Lecture 7

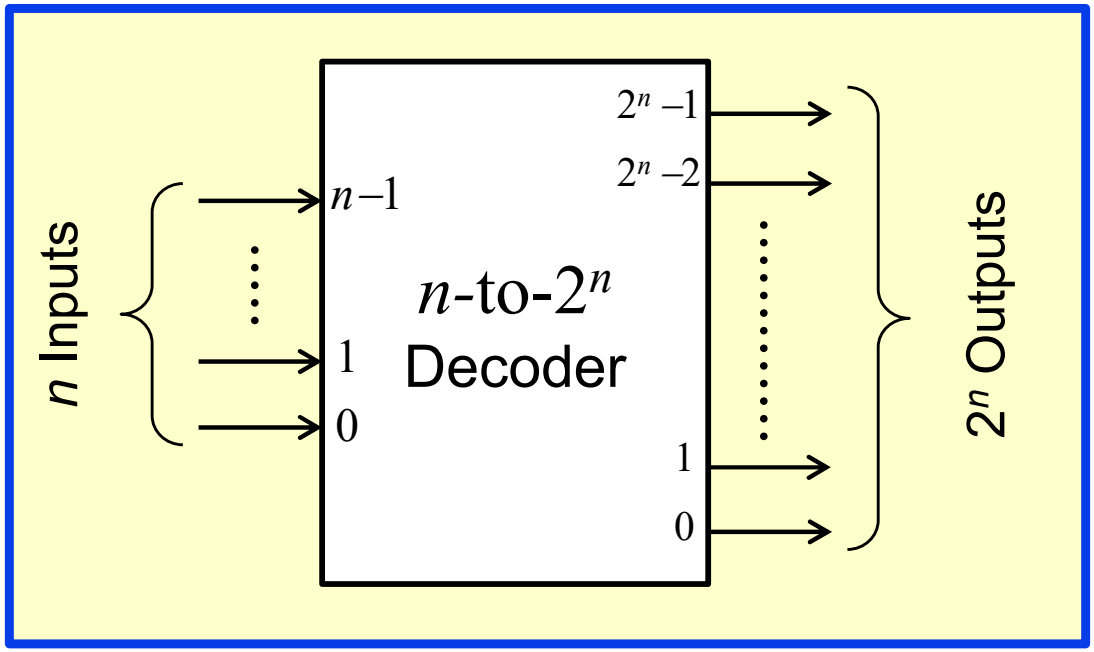
Decoders & Encoders

Learning outcomes

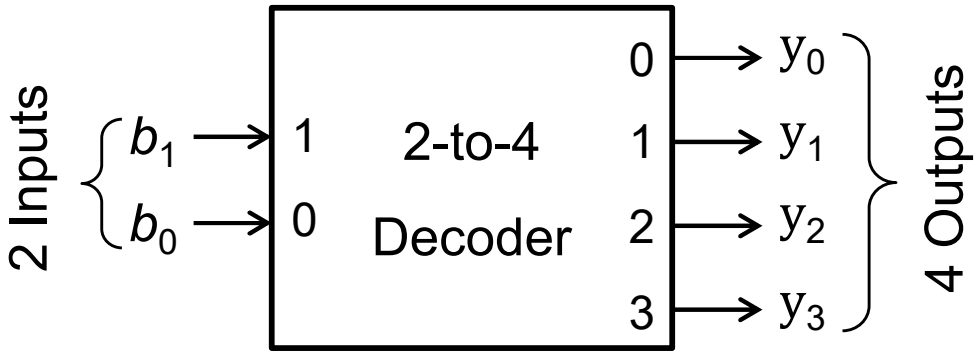
- Explain the operation of a decoder and an encoder.
- Expand a decoder to handle more data inputs.
- Use a decoder to implement a Boolean function.
- Apply decoders to specific applications.

Decoder

- A decoder is a combinational circuit that:
  - Has an n-bit binary code as input.
  - Has 2^n output lines, with each output corresponding to one n-bit binary code.
  - Decodes the input code and activates the corresponding output line, which becomes different from all other output lines.
  - Can have less than 2^n outputs (i.e. $m \leq 2^n$) outputs lines if not all input combinations are used.
  - Essentially performs a conversion from an n-bit input code to an m-bit output code, where $n \leq m \leq 2^n$.

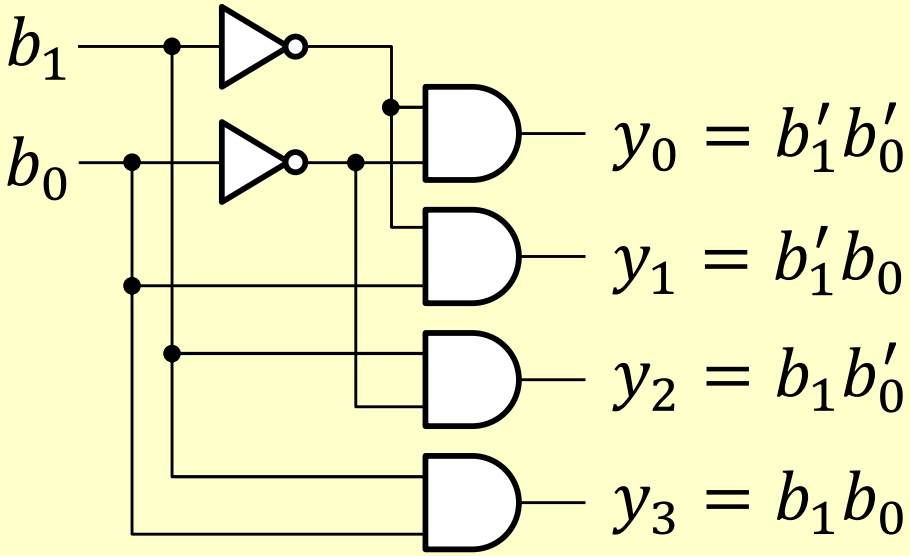


2-to-4 Decoder – Truth Table and implementation



Inputs		Outputs			
b_1	b_0	y_0	y_1	y_2	y_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

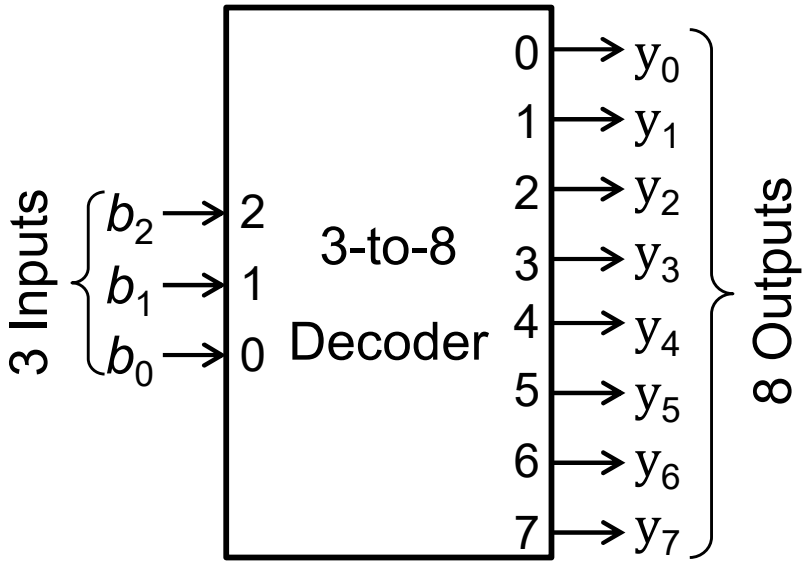
- Each input code activates a single output

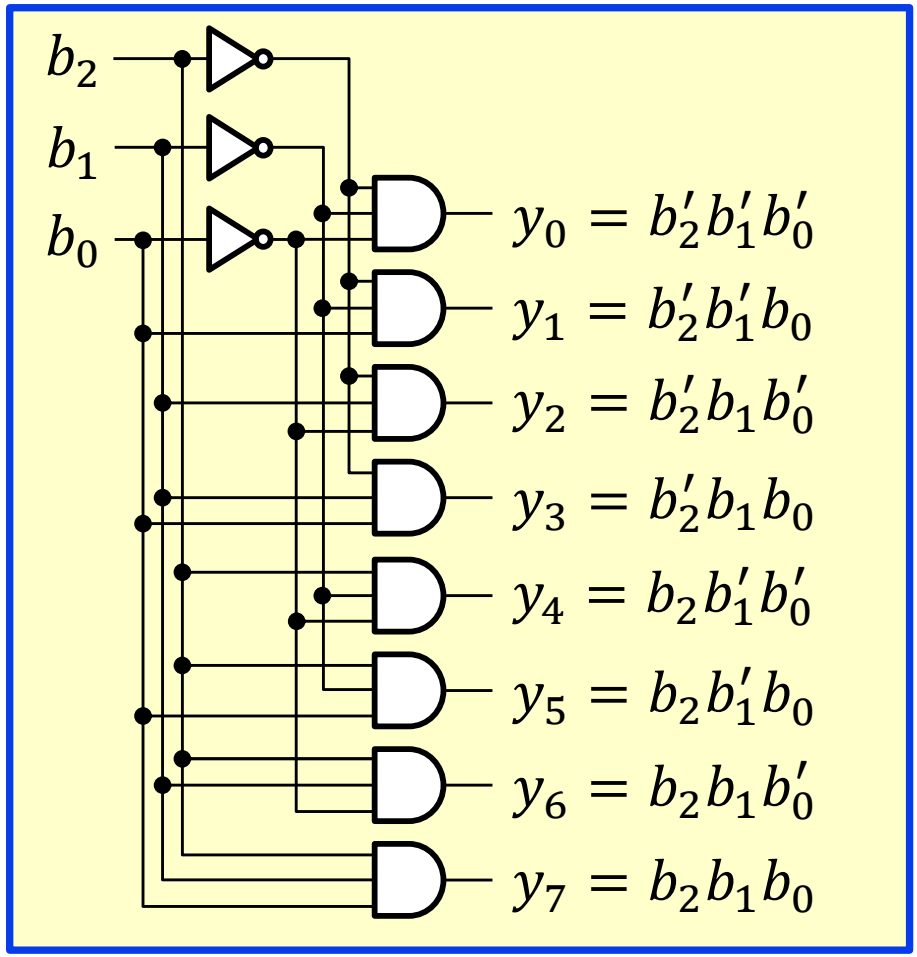


- Each decoder output is a minterm (i.e. product term)
- Implementing a decoder thus only requires AND gates with inverters

3-to-8 Decoder – Truth Table and implementation

Inputs			Outputs							
b_2	b_1	b_0	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_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



