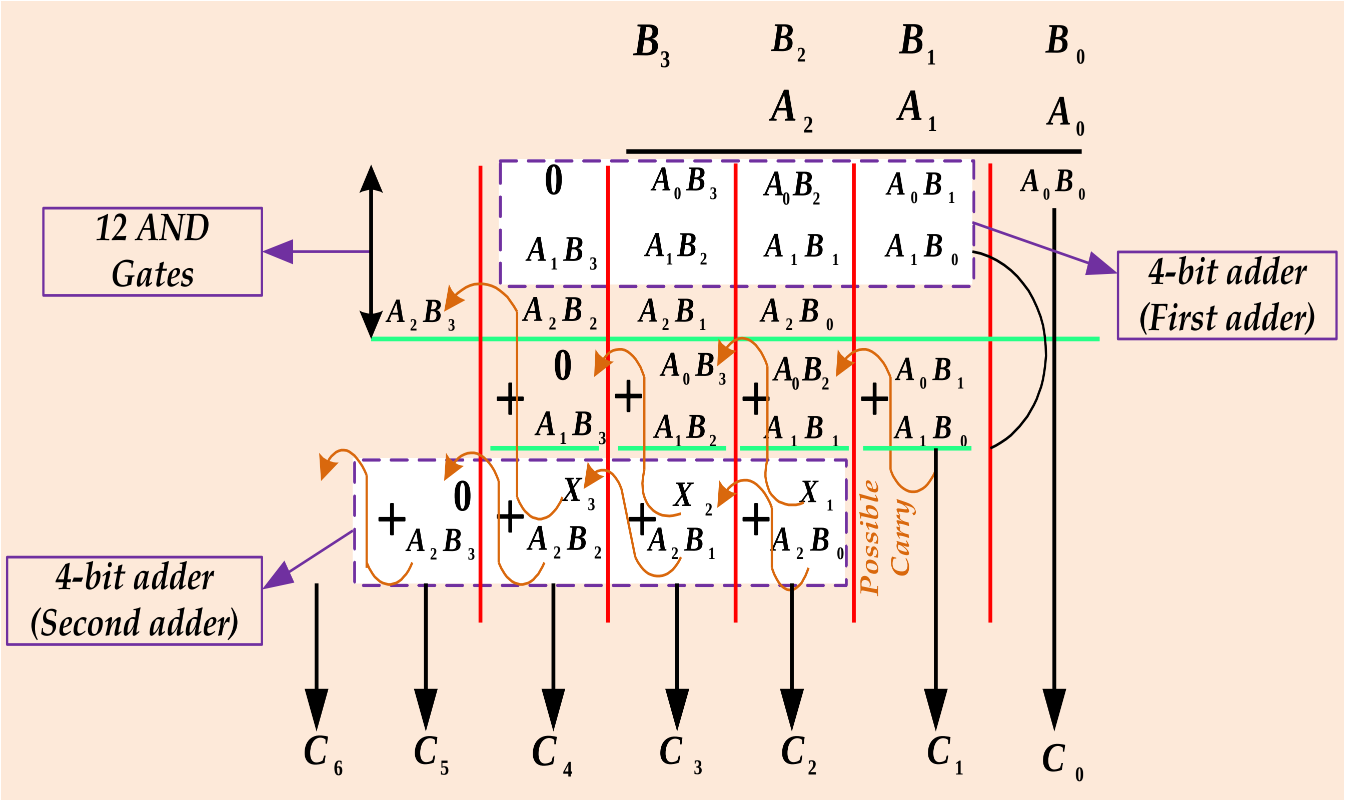
Solution:

