Digital Logic Design

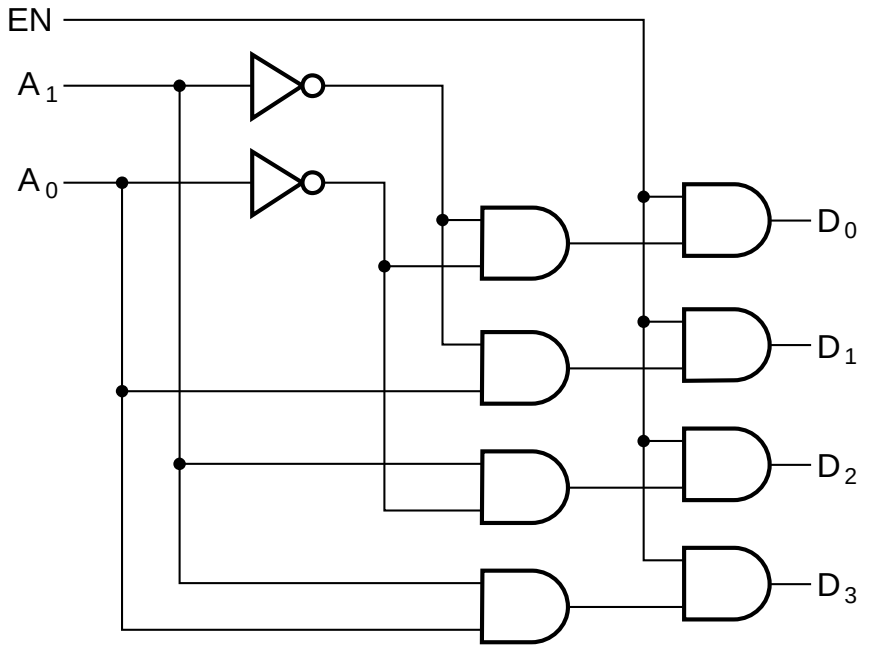
Lecture 14

Decoder with Enable

- In general, attach *m*-enabling circuits to the outputs
- See truth table below for function
  - Note use of X's to denote both 0 and 1
  - Combination containing two X's represent four binary combinations
- Alternatively, can be viewed as distributing value of signal EN to 1 of 4 outputs
- In this case, called a *demultiplexer*

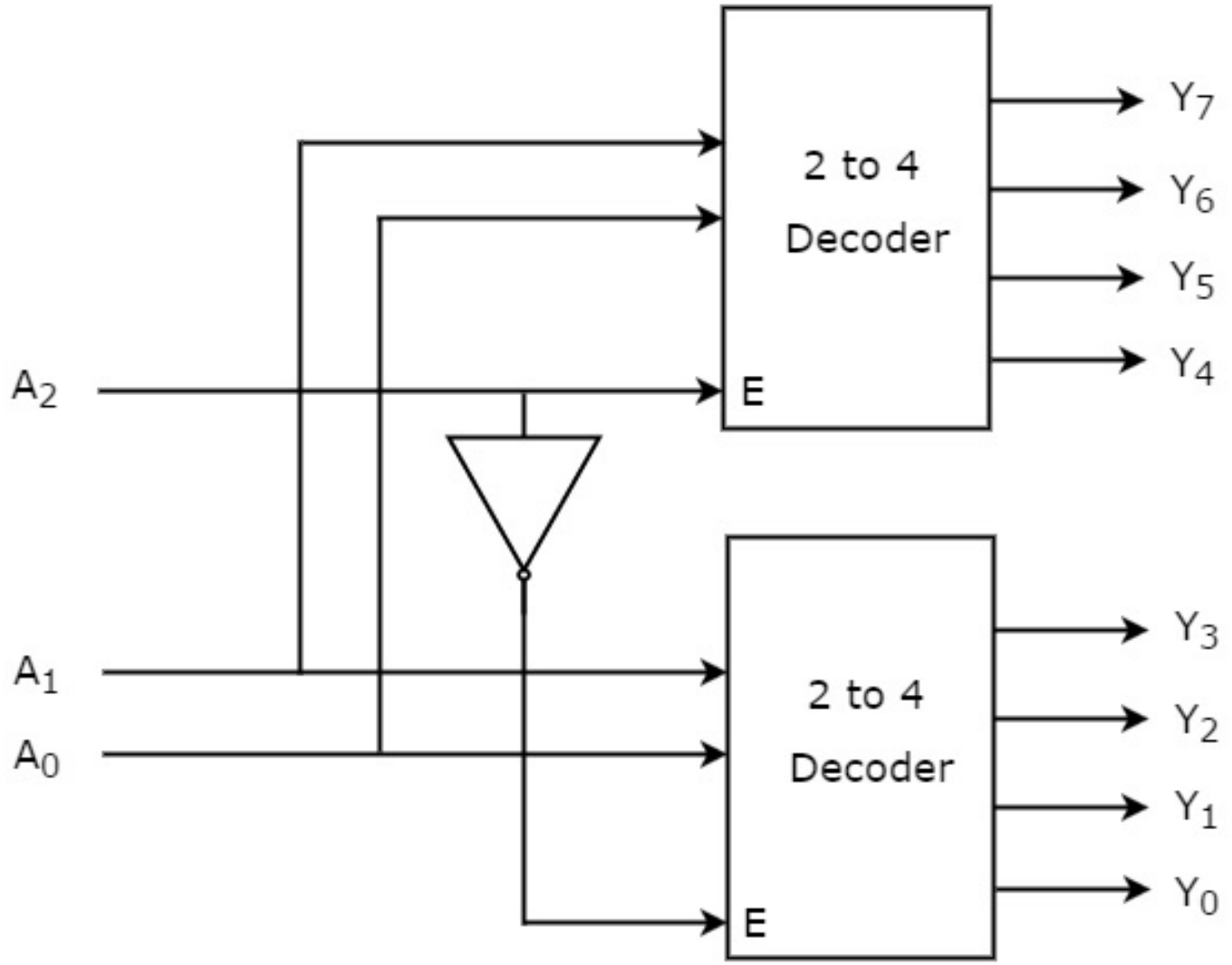
EN	A_1	A_0	D_0	D_1	D_2	D_3
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