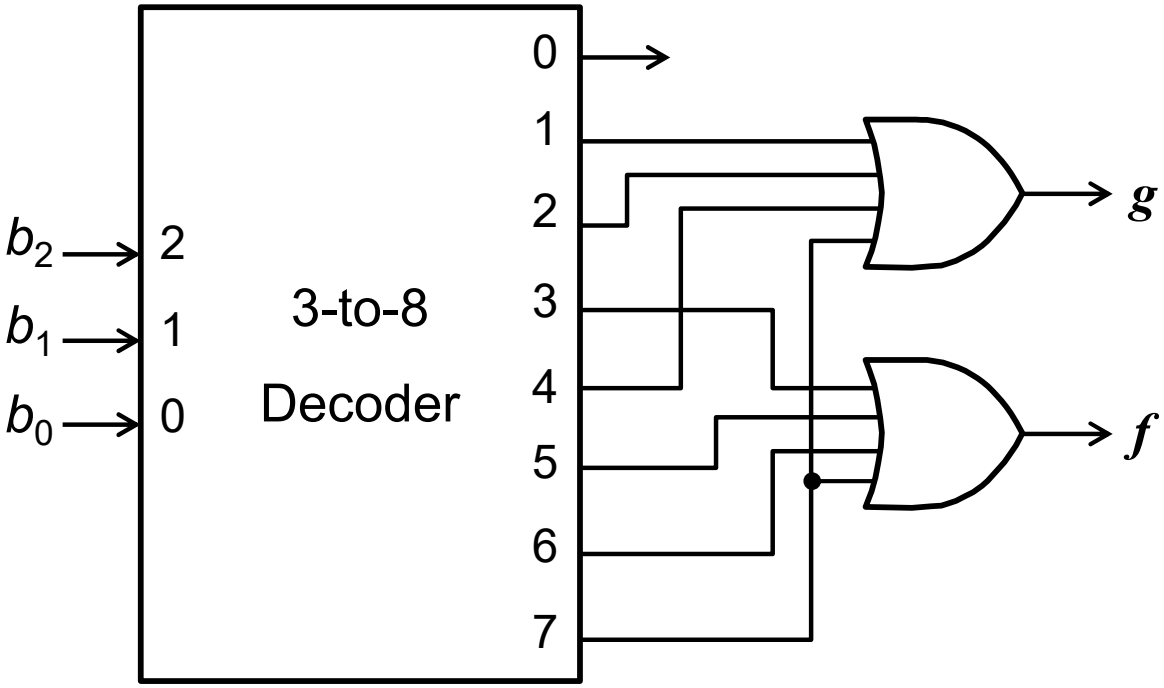


- Each decoder output is a minterm
- Implementing a decoder thus only requires AND gates with inverters

Implementing Boolean functions with decoders

- Any Boolean function of n input variables can be implemented using a n-to-2^n decoder and a logic gate.
- Since decoders generate all minterms (i.e. product terms), one can:
  - Connect input variables to the inputs of the decoder to generate the minterms
  - OR together the minterms for which Boolean function =1
- Decoder is particularly advantageous when multiple Boolean functions of the same input variables need to be implemented.

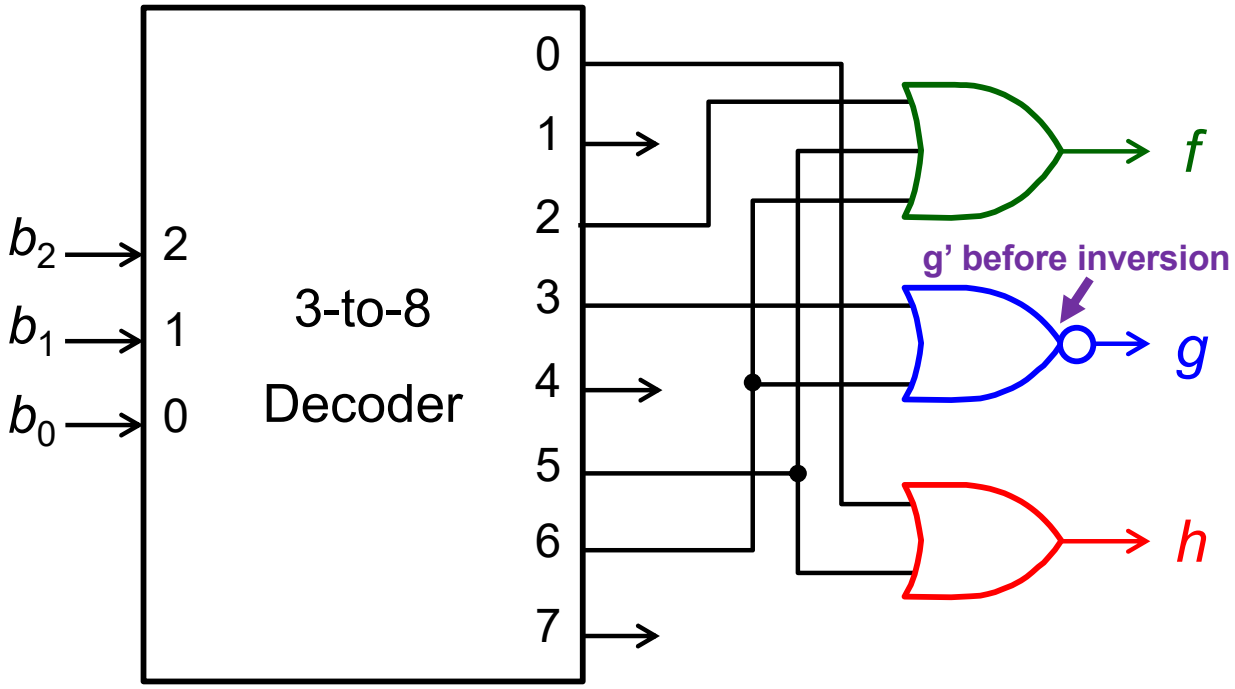
Inputs	Outputs	
$b_2\ b_1\ b_0$	f	g
0 0 0	0	0
0 0 1	0	1
0 1 0	0	1
0 1 1	1	0
1 0 0	0	1
1 0 1	1	0
1 1 0	1	0
1 1 1	1	1



Implementing Boolean functions with decoders

- Using the following 3-to-8 decoder and gates, implement the following Boolean functions: $f = \Sigma(2, 5, 6)$, $g = \Pi(3, 6)$, $h = \Sigma(0, 5)$

Inputs	Outputs		
$b_2b_1b_0$	f	g	h
0 0 0	0	1	1
0 0 1	0	1	0
0 1 0	1	1	0
0 1 1	0	0	0
1 0 0	0	1	0
1 0 1	1	1	1
1 1 0	1	0	0
1 1 1	0	1	0



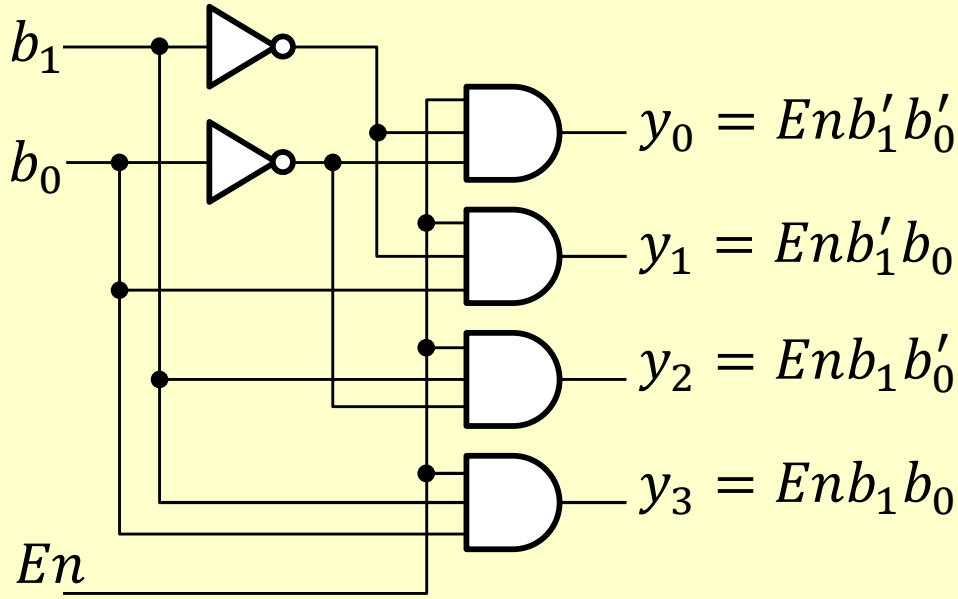
- Avoid large fan-in output gates:
  - when there are fewer ones, use an OR gate to implement the function by Oring the output lines for which function is equal to 1.
  - when there are fewer zeroes, use an NOR gate to implement the function by NORing the output lines for which function is equal to 0.

2-to-4 Decoder with Enable input

- The enable input controls the operation of the decoder.
- It is also useful when connecting two or more IC decoder packages to form an expanded decoder with more inputs and outputs.