$$K = 4$$

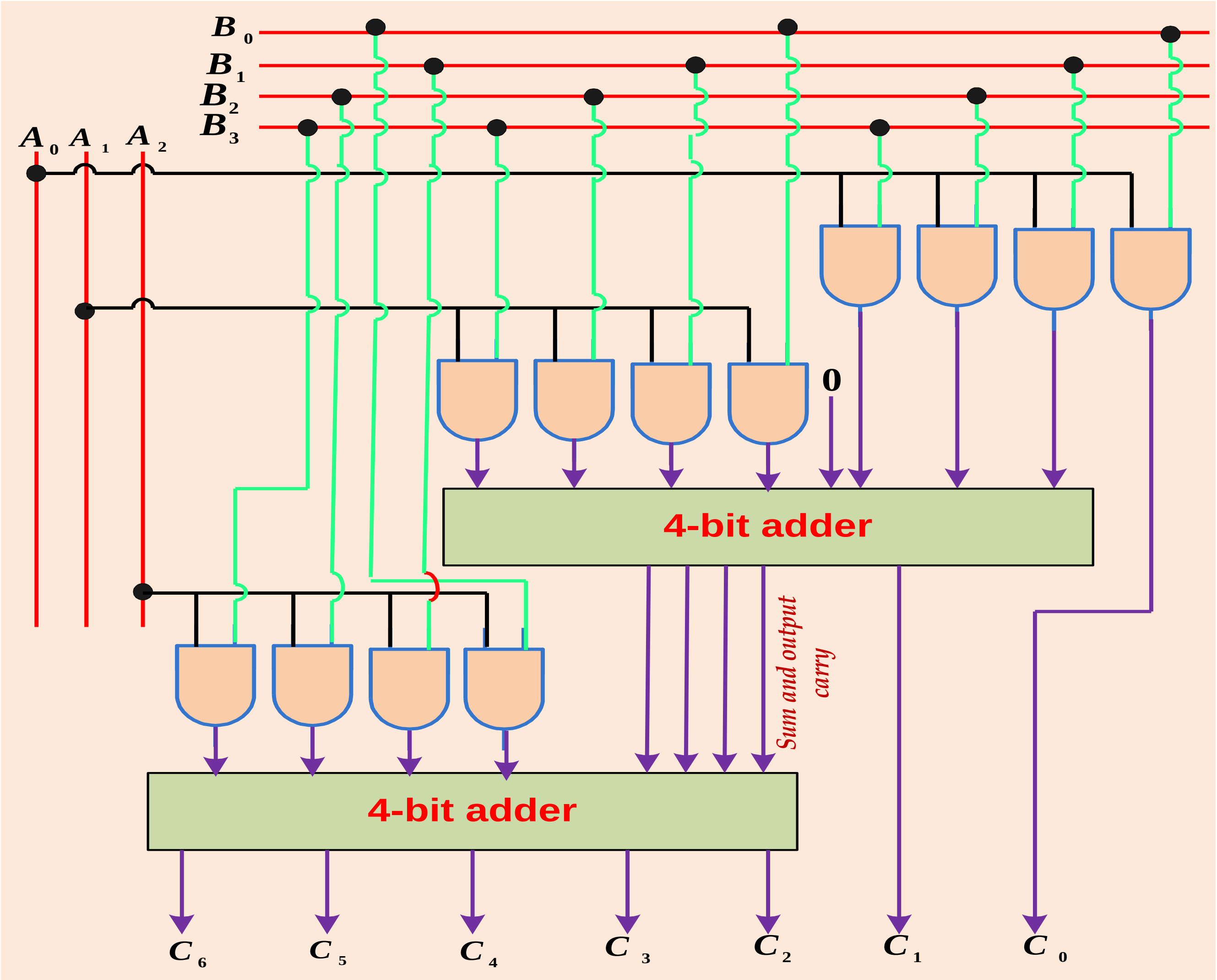
$$J = 3$$

So:

We need ***12 AND*** gates and ***2 four-bit adders*** to produce a product of ***seven bits***: $C_6C_5C_4C_3C_2C_1C_0$.


B_3 B_2 B_1 B_0
A_2 A_1 A_0
0 A_0B_3 A_0B_2 A_0B_1 A_0B_0
A_1B_3 A_1B_2 A_1B_1 A_1B_0
A_2B_3 A_2B_2 A_2B_1 A_2B_0
12 AND Gates
4-bit adder (First adder)
0 A_0B_3 A_0B_2 A_0B_1
+ + + +
A_1B_3 A_1B_2 A_1B_1 A_1B_0
0 X_3 X_2 X_1
+ + + +
A_2B_3 A_2B_2 A_2B_1 A_2B_0
4-bit adder (Second adder)
Possible Carry
C_6 C_5 C_4 C_3 C_2 C_1 C_0