(a)



(b)

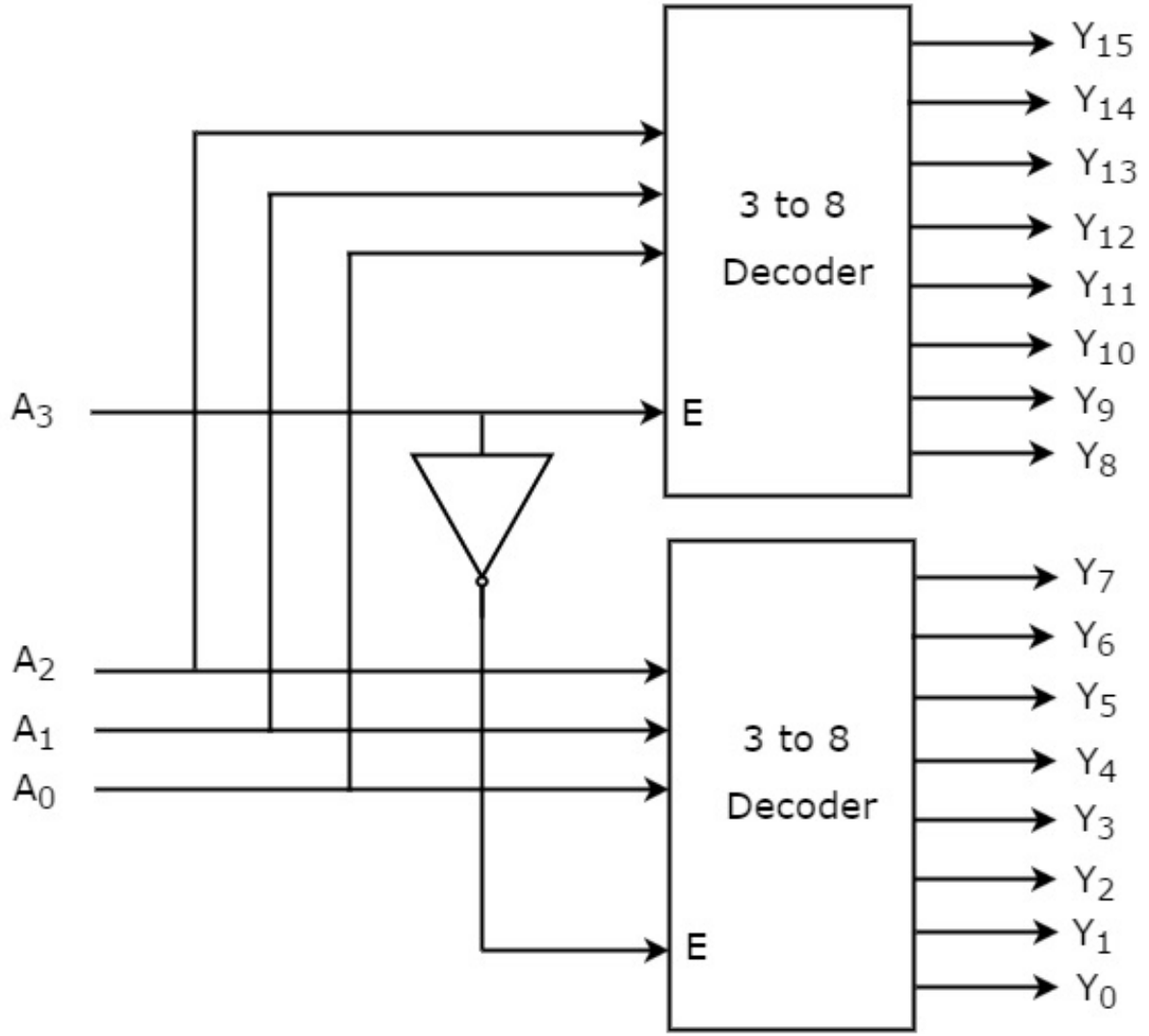
3 to 8 line decoder using 2 to 4 line decoder(s)



