Digital Logic Design

Lecture 14

Decoding

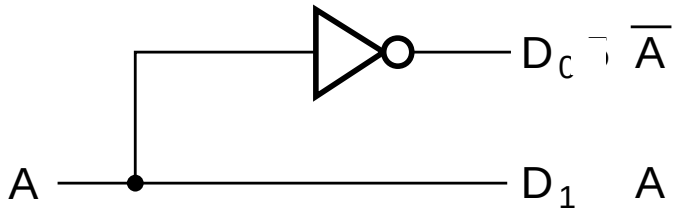
- **Decoding - the conversion of an *n*-bit input code to an *m*-bit output code with $n \leq m \leq 2^n$ such that each valid code word produces a unique output code**
- **Circuits that perform decoding are called *decoders***
- **Here, functional blocks for decoding are**
  - **called *n*-to-*m* line decoders, where $m \leq 2^n$, and**
  - **generate 2^n (or fewer) minterms for the *n* input variables**

Decoder Examples

- **1-to-2-Line Decoder**

A	D_0	D_1
0	1	0
1	0	1

(a)

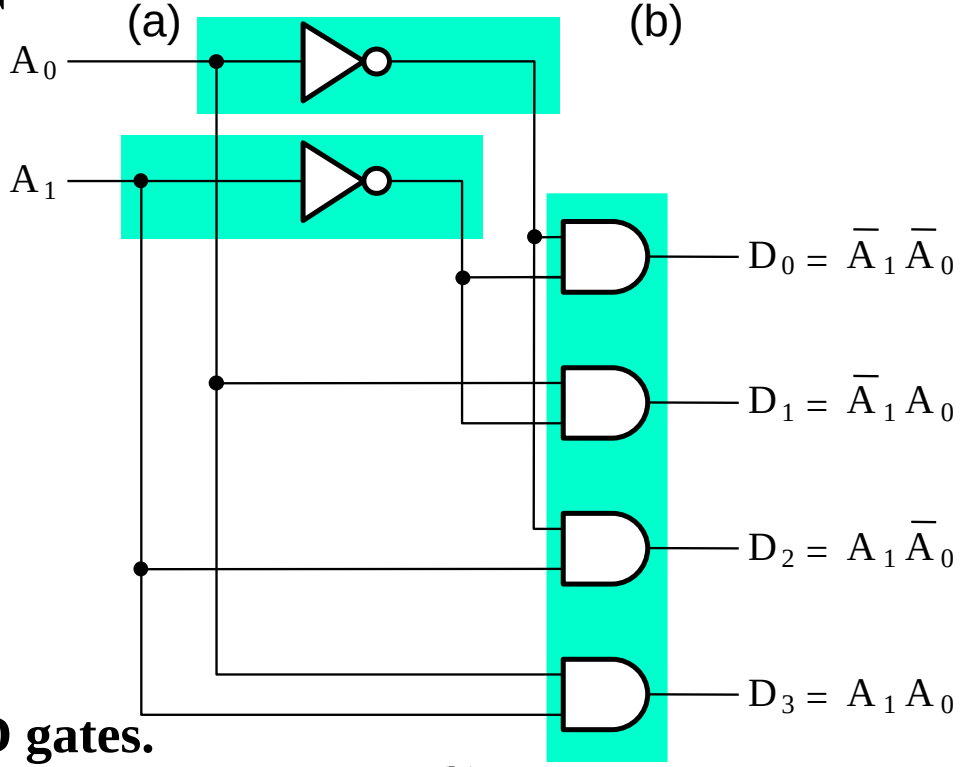


(b)

- **2-to-4-Line Decoder**

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

(a)



(b)