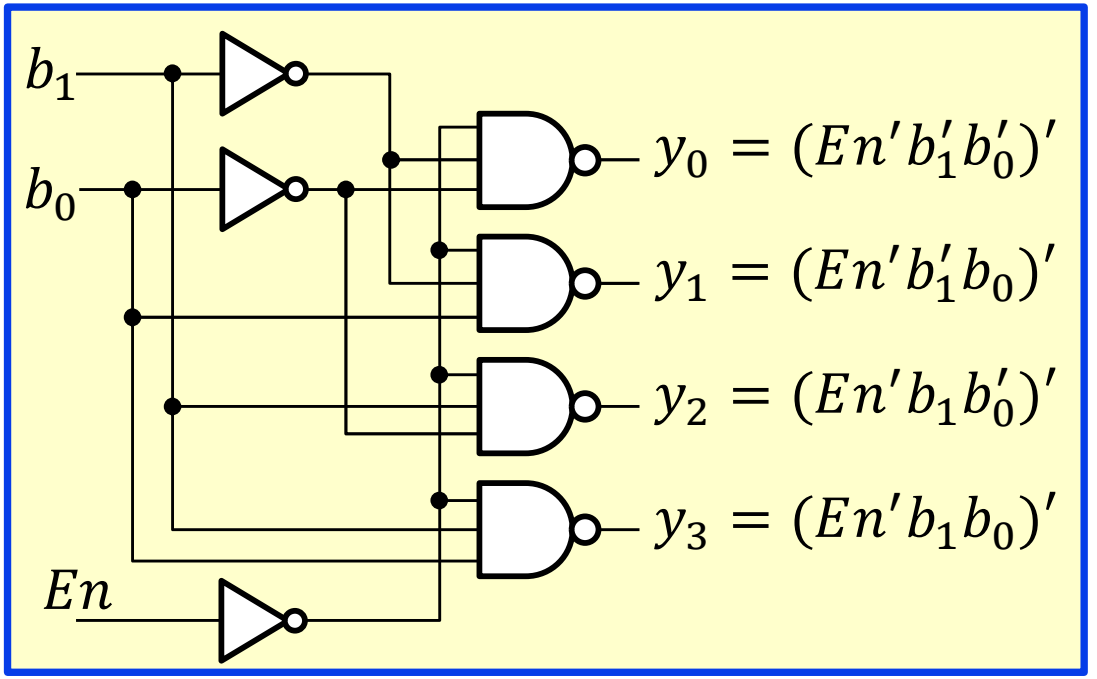
Inputs		Outputs			
En	b_1 b_0	y_0	y_1	y_2	y_3
0	X X	0	0	0	0
1	0 0	1	0	0	0
1	0 1	0	1	0	0
1	1 0	0	0	1	0
1	1 1	0	0	0	1

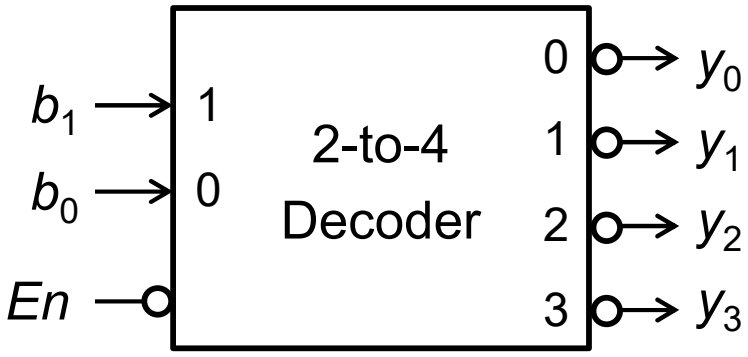
- If *En* is not active (i.e. zero) then all outputs are here zeros, regardless of b_1 and b_0



NAND Decoders with inverted outputs

Inputs		Outputs			
En	b_1 b_0	y_0	y_1	y_2	y_3
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



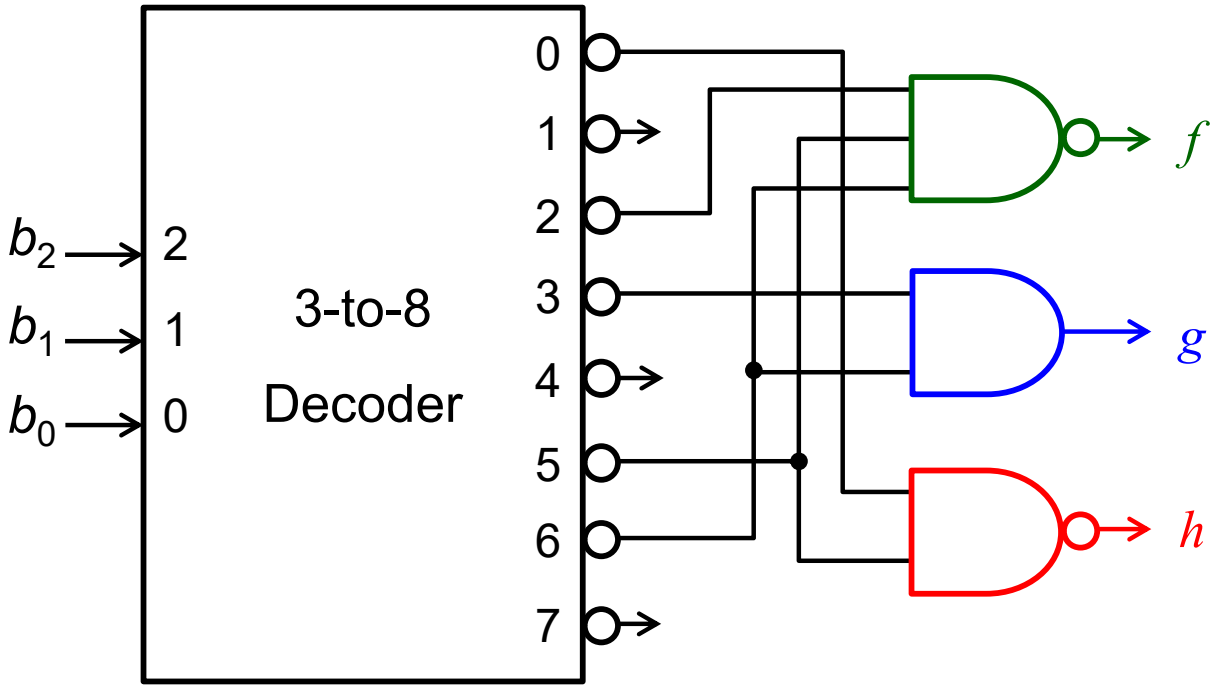


- Decoders can also be implemented using NAND gates (instead of AND gates).
- Their outputs are inverted: the selected output line is driven low while the rest of output lines remain high.
- The enable signal is here also active when driven low.

NAND Decoders with inverted outputs

- Using the following 3-to-8 decoder and NAND gates only, implement the following Boolean functions: $f = \sum(2, 5, 6)$, $g = \prod(3, 6)$, $h = \sum(0, 5)$

Inputs	Outputs		
$b_2b_1b_0$	f	g	h
0 0 0	0	1	1
0 0 1	0	1	0
0 1 0	1	1	0
0 1 1	0	0	0
1 0 0	0	1	0
1 0 1	1	1	1
1 1 0	1	0	0
1 1 1	0	1	0



- Avoid large fan-in output gates:
  - when there are fewer 1's, use a NAND gate to implement the function by NANDing the output lines for which function is equal to 1.
  - when there are fewer zeroes, use an AND gate to implement the function by ANDing the output lines for which function is equal to 0.

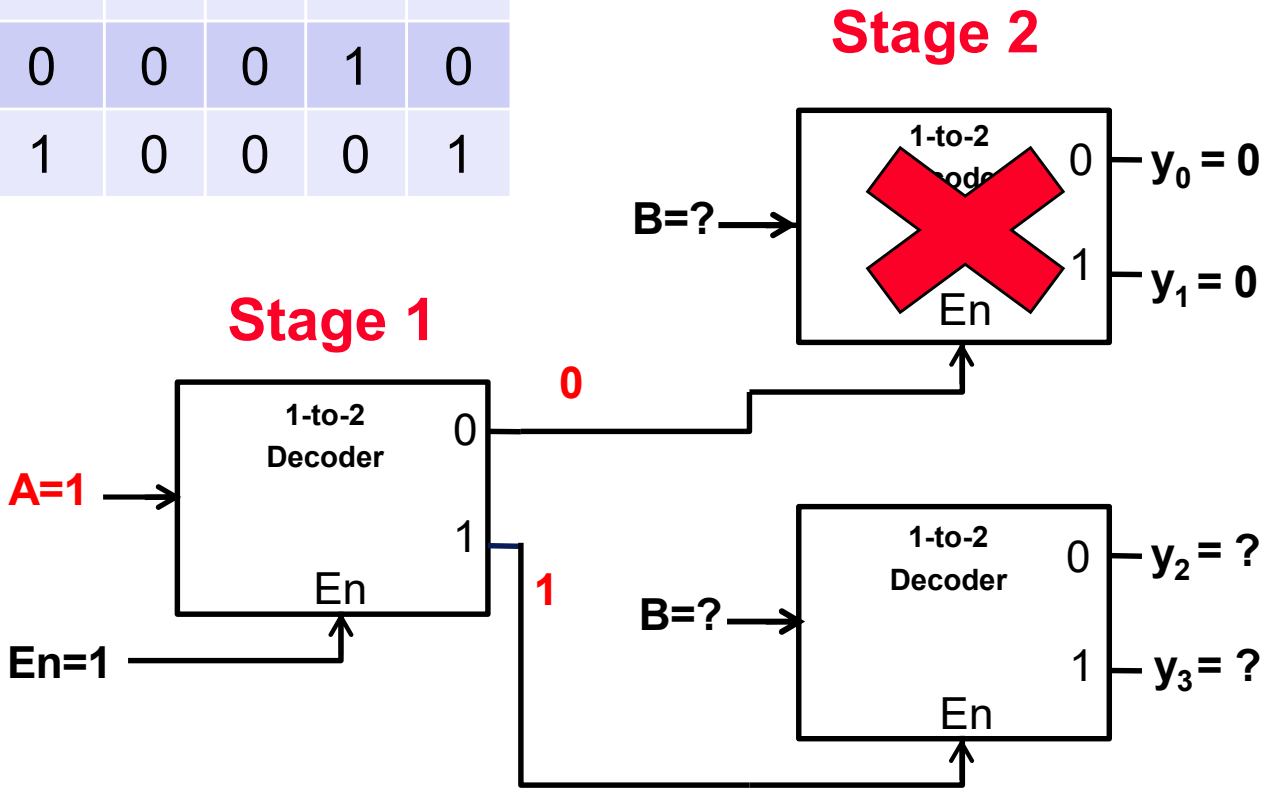
How to build larger decoders?