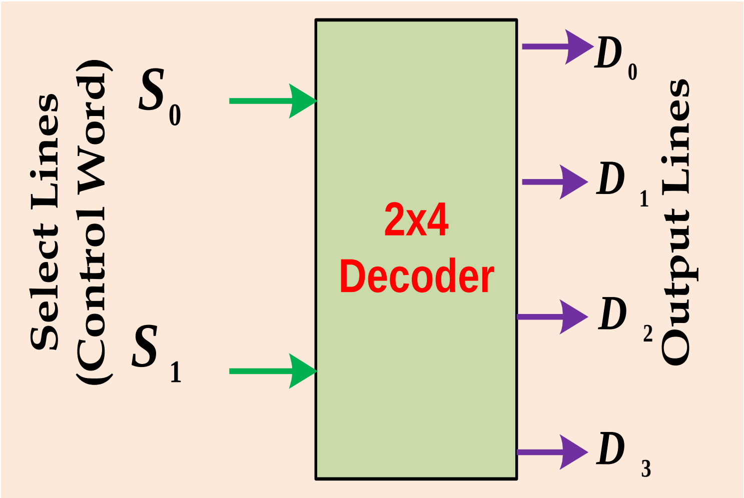
Circuit Implementation:



2) Decoders

- Information is represented in digital system by binary codes. A binary code of n bits is capable of representing up to 2^n distinct elements of coded information.
- A ***decoder*** is a combinational circuit that covers binary information from n input lines to a maximum of 2^n unique output lines.
- The decoders are called $n-to-m$ line decoders, where $m \leq 2^n$(for example ***BCD-to-seven-segment decoder***, $3-to-8$ decoder).
- The purpose of the decoders is to ***generate*** the 2^n (or fewer) minterms of n input variables.
- Decoder is a circuit that allows us to ***activate an output line by specifying a control word***; it is either active-high or active-low.
    - **Active-high** sets a particular output to logic ***1***. While remaining other outputs to logic ***0***.
    - **Active-low** sets a particular output to logic ***0***, other outputs to logic ***1***.

2.1) 2x4 decoder (Active-high)

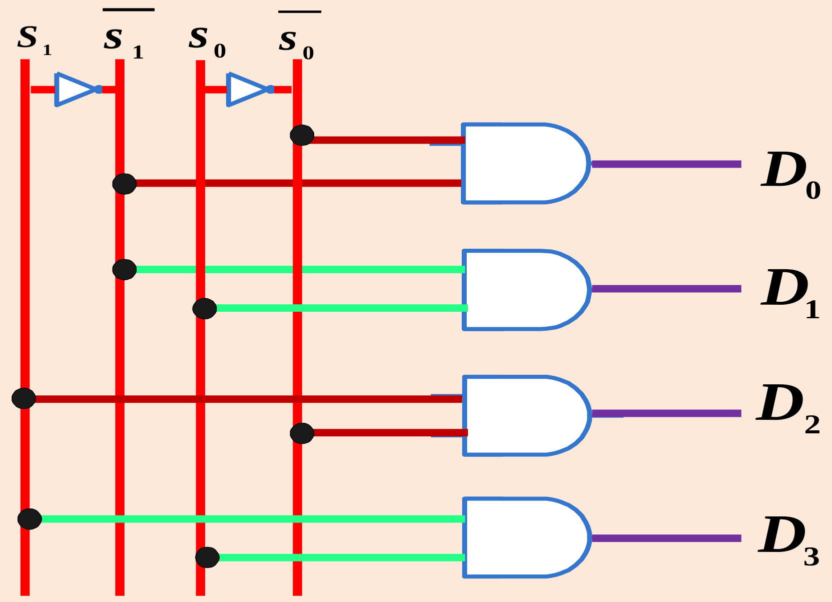


1. $S_1 = 0,\ S_0 = 0 \Longrightarrow D_0 = 1$
   $D_0 = \overline{S_1} \cdot \overline{S_0};\ D_1 = D_2 = D_3 = 0$
2. $S_1 = 0,\ S_0 = 1 \Longrightarrow D_1 = 1$
   $D_1 = \overline{S_1} \cdot S_0;\ D_0 = D_2 = D_3 = 0$
3. $S_1 = 1,\ S_0 = 0 \Longrightarrow D_2 = 1$
   $D_2 = S_1 \cdot \overline{S_0};\ D_0 = D_1 = D_3 = 0$
4. $S_1 = 1,\ S_0 = 1 \Longrightarrow D_3 = 1$
   $D_3 = S_1 \cdot S_0;\ D_0 = D_1 = D_2 = 0$

