Digital Logic Design (CSNB163)

Module 11

Recapss..

- In Module 9, we have been introduced to the concept of combinational logic circuits through the examples of binary adders.
- Meanwhile, in Module 10, we have learnt about two more examples of combinational logic circuits:
  - **Binary multiplier**
  - **Magnitude comparator**
- In this module, we shall study about two more examples of combinational logic circuits:
  - **Binary decoder**
  - **Binary encoder**

Binary Decoder

- A binary decoder is a combinational circuit that converts **binary information** from ***n* input lines** to a maximum of **2^n unique output lines**.
- For example, a decoder that converts binary representation in:
  - 2 bits into $2^2 = 4$,
  - 3 bits into $2^3 = 8$,
  - 4 bits into $2^4 = 16$ unique output lines
- If the *n*-bit coded information **has unused combinations**, the decoder may have **fewer than 2^n outputs**. For example, a BCD decoder of 4 bits input, only produces 10 outputs.

3 to 8 Binary Decoder

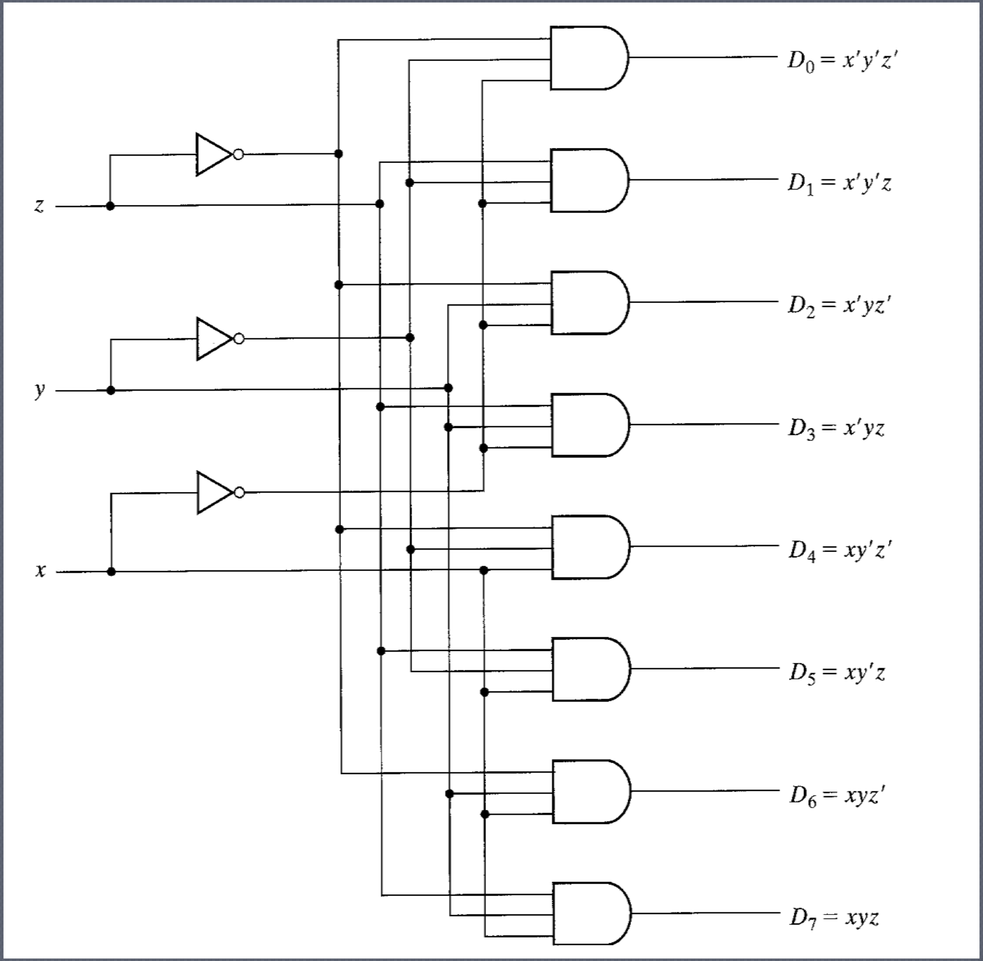
- The truth table for a 3-to-8 decoder whereby:
  - 3 bit inputs (x, y, z) are decoded into
  - 8 outputs (D0, D1, D2, D3, D4, D5, D6, D7}
  - where each output representing one of the minterm.

x	y	z	D0	D1	D2	D3	D4	D5	D6	D7
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**

$D0 = x'.y'.z'$

3 to 8 Binary Decoder (cont.)

- The logic circuit diagram:
  - For each possible input combination, 7 outputs are equal to 0 and only one equal to 1.
  - The output whose value is equal to 1 represents the equivalent minterm.



3 to 8 Binary Decoder (cont.)