- Larger decoders can be built by cascading smaller decoders:
  - Starting from the last stage (outputs), place as many small decoders as required to provide the desired number (i.e. 2^n) of outputs
  - Connect the Enable inputs of decoders of a given stage to the outputs of the previous stage.
  - In a given stage, the decoders should decode the same input bit(s). Usually, MSB is fed to the first stage and LSB to the last output stage.
- The basic idea involves a process of elimination:
  - Given a 2-bit number AB, with A being the MSB
  - If you are told that A=1, then AB can only be 2 or 3
  - If you are then told that B=0, then you know the number is 2
- By decoding one bit at a time, one can eliminate half of the possible numbers until the actual number is decoded

Building a 2-to-4 decoder from 1-to-2 decoders

Inputs			Outputs			
En	A	B	y_0	y_1	y_2	y_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

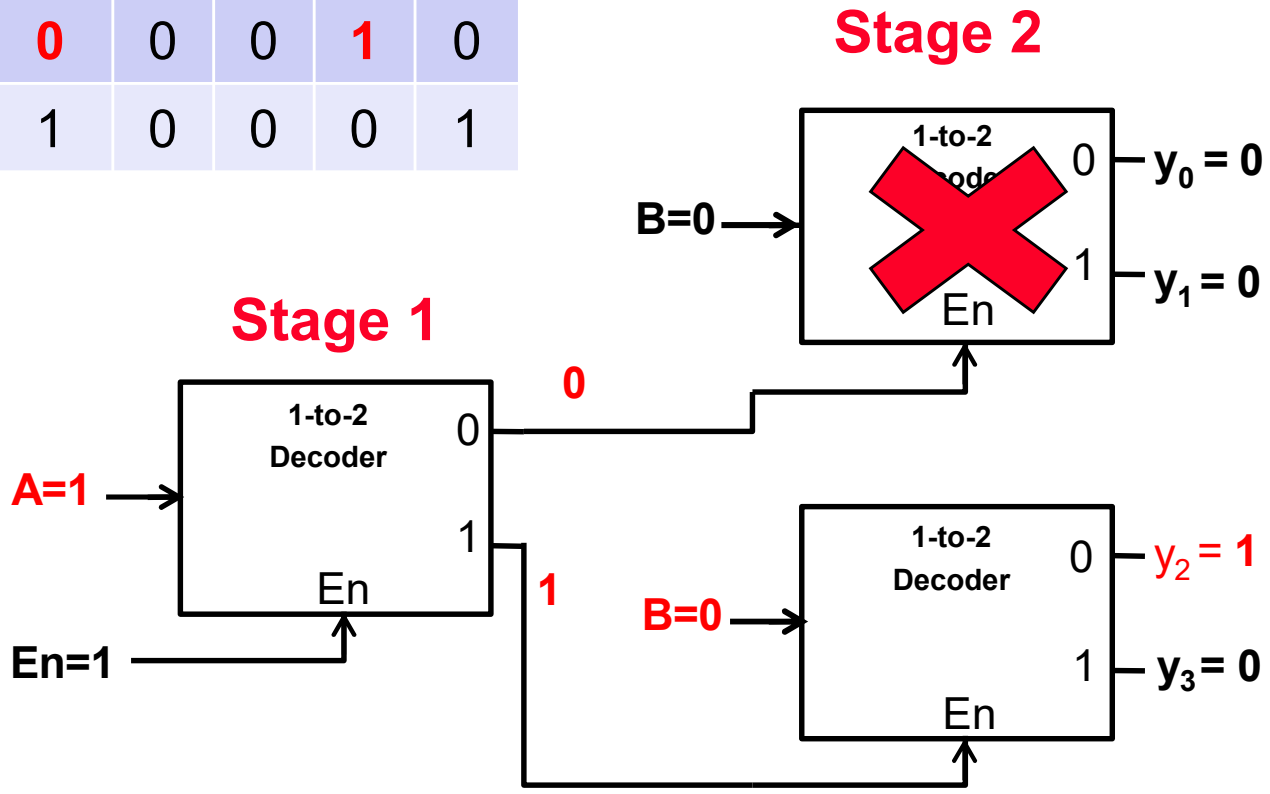
- If **A=1** then the top decoder of the second stage is disabled while the bottom decoder is enabled.
- We can thus narrow it down to **y_2 or y_3**



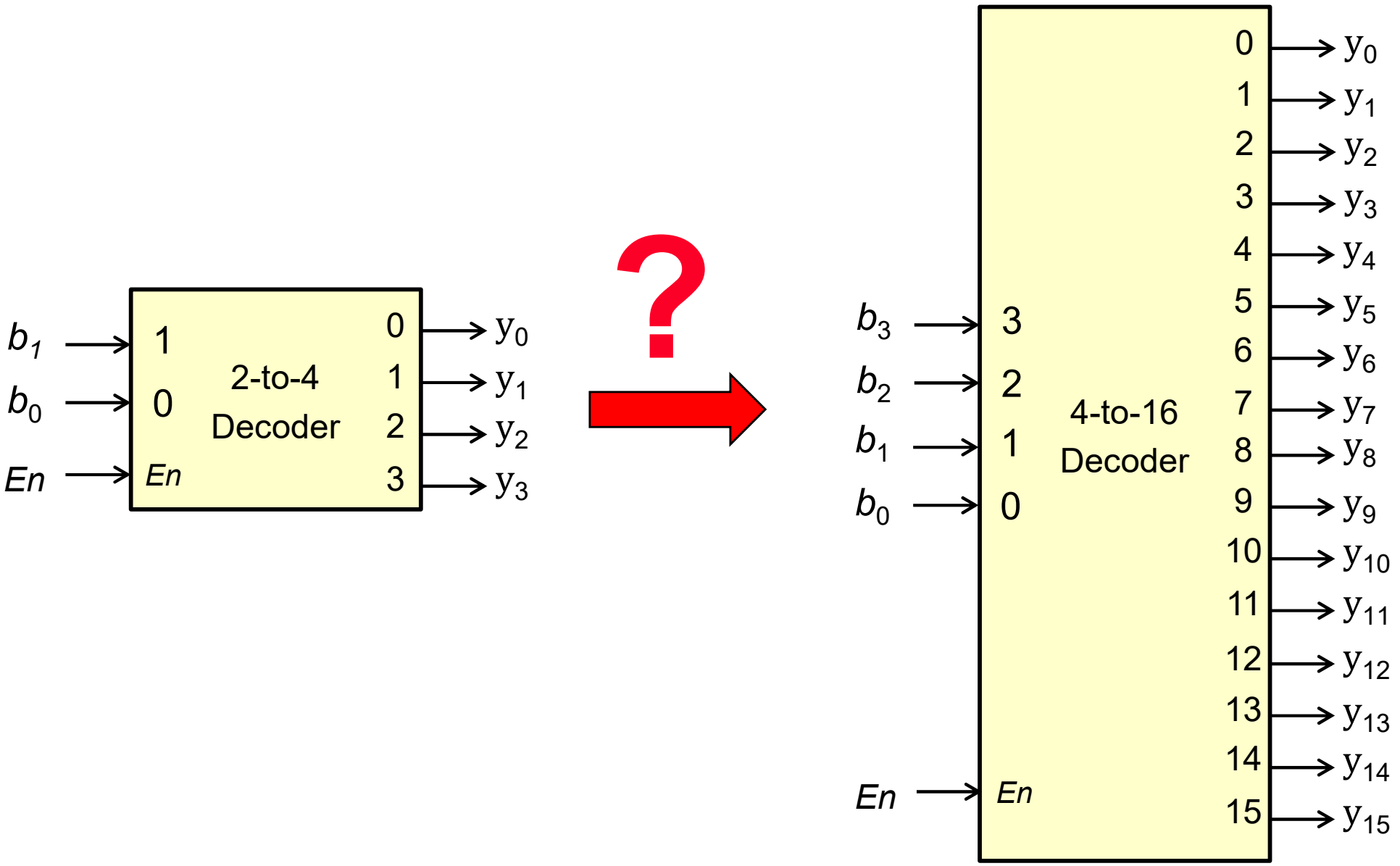
Building 2-to-4 decoder from 1-to-2 decoders

Inputs			Outputs			
En	*A*	*B*	y_0	y_1	y_2	y_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

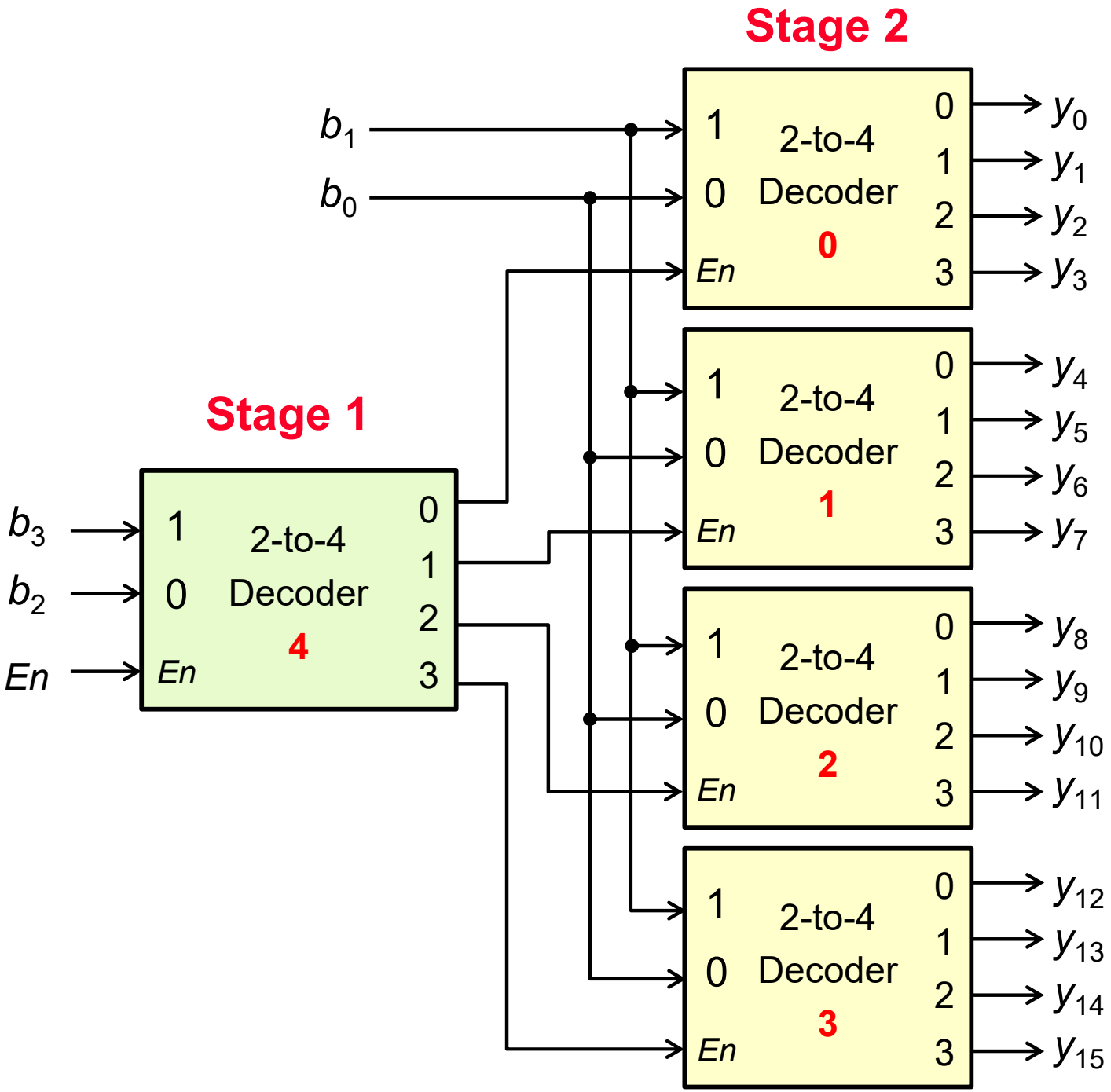
- If **A=1** then the top decoder of the second stage is disabled while the bottom decoder is enabled.
- We can thus narrow it down to y_2 or y_3
- If now **B=0** then y_2=1 as expected for AB=10 of the built 2-to-4 decoder.