Inputs **Select Lines**		**Output** **Output Lines**			
S_1	S_0	D_3	D_2	D_1	D_0
0	*0*	*0*	*0*	*0*	*1*
0	*1*	*0*	*0*	*1*	*0*
1	*0*	*0*	*1*	*0*	*0*
1	*1*	*1*	*0*	*0*	*0*

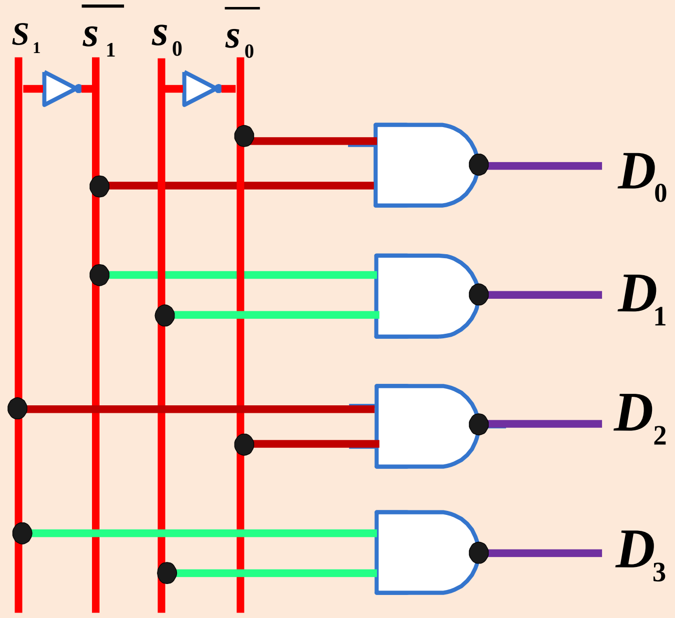
Truth table

- From truth table:
    - ✓ **Two inputs are decoded to 4 outputs.**
    - ✓ **Each output represents one of the minterms of the three input variables.**
    - ✓ **For each possible combination (input), there are 3 outputs that are equal to 0 and only one that is equal to 1.**



Decoder Logic Circuit

2.2) Active-low decoders



Active-Low Decoder Logic Circuit

Inputs Select Lines		Output Output Lines			
S_1	S_0	D_3	D_2	D_1	D_0
0	*0*	*1*	*1*	*1*	*0*
0	*1*	*1*	*1*	*0*	*1*
1	*0*	*1*	*0*	*1*	*1*
1	*1*	*0*	*1*	*1*	*1*

Truth table

AND becomes NAND

$$D_0 = \overline{\overline{S_{1.}}\,\overline{S_0}} \quad \textbf{\textit{Complement}}$$

$$= S_1 + S_0$$

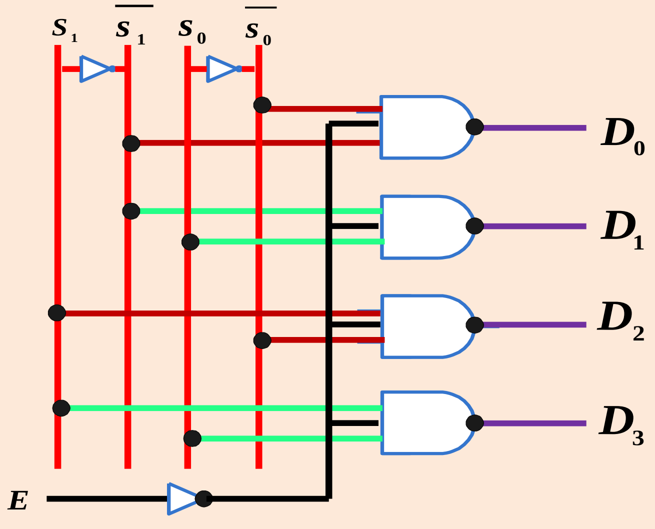
$D_1 = \overline{\overline{S_{1.}}\,S_0} = S_1 + \overline{S_0}$;
$D_2 = \overline{S_1} + S_0$;
$D_3 = \overline{S_1} + \overline{S_0}$