3 to 8 line decoder using 2 to 4 line decoder(s)

- The parallel inputs A_1 & A_0 are applied to each 2 to 4 decoder.
- The complement of input A_2 is connected to Enable, E of lower 2 to 4 decoder in order to get the outputs, Y_3 to Y_0. These are the **lower four min terms**.
- The input, A_2 is directly connected to Enable, E of upper 2 to 4 decoder in order to get the outputs, Y_7 to Y_4. These are the **higher four min terms**.

4 to 16 decoder using 3 to 8 decoders



Decoder-Based Combinational Circuits

- **A decoder provides the 2^n minterms of n input variables.**
- **Since any Boolean function can be expressed as a sum of minterms, one can use a decoder to generate the minterms and combine them with an external OR gate to form a sum-of-minterms implementation.**
- **A combinational circuit with n inputs and m outputs can be implemented with an n–to–2^n-line decoder and m OR gates.**

Binary adder Bit using Decoder and OR gate

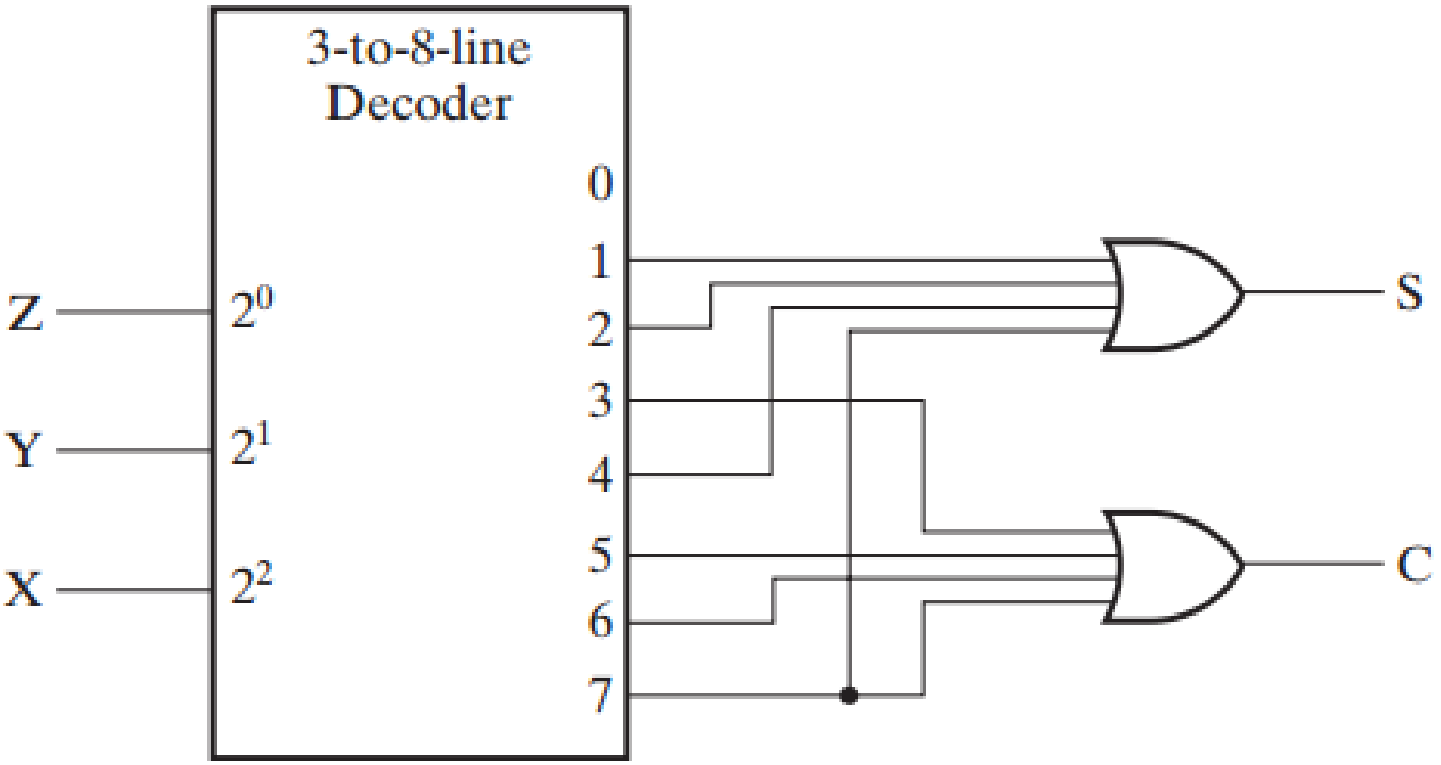
Truth Table for 1-Bit Binary Adder

X	Y	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$S(X, Y, Z) = \Sigma m(1, 2, 4, 7)$

$C(X, Y, Z) = \Sigma m(3, 5, 6, 7)$

Circuit Diagram



Encoding

- **Encoding - the opposite of decoding - the conversion of an *m*-bit input code to a *n*-bit output code with $n \leq m \leq 2^n$ such that each valid code word produces a unique output code**
- **Circuits that perform encoding are called *encoders***
- **An encoder has 2^n (or fewer) input lines and *n* output lines which generate the binary code corresponding to the input values**
- **Typically, an encoder converts a code containing exactly one bit that is 1 to a binary code corresponding to the position in which the 1 appears.**

Encoder Example

- **A decimal-to-BCD encoder**
  - **Inputs: 10 bits corresponding to decimal digits 0 through 9, (D_0, …, D_9)**
  - **Outputs: 4 bits with BCD codes**
  - **Function: If input bit D_i is a 1, then the output (A_3, A_2, A_1, A_0) is the BCD code for i,**
- **The truth table could be formed, but alternatively, the equations for each of the four outputs can be obtained directly.**