Building a 4-to-16 decoder from 2-to-4 decoders

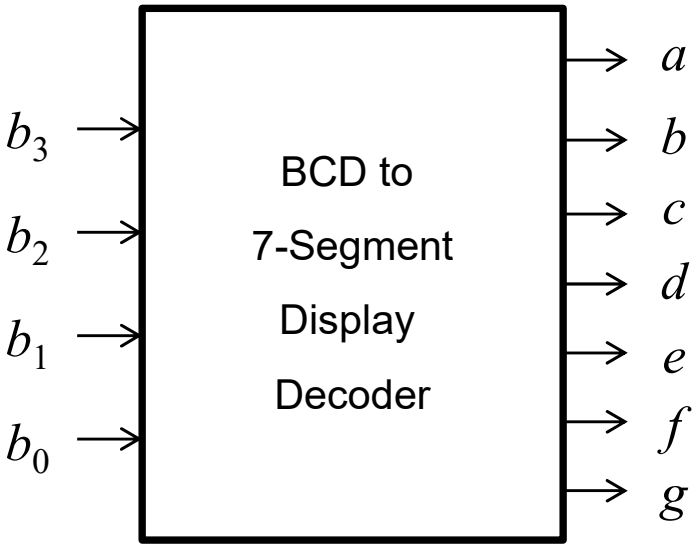


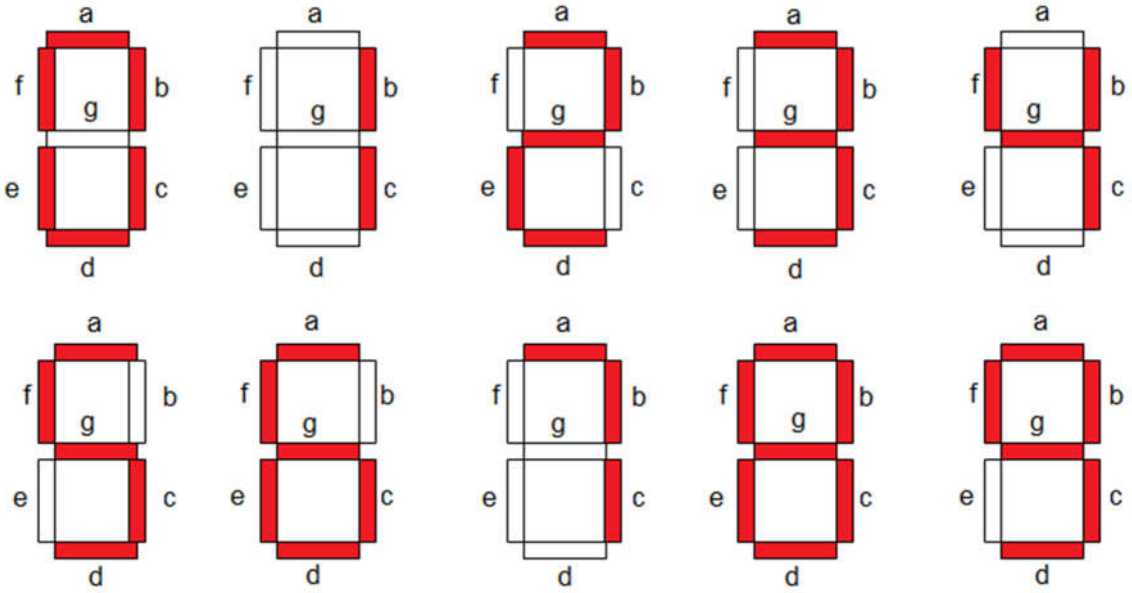
Building a 4-to-16 decoder from 2-to-4 decoders



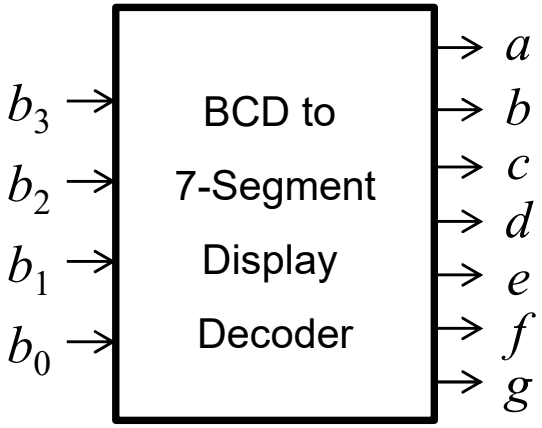
BCD to 7-Segment Display Decoder

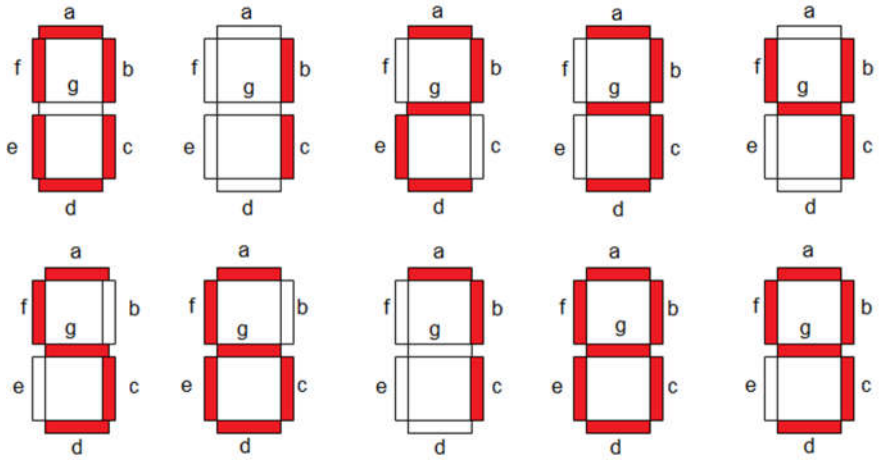
- A 7-segment display is made up of seven individual LEDs, typically labeled a–g. It is commonly found in digital clocks and household appliances.
- The input to the decoder is the binary equivalent of the decimal (i.e. Binary Coded Decimal or BCD) that is to be displayed.
- The outputs of the decoder determine which individual LED segments are turned ON to display the input decimal number.





BCD to 7-Segment Display Decoder



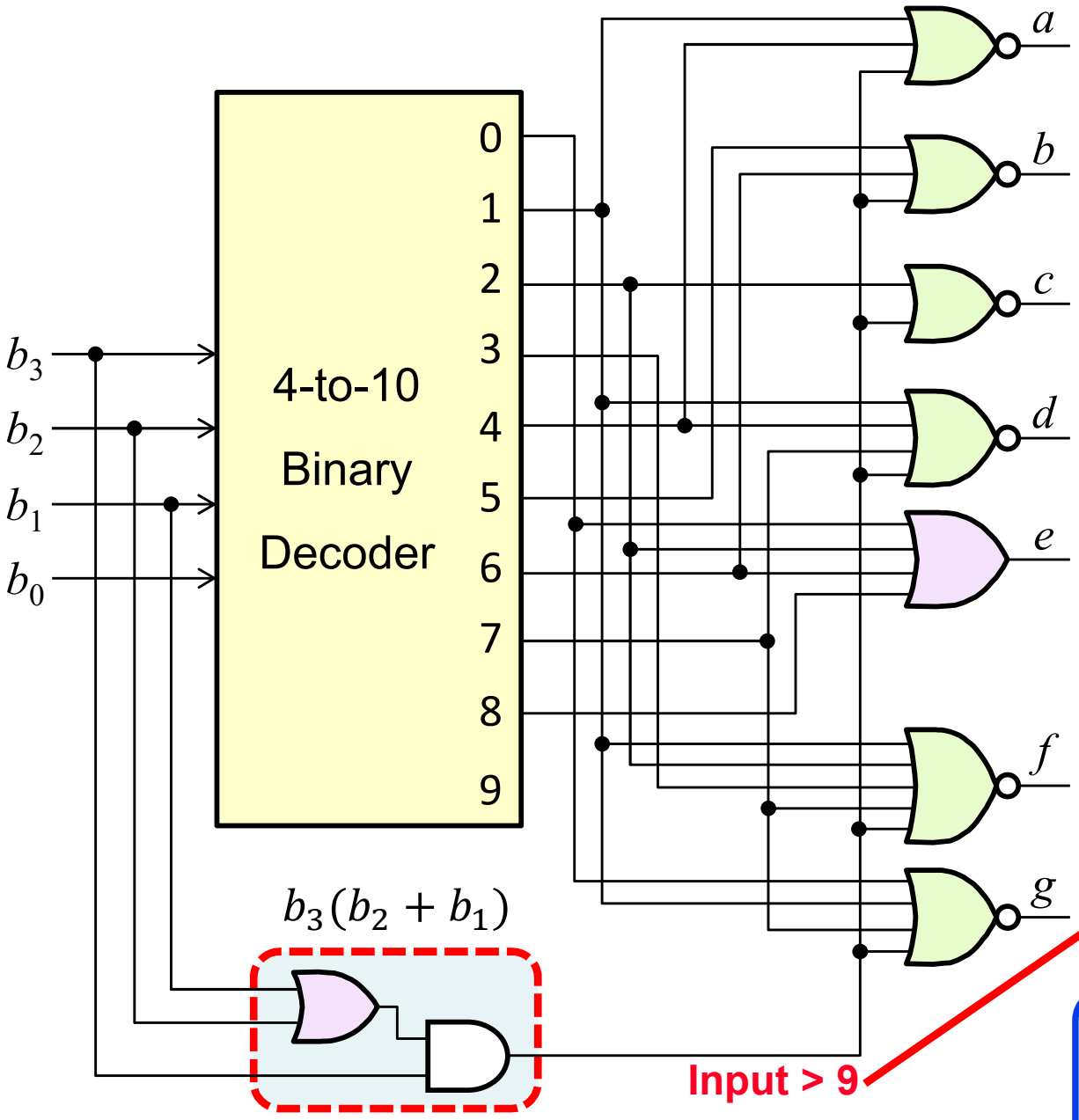


- Decoders with four inputs could also drive characters “0” to “F” with the case of the hex characters being “A, b, c or C, d, E, and F.”