2.3) Decoders with enable inputs

- Some decoders include one or more ***enable inputs*** to control the circuit operation.
- A two-to-four line decoder with an ***enable input***_constructed with ***NAND gate*** is the following:

Inputs			Output Output Lines			
E	S_1	S_0	D_3	D_2	D_1	D_0
1	***X***	***X***	*1*	*1*	*1*	*1*
0	*0*	*0*	*1*	*1*	*1*	*0*
0	*0*	*1*	*1*	*1*	*0*	*1*
0	*1*	*0*	*1*	*0*	*1*	*1*
0	*1*	*1*	*0*	*1*	*1*	*1*

Truth table

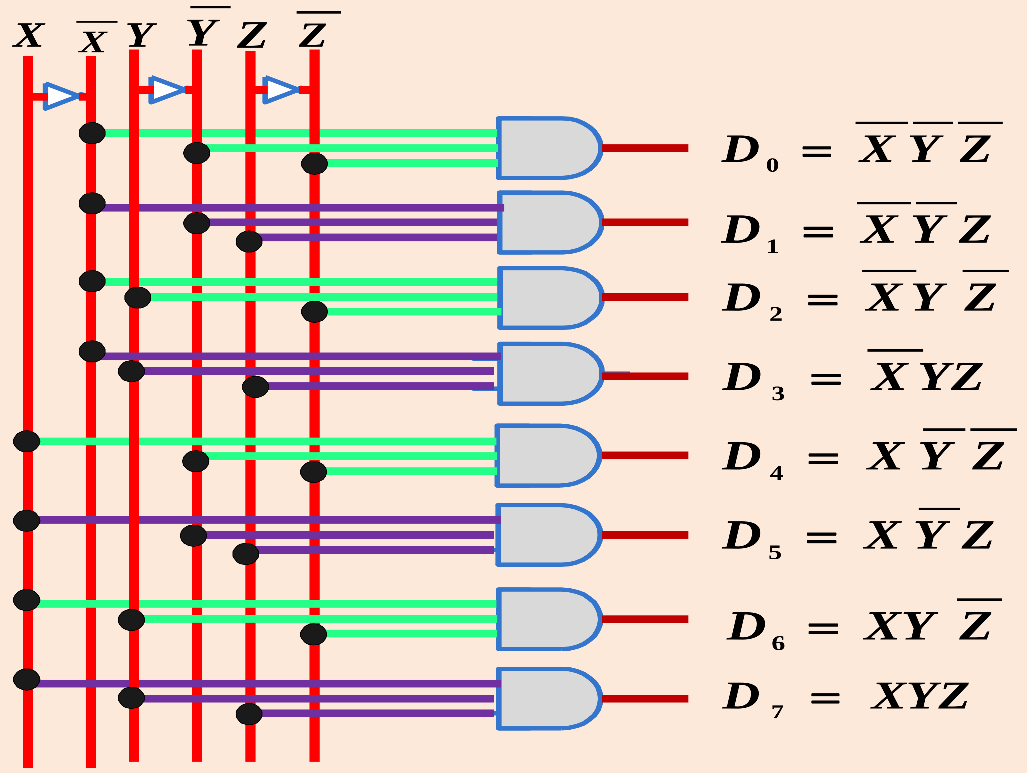


Active-Low Decoder Logic Circuit with enable input

- The circuit operates with ***complemented output and a complemented enable input*** (***Active-Low Enable***):
  - ✓ $E = 0$ then the decoder is enabled.
  - ✓ $E = 1$ then the decoder is disabled.
- The enable input may be active with ***0*** or with a ***1*** signal.