- In the previous examples, the outputs are not complemented, whereby AND gates are used.
- We can also use complemented outputs for the 3 to 8 decoder, as such NAND gates can be used.

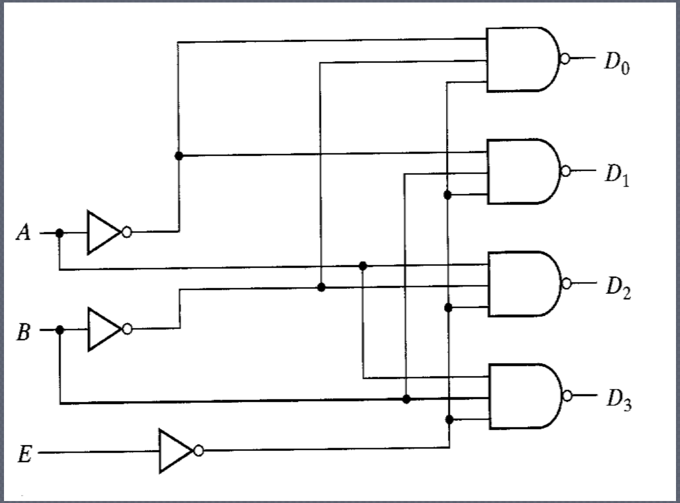
x	y	z	D0	D1	D2	D3	D4	D5	D6	D7
0	0	0	**0**	1	1	1	1	1	1	1
0	0	1	1	**0**	1	1	1	1	1	1
0	1	0	1	1	**0**	1	1	1	1	1
0	1	1	1	1	1	**0**	1	1	1	1
1	0	0	1	1	1	1	**0**	1	1	1
1	0	1	1	1	1	1	1	**0**	1	1
1	1	0	1	1	1	1	1	1	**0**	1
1	1	1	1	1	1	1	1	1	1	**0**

Here, we use MAXTERM instead of minterm

$$D0 = x + y + z$$
$$D0 = x'' + y'' + z''$$
$$D0 = (x'.y'.z')'$$

Binary Decoder with Enable Input

- Decoders may include one or more enable inputs to control the circuit operation.
- E.g. 2-to-4 line decoder with an enable input and complimented outputs:

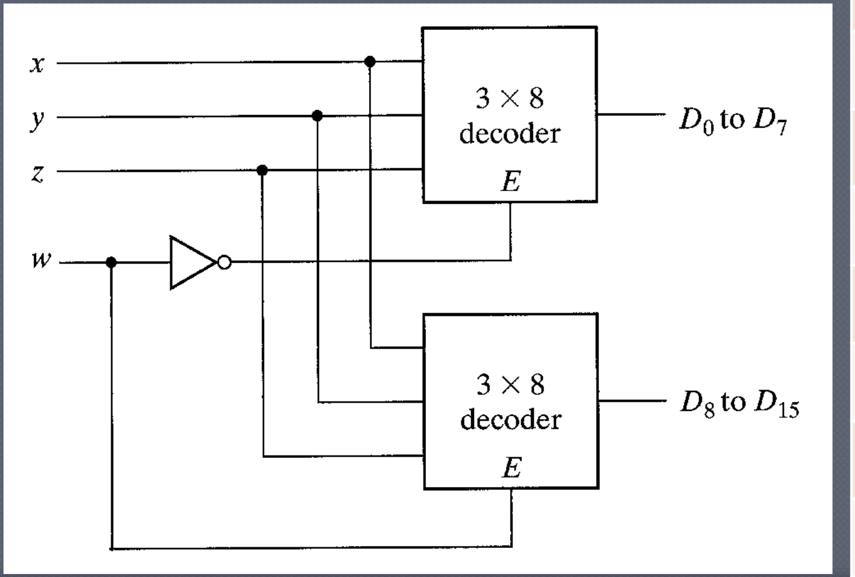


E	A	B	D0	D1	D2	D3
1	X	X	1	1	1	1
0	0	0	**0**	1	1	1
0	0	1	1	**0**	1	1
0	1	0	1	1	**0**	1
0	1	1	1	1	1	**0**

In general, the decoder can be activated when the enable is 1 or 0. In this case, the decoder is activate when enable is 0.