BCD input $b_3\ b_2\ b_1\ b_0$	7-Segment Outputs a b c d e f g
0 0 0 0	1 1 1 1 1 1 0
0 0 0 1	0 1 1 0 0 0 0
0 0 1 0	1 1 0 1 1 0 1
0 0 1 1	1 1 1 1 0 0 1
0 1 0 0	0 1 1 0 0 1 1
0 1 0 1	1 0 1 1 0 1 1
0 1 1 0	1 0 1 1 1 1 1
0 1 1 1	1 1 1 0 0 0 0
1 0 0 0	1 1 1 1 1 1 1
1 0 0 1	1 1 1 1 0 1 1
1 0 1 0	0 0 0 0 0 0 0
.....	0 0 0 0 0 0 0
1 1 1 1	0 0 0 0 0 0 0

All 0’s

BCD to 7-Segment Display Decoder



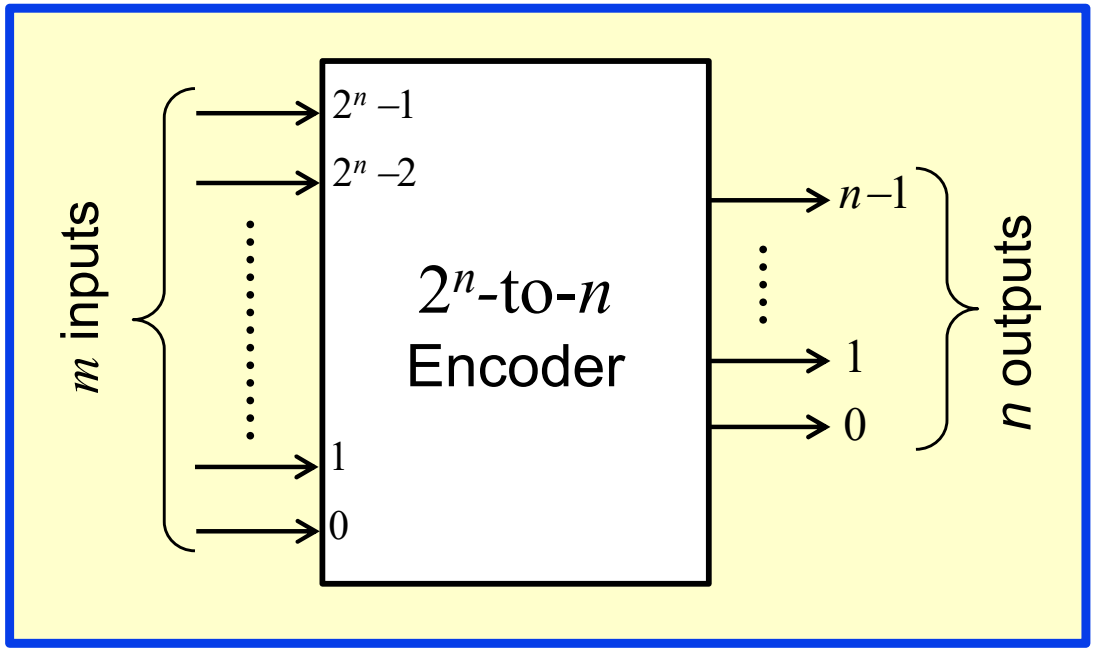
BCD input $b_3\ b_2\ b_1\ b_0$	7-Segment Outputs a b c d e f g
0 0 0 0	1 1 1 1 1 1 0
0 0 0 1	0 1 1 0 0 0 0
0 0 1 0	1 1 0 1 1 0 1
0 0 1 1	1 1 1 1 0 0 1
0 1 0 0	0 1 1 0 0 1 1
0 1 0 1	1 0 1 1 0 1 1
0 1 1 0	1 0 1 1 1 1 1
0 1 1 1	1 1 1 0 0 0 0
1 0 0 0	1 1 1 1 1 1 1
1 0 0 1	1 1 1 1 0 1 1
1 0 1 0	0 0 0 0 0 0 0
.....	0 0 0 0 0 0 0
1 1 1 1	0 0 0 0 0 0 0

- Use OR gates when fewer 1's.
- Use NOR gates when fewer 0's.

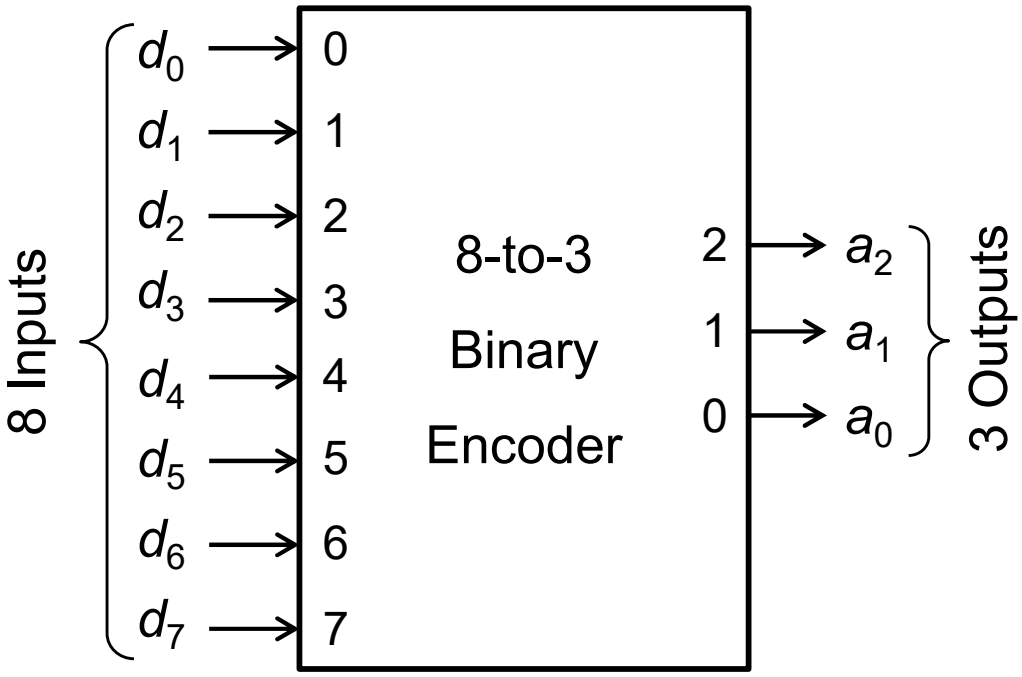
Encoders

m-to-*n* Encoder

- A combinational circuit that performs the opposite function of a decoder:
  - Performs a conversion (i.e. encoding) of an *m*-bit input code to an *n*-bit output code, where $n \leq m \leq 2^n$.
  - Has at most 2^n input lines or $m \leq 2^n$ inputs lines if not all input combinations are used.
  - Has an *n*-bit binary code as output.
  - For correct operation, requires that only a single input line is asserted at any one time (i.e. one line is 1 and all others are 0's or one input line is 0 and all others are 1's).



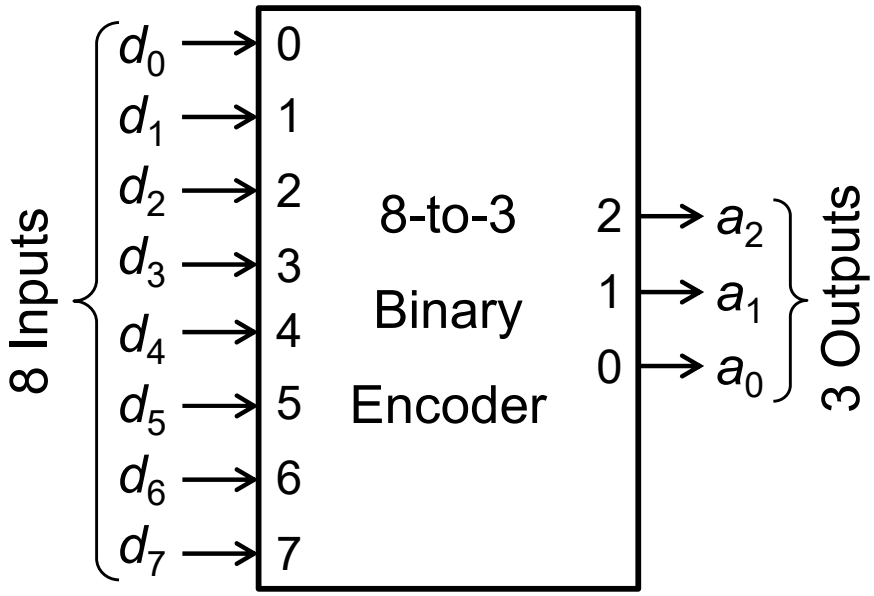
8-to-3 Binary Encoder – Truth Table



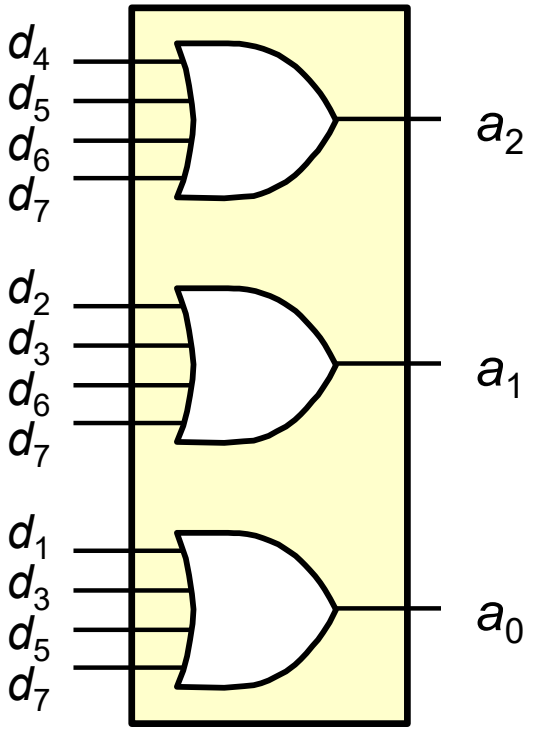
Inputs								Outputs		
d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0	a_2	a_1	a_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

- 3 outputs and 8 inputs with only a single input line asserted at any one time (i.e. one line is 1 and all others are 0's or one input line is 0 and all others are 1's).
- Encoder generates an output binary code for the active input.