Decimal to BCD Encoder

INPUTS										OUTPUTS			
D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	A3	A2	A1	A0
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	1

Encoder Example (continued)

- **Input D_i is a term in equation A_j if bit A_j is 1 in the binary value for i.**
- **Equations:**

  $A_3 = D_8 + D_9$

  $A_2 = D_4 + D_5 + D_6 + D_7$

  $A_1 = D_2 + D_3 + D_6 + D_7$

  $A_0 = D_1 + D_3 + D_5 + D_7 + D_9$

- **$F_1 = D_6 + D_7$ can be extracted from A_2 and A_1 Is there any cost saving?**

Priority Encoder

- **If more than one input value is 1, then the encoder just designed does not work.**
- **One encoder that can accept all possible combinations of input values and produce a meaningful result is a *priority encoder*.**
- **Among the 1s that appear, it selects the most significant input position (or the least significant input position) containing a 1 and responds with the corresponding binary code for that position.**

Priority Encoder Example

- **Priority encoder with 5 inputs (D_4, D_3, D_2, D_1, D_0) - highest priority to most significant 1 present - Code outputs A2, A1, A0 and V where V indicates at least one 1 present.**

No. of Min-terms/Row	Inputs					Outputs			
	D4	D3	D2	D1	D0	A2	A1	A0	V
1	0	0	0	0	0	X	X	X	0
1	0	0	0	0	1	0	0	0	1
2	0	0	0	1	X	0	0	1	1
4	0	0	1	X	X	0	1	0	1
8	0	1	X	X	X	0	1	1	1
16	1	X	X	X	X	1	0	0	1

- **Xs in input part of table represent 0 or 1; thus table entries correspond to product terms instead of minterms. The column on the left shows that all 32 minterms are present in the product terms in the table**

Priority Encoder Example (continued)

- **Could use a K-map to get equations, but can be read directly from table and manually optimized if careful:**

$$A_2 = D_4$$

$$A_1 = \overline{D}_4 D_3 + \overline{D}_4 \overline{D}_3 D_2 = \overline{D}_4 F_1, \; F_1 = (D_3 + D_2)$$

$$A_0 = \overline{D}_4 D_3 + \overline{D}_4 \overline{D}_3 \overline{D}_2 D_1 = \overline{D}_4 (D_3 + \overline{D}_2 D1)$$

$$V = D_4 + F_1 + D_1 + D_0$$