- **Note that the 2-4-line made up of 2 1-to-2-line decoders and 4 AND gates.**

Decoder Expansion

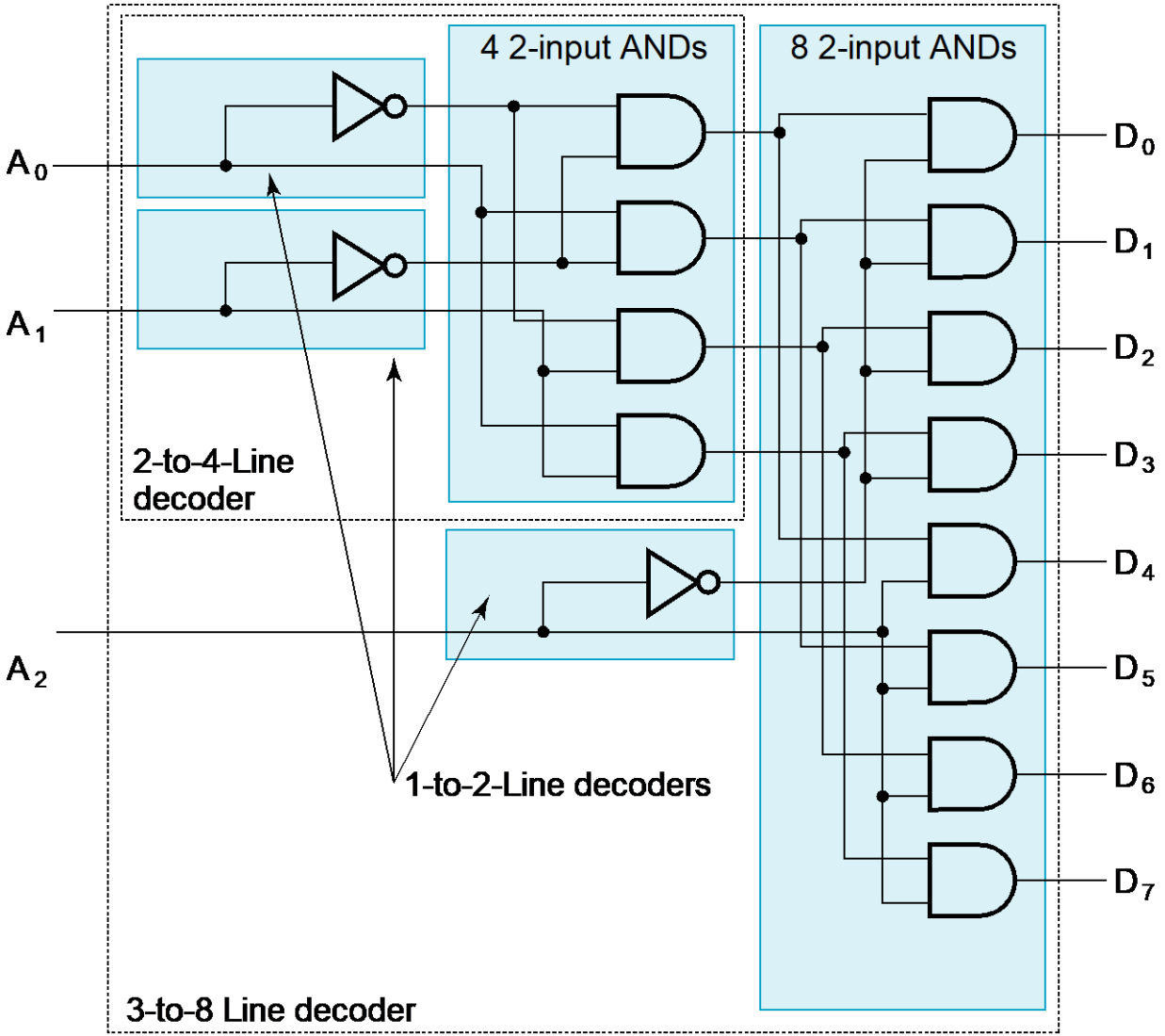
- **General procedure given in book for any decoder with *n* inputs and 2^n outputs.**
- **This procedure builds a decoder backward from the outputs.**
- **The output AND gates are driven by two decoders with their numbers of inputs either equal or differing by 1.**
- **These decoders are then designed using the same procedure until 2-to-1-line decoders are reached.**
- **The procedure can be modified to apply to decoders with the number of outputs $\neq 2^n$**

Decoder Expansion - Example 1

- 3-to-8-line decoder
  - Number of output ANDs = 8
  - Number of inputs to decoders driving output ANDs = 3
  - Closest possible split to equal
    - 2-to-4-line decoder
    - 1-to-2-line decoder
  - 2-to-4-line decoder
    - Number of output ANDs = 4
    - Number of inputs to decoders driving output ANDs = 2
    - Closest possible split to equal
      - Two 1-to-2-line decoders
- See next slide for result

Decoder Expansion - Example 1

- **Result**



Decoder Expansion - Example 2

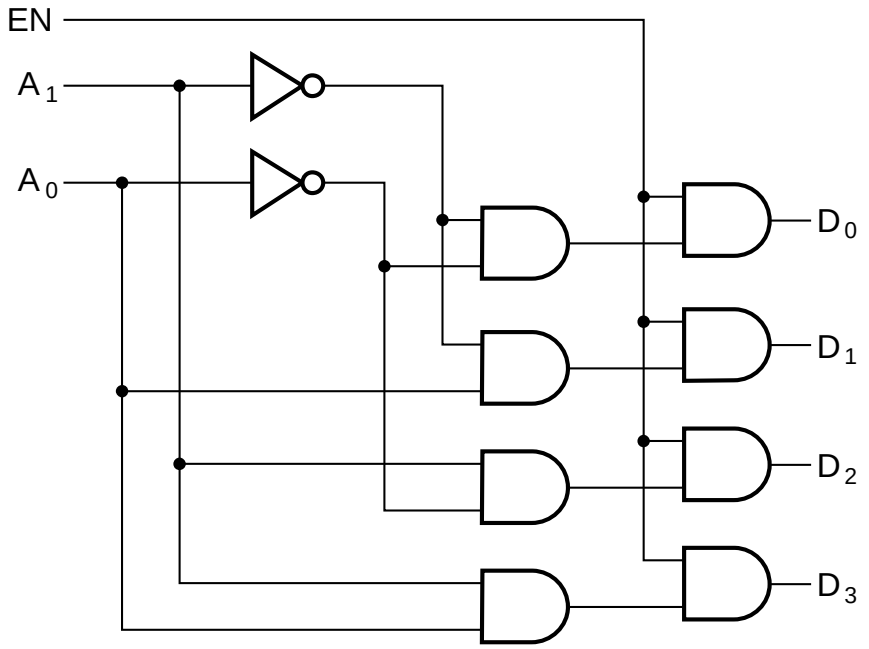
- 7-to-128-line decoder
  - Number of output ANDs = 128
  - Number of inputs to decoders driving output ANDs = 7
  - Closest possible split to equal
    - 4-to-16-line decoder
    - 3-to-8-line decoder
  - 4-to-16-line decoder
    - Number of output ANDs = 16
    - Number of inputs to decoders driving output ANDs = 2
    - Closest possible split to equal
      - 2 2-to-4-line decoders
  - Complete using known 3-8 and 2-to-4 line decoders