2.4) Three-to-eight-line decoder circuit

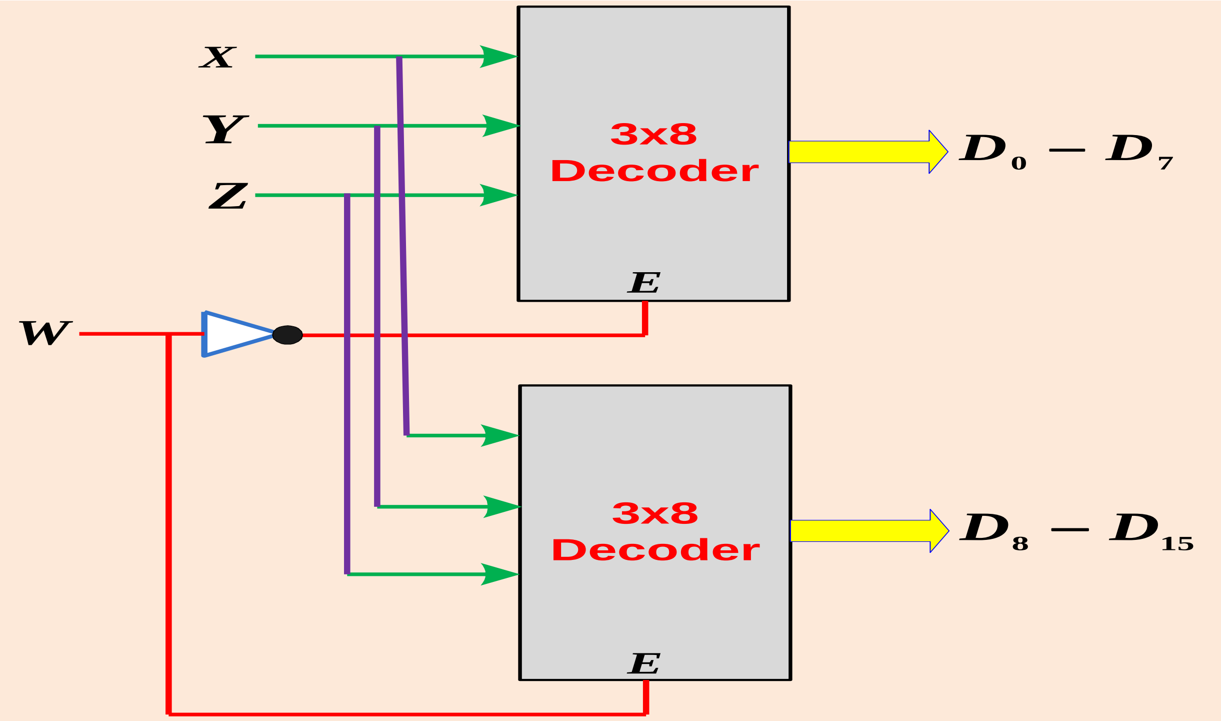
inputs			***outputs***							
X	Y	Z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	*0*	*0*	*1*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
0	*0*	*1*	*0*	*1*	*0*	*0*	*0*	*0*	*0*	*0*
0	*1*	*0*	*0*	*0*	*1*	*0*	*0*	*0*	*0*	*0*
0	*1*	*1*	*0*	*0*	*0*	*1*	*0*	*0*	*0*	*0*
1	*0*	*0*	*0*	*0*	*0*	*0*	*1*	*0*	*0*	*0*
1	*0*	*1*	*0*	*0*	*0*	*0*	*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*


X
$\overline{X}$
Y
$\overline{Y}$
Z
$\overline{Z}$
$D_0 = \overline{X}\,\overline{Y}\,\overline{Z}$
$D_1 = \overline{X}\,\overline{Y}\,Z$
$D_2 = \overline{X}\,Y\,\overline{Z}$
$D_3 = \overline{X}\,YZ$
$D_4 = X\,\overline{Y}\,\overline{Z}$
$D_5 = X\,\overline{Y}\,Z$
$D_6 = XY\,\overline{Z}$
$D_7 = XYZ$

Decoder Logic Diagram

2.5) Larger decoder circuit.

- ➢ Decoders with enable inputs can be connected together to form a larger decoder circuit.
- ➢ To design a $4-to-16$ line decoder. Using two $3-to-8$ decoders with enable inputs, we do the following connection:
  - ✓ When $W = 0$ the top decoder is enable and the other is disable: The bottom decoder outputs are all $0's$ and the top eight outputs generate minterms 0000 to 0111.
  - ✓ When $W = 1$ the enable condition is reversed, the bottom decoder outputs generate minterms 1000 to 1111.