8-to-3 Binary Encoder – Implementation



Inputs								Outputs		
d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0	a_2	a_1	a_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1



- A simple binary encoder can be made with just OR gates

$$a_2 = d_4 + d_5 + d_6 + d_7$$

$$a_1 = d_2 + d_3 + d_6 + d_7$$

$$a_0 = d_1 + d_3 + d_5 + d_7$$

8-to-3 Binary Encoder – Limitations

- **Two problems:**
  1) We assumed that a single input line can be active at a time (i.e. one line is 1 and all others are 0's or one input line is 0 and all others are 1's).
  2) What happens if multiple inputs are active (e.g. at 1) or if no inputs are active?
- If $d_3 = d_6 = 1$ then output **$a_2\ a_1\ a_0 = 111$** is neither corresponding to having only d_3=1 or having only d_6=1. It is something else based on equations below.
- If no input is active (e.g. all inputs are 0's) then the output $a_2\ a_1\ a_0 = 000$. This code should only be for $d_0 = 1$ and all other inputs are 0's.
- How to resolve these 2 problems?

Inputs								Outputs		
d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0	a_2	a_1	a_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

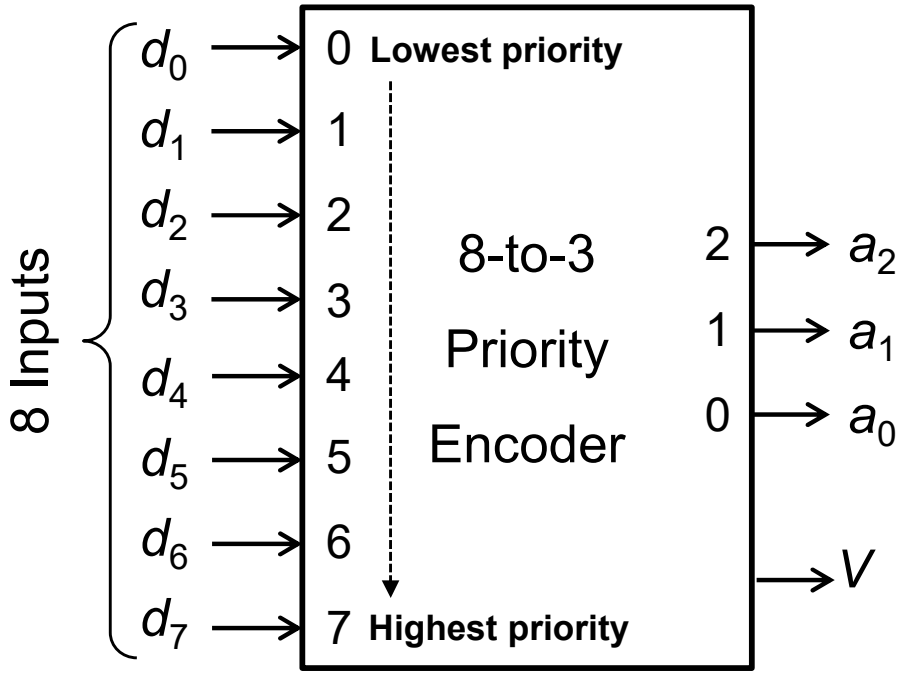
$$a_2 = d_4 + d_5 + d_6 + d_7$$

$$a_1 = d_2 + d_3 + d_6 + d_7$$

$$a_0 = d_1 + d_3 + d_5 + d_7$$

8-to-3 Priority Encoder

- A priority encoder can be used to solve the 2 problems highlighted previously:
  - We will assign priority to inputs.
  - We will only encode the highest priority active input.
  - We will introduce an extra output to indicate whether all inputs are inactive.

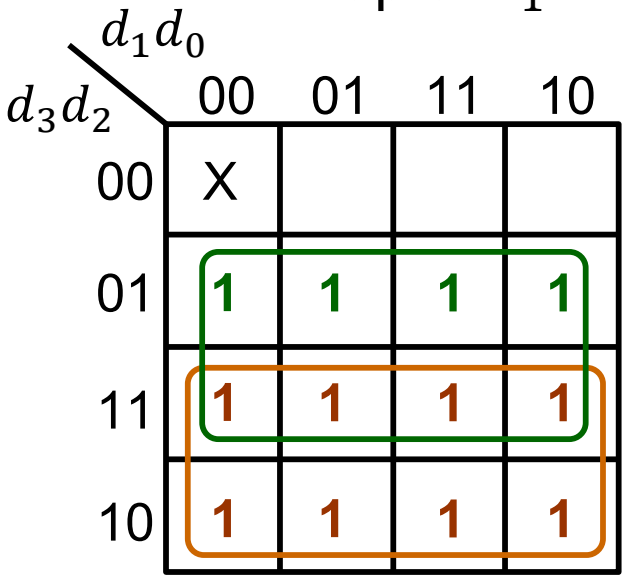


Inputs								Outputs			
d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0	a_2	a_1	a_0	V
0	0	0	0	0	0	0	0	X	X	X	0
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	1	X	0	0	1	1
0	0	0	0	0	1	X	X	0	1	0	1
0	0	0	0	1	X	X	X	0	1	1	1
0	0	0	1	X	X	X	X	1	0	0	1
0	0	1	X	X	X	X	X	1	0	1	1
0	1	X	X	X	X	X	X	1	1	0	1
1	X	X	X	X	X	X	X	1	1	1	1

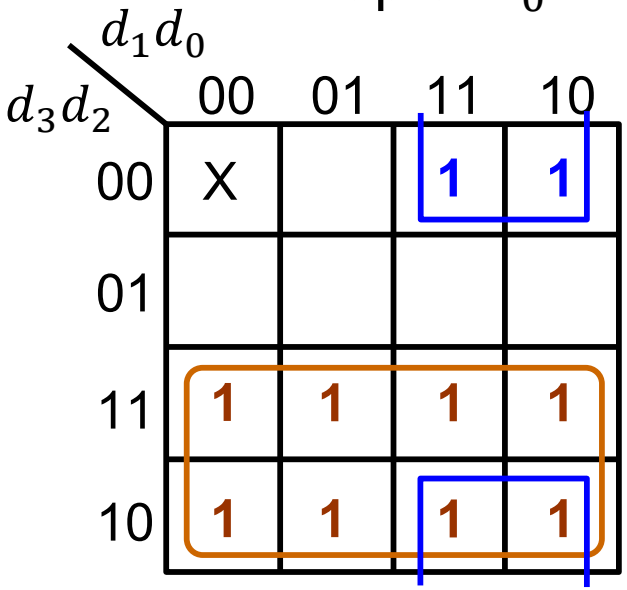
4-to-2 Priority Encoder – Implementation

Inputs				Outputs		
d_3	d_2	d_1	d_0	a_1	a_0	V
0	0	0	0	X	X	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1

K-Map of a_1



K-Map of a_0

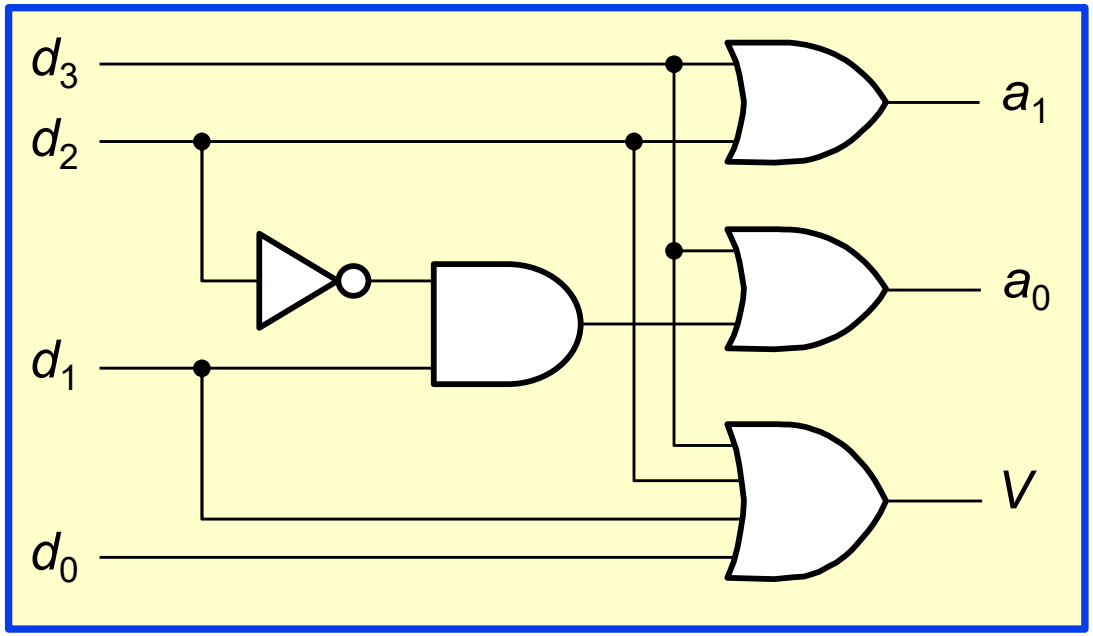


Output Expressions:

$a_1 = d_3 + d_2$

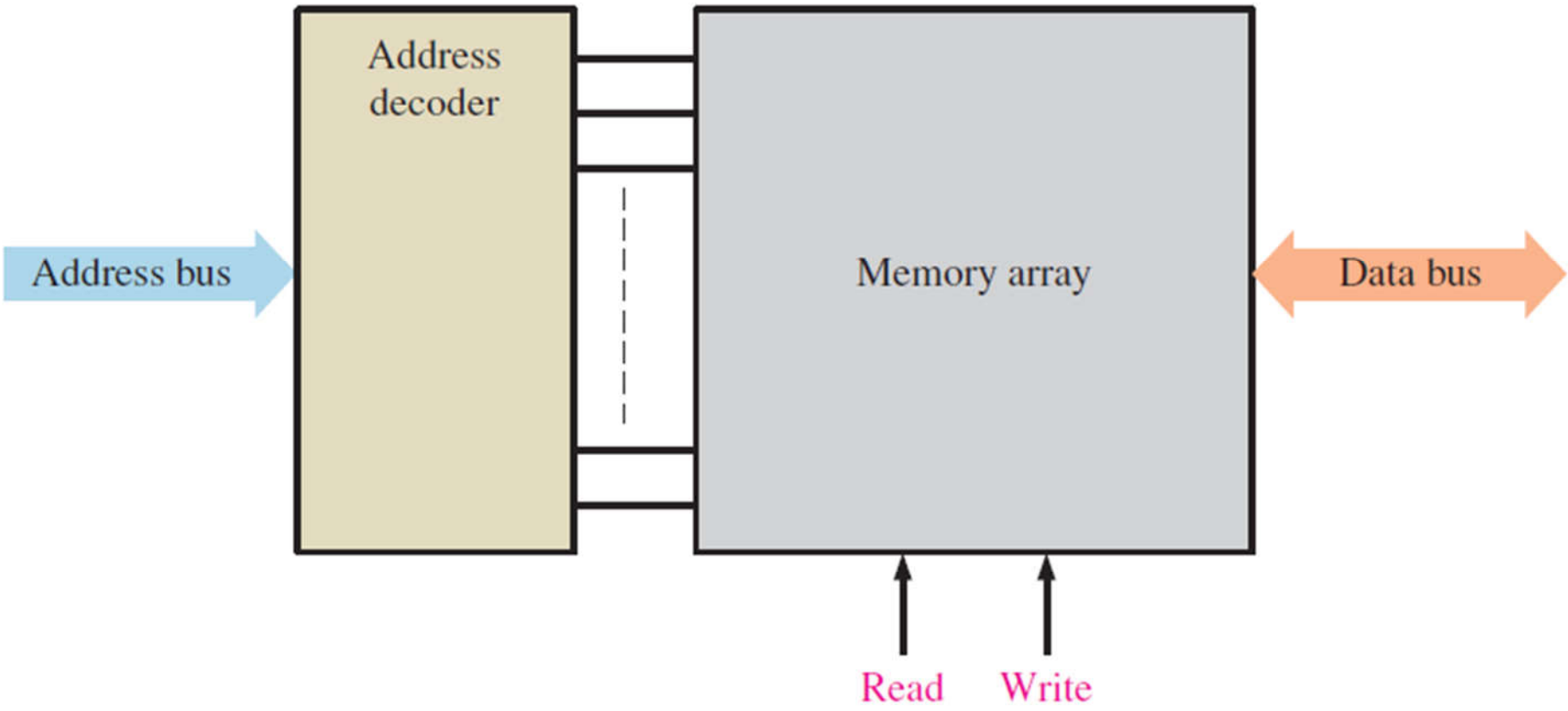
$a_0 = d_3 + d_1\, d_2'$

$V = d_3 + d_2 + d_1 + d_0$



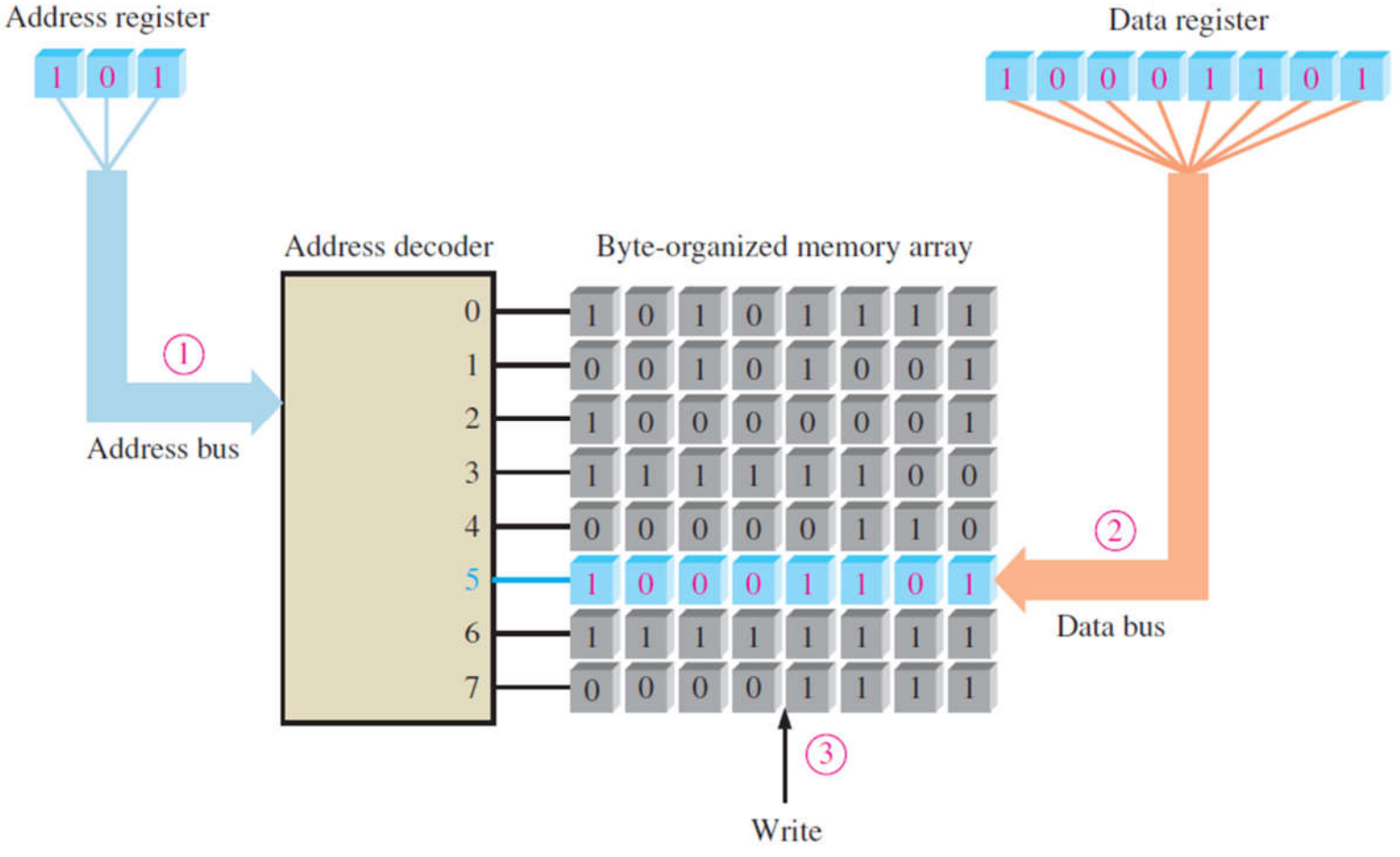
Examples of decoder applications

- Decoders are used whenever an output or a group of outputs is to be activated only on the occurrence of a specific combination of input levels.
- Decoders are widely used in memories: they respond to the input address code to activate a particular memory location.



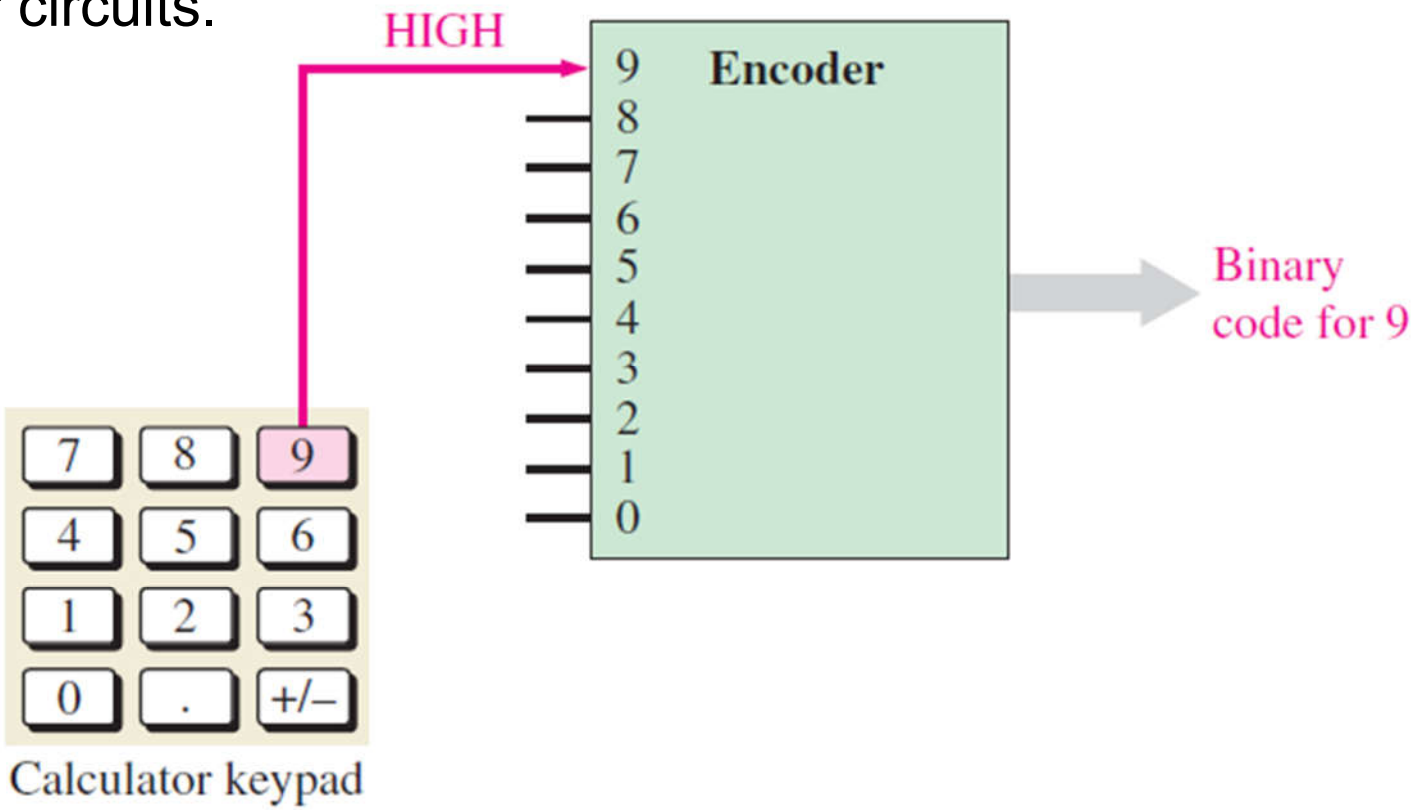
Examples of decoder applications

- The number of lines in the address bus depends on the capacity of the memory. For example, a 15-bit address code can select 32,768 locations (2^{15}) in the memory, a 16-bit address code can select 65,536 locations (2^{16}) in the memory, and so on. In personal computers a 32-bit address bus can select 4,294,967,296 locations (2^{32}), expressed as 4G.
- If you are addressing 2^N memory locations, the decoder helps reduce the number of address pins from 2^N to N.



Encoding Function

- The encoder converts information, such as a decimal number or an alphabetic character, into some coded form. For example, one certain type of encoder converts each of the decimal digits, 0 through 9, to a binary code. A HIGH level on the input corresponding to a specific decimal digit produces logic levels that represent the proper binary code on the output lines.
- Below is a simple illustration of an encoder used to convert (encode) a calculator keystroke into a binary code that can be processed by the calculator circuits.



Acknowledgments

- Credit is acknowledged where credit is due. Please refer to the full list of references.