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*

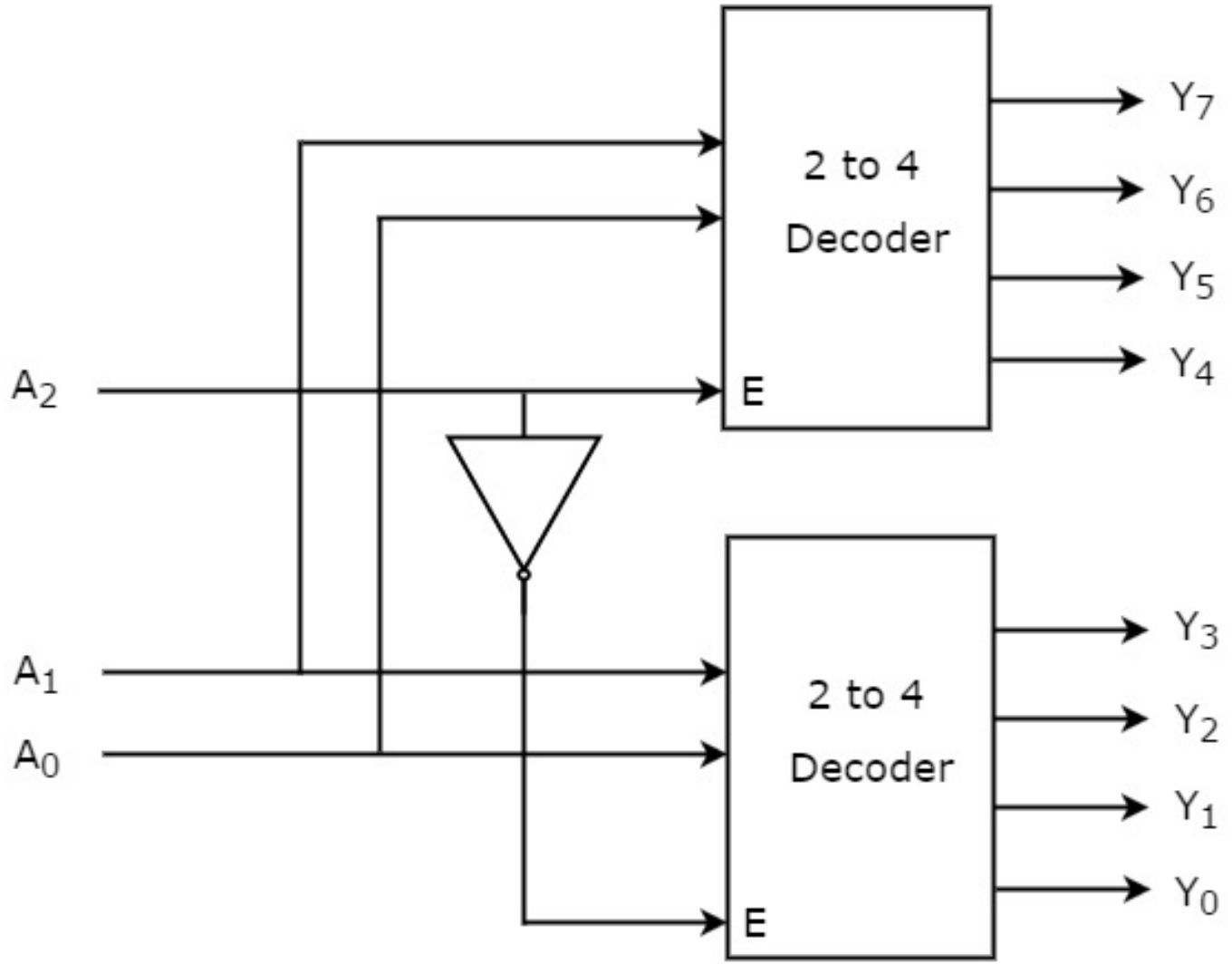
EN	A_1	A_0	D_0	D_1	D_2	D_3
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

(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