2.6) Combinational logic implementation:

- Since, a decoder provides the 2^n minterms of n input, and any Boolean function can be expressed in sum-of-minterms form, A decoder that generates the minterms of the functions, together with an ***external OR gate*** that forms their logical sum, provides a hardware implementation of the function.
- In this way, any combinational circuit with n inputs and m outputs can be implemented using $n - to - 2^n$ ***line decoder*** and m ***OR gates***.
- The ***procedure*** for implementing a combinational circuit by means of ***decoders and OR gates*** requires that the Boolean function for the circuit be ***expressed as a sum of minterms.***

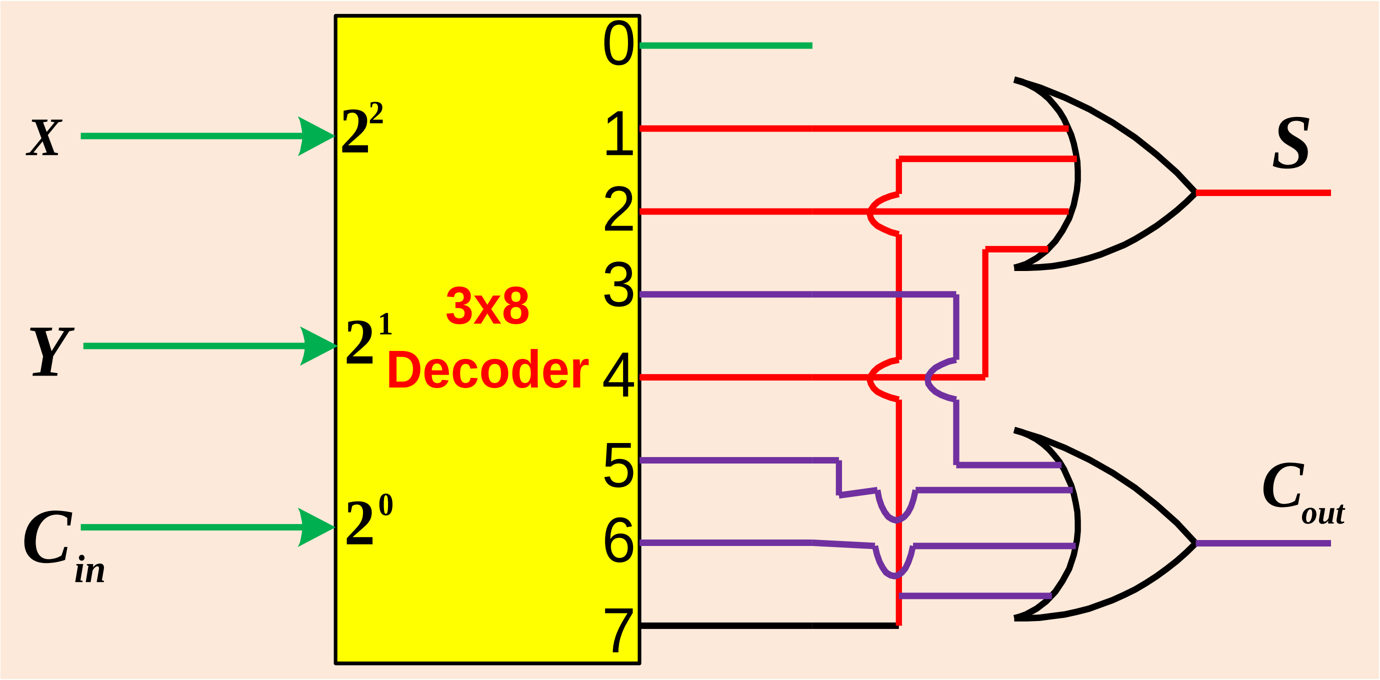
*Example:-**Implementation a full-adder circuit using the decoders***.

- From the ***truth table*** of the full adder, we obtain the functions for the ***sum*** and ***carry*** in ***sum-of-minterms*** form:

$$S(X, Y,\ C_{in}) = \sum(1, 2, 4, 7)$$

$$C_{out}\ (X, Y, C_{in}) = \sum(3, 5, 6, 7)$$

- We have ***3 inputs***: so, we need a ***three-to-eight line decoder***.


X
Y
C_in
2^2
2^1
2^0
3x8
Decoder
0
1
2
3
4
5
6
7
S
C_out