Multiple Binary Decoder with Enable Input

- Decoders with enable inputs can be connected to form a larger decoder circuit.
- E.g.: two 3-to-8 line decoders (uncomplemented outputs)can be connected to form a 4-to-16 line decoder.



wxyz	D(0-7)	wxyz	D(8-15)
0000	10000000	1000	10000000
0001	01000000	1001	01000000
0010	00100000	1010	00100000
0011	00010000	1011	00010000
0100	00001000	1100	00001000
0101	00000100	1101	00000100
0110	00000010	1110	00000010
0111	00000001	1111	00000001

Combinational Logic Circuits using Binary Decoder

- Recapss from Module 5 - any logic circuit design can be represented in Sum of Mintems or Product of Maxterms Boolean Expression.
- We have also learned that:
  - each output of a decoder (of a **uncomplemented outputs** type) representing one **minterm**
  - each output of a decoder (of a **complemented outputs type**) representing one **maxterm**
- Thus, a logic circuit can be built using decoder:
  - A decoder with uncomplemented outputs + OR gates = Sum of Minterms
  - A decoder with complemented outputs + AND gates = Product of Maxterms

Building a Full Adder using a Binary Decoder

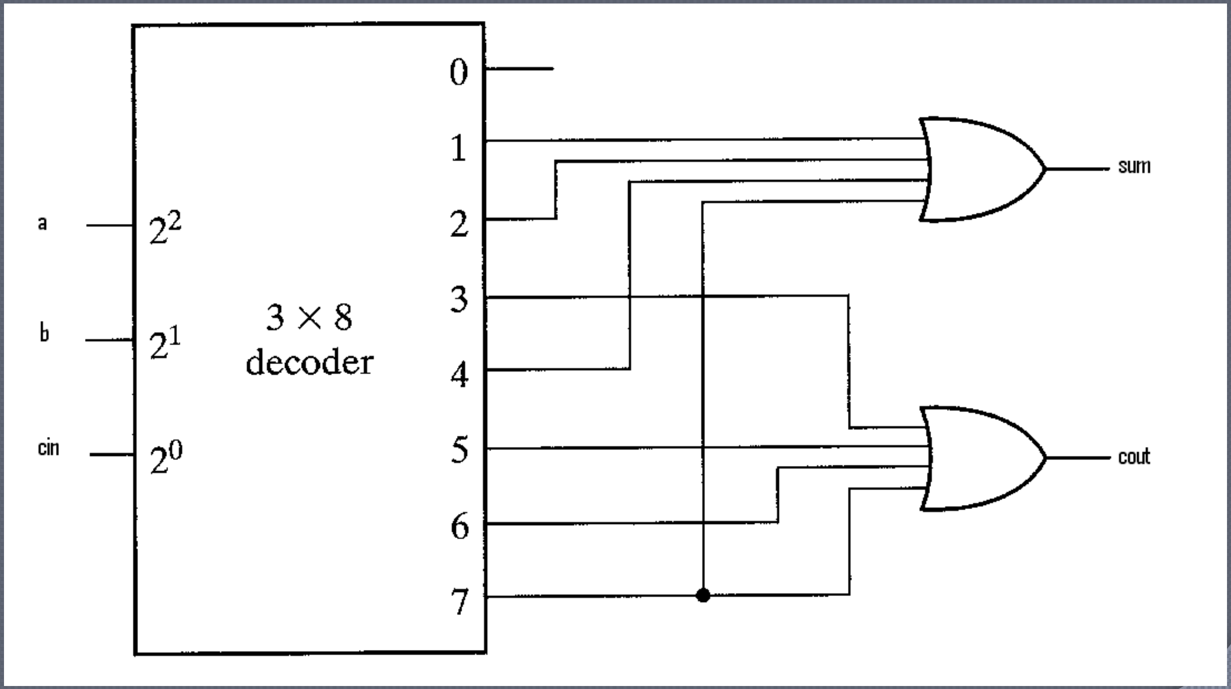
- Recapss from Module 9, a full adder properties:

  The input is **three** binary variables (***a, b, c_{in}*** being the input carry)

  The **output** is **two** binary variables (***sum, c_{out}*** being the output carry)

Sum(a, b, c_{in}) = $\sum$(m1, m2, m4, m7)

C_{out}(a, b, c_{in}) = $\sum$(m3, m5, m6, m7)



Binary Encoder

- Encoder is a digital circuit that performs the **inverse** operation of a decoder.
- An encoder has **2^n input lines** and n output lines.
- The output line generates the **binary codeword** corresponding to the input value.
- For example:
  - octal to binary encoder (8 to 3 encoder)
  - hexadecimal to binary encoder(16 to 4 encoder)

Octal to Binary Encoder

- Truth table:

D0	D1	D2	D3	D4	D5	D6	D7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

$$x = D_4 + D_5 + D_6 + D_7$$
$$y = D_2 + D_3 + D_6 + D_7$$
$$z = D_1 + D_3 + D_5 + D_7$$

- The encoder can be implemented with OR gates which boolean expressions can determined directly from the truth table.

Priority Binary Encoder

- Problem:
  - The problem with the previous design is that **only one** of the 8 inputs can have the value of '1' at a given time.
  - If more than one are 1 simultaneously, the output produces an **undefined combination**.
- Solution:
  - This give rise to the design of a priority encoder that includes priority function.
  - The operation of the priority encoder is such that if more than 1 inputs are equal to 1 at the same time, the input having the **highest priority** will take the precedence.

4bits to Binary Priority Encoder

- Truth table:

D0	D1	D2	D3	x	y	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

- The third output, V is the valid bit indicator that is set to 1 when one or more inputs are equal to 1, otherwiseV is equal to 0.
- D_3 has highest priority. When D_3 is 1, the output for x, y will be 11 regardless other inputs.
- D_0 has lowest priority.

4bits to Binary Priority Encoder

- Truth table:

D0	D1	D2	D3	x	y	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

- The third output, V is the valid bit indicator that is set to 1 when one or more inputs are equal to 1, otherwiseV is equal to 0.
- D_3 has highest priority. When D_3 is 1, the output for x, y will be 11 regardless other inputs.
- D_0 has lowest priority.

4bits to Binary Priority Encoder (cont.)

- *V* is 1 whenever any of the inputs are 1.
- Thus *V* can be derived directly from the truth table using OR gates that checks if any of the inputs are 1:

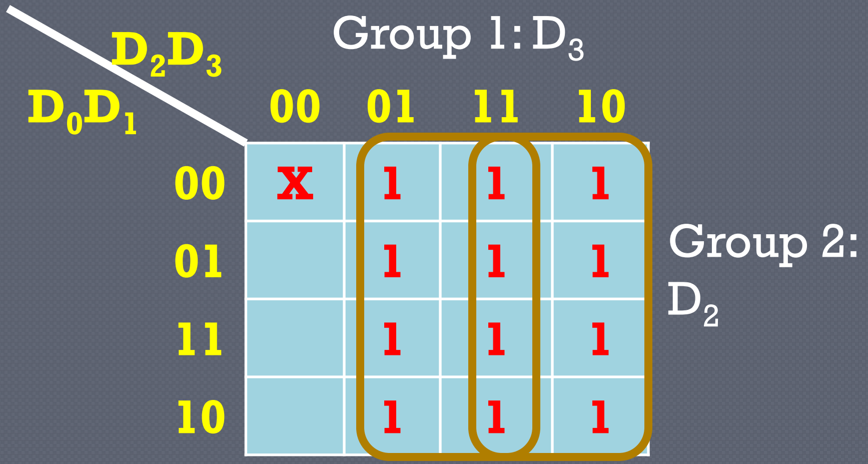
$$V = D_0 + D_1 + D_2 + D_3$$

D0	D1	D2	D3	V
0	0	0	0	0
1	0	0	0	1
X	1	0	0	1
X	X	1	0	1
X	X	X	1	1

- Whereas, the simplified Boolean equations for *x* and *y* can be derived using Karnaugh Maps.

4bits to Binary Priority Encoder (cont.)

- Deriving Boolean expression for *x*:



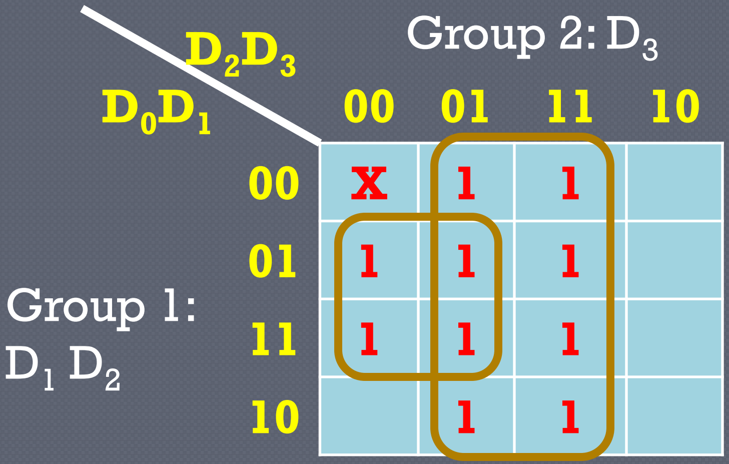
D_0D_1 \ D_2D_3	00	01	11	10
00	X	1	1	1
01		1	1	1
11		1	1	1
10		1	1	1

D0	D1	D2	D3	x
0	0	0	0	X
1	0	0	0	0
X	1	0	0	0
X	X	1	0	1
X	X	X	1	1

Thus simplified $x = D_2 + D_3$

4bits to Binary Priority Encoder (cont.)

- Deriving Boolean expression for *y*:



D_0D_1 \ D_2D_3	00	01	11	10
00	X	1	1	
01	1	1	1	
11	1	1	1	
10		1	1	

D0	D1	D2	D3	y
0	0	0	0	X
1	0	0	0	0
X	1	0	0	1
X	X	1	0	0
X	X	X	1	1

Thus simplified $y = D_{21}D_2 + D_3$

Digital Logic Design (CSNB163)

End of Module 11