COE 202: Digital Logic Design
Combinational Circuits
Part 3

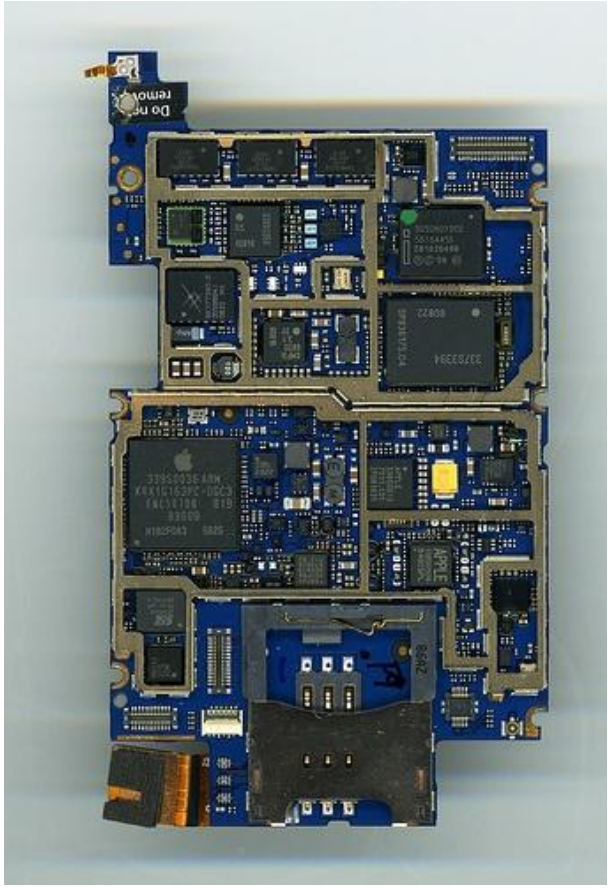
Dr. Ahmad Almulhem
Email: ahmadsm AT kfupm
Phone: 860-7554
Office: 22-324

Objectives

- Decoders
- Encoders
- Multiplexers
- DeMultiplexers

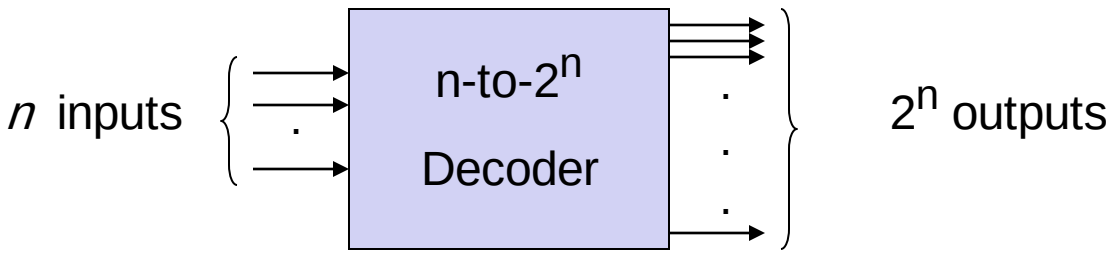
Functional Blocks

- Digital systems consists of many components (blocks)
- Useful blocks needed in many designs
  - Arithmetic blocks
  - Decoders
  - Encoders
  - Multiplexers



iPhone motherboard (torontophonerepair.com)

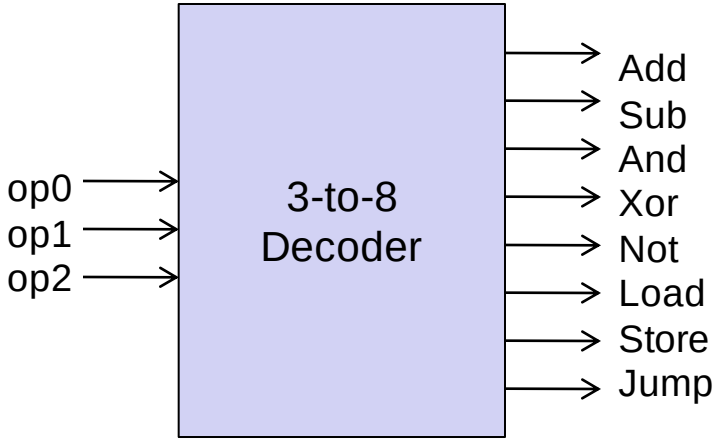
Decoder



- Information is represented by binary codes
- **Decoding** - the conversion of an *n*-bit input code to an *m*-bit output code with n <= m <= 2^n such that each valid code word produces a unique output code
- Circuits that perform decoding are called **decoders**
- A decoder is a minterm generator

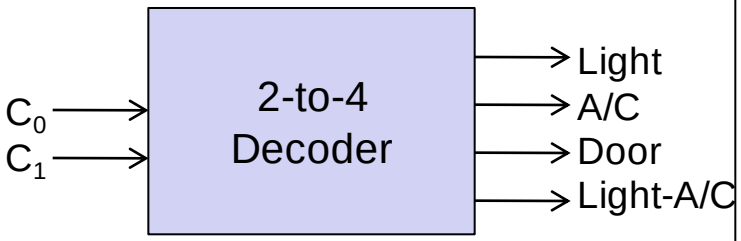
Decoder (Uses)

Decode a 3-bit op-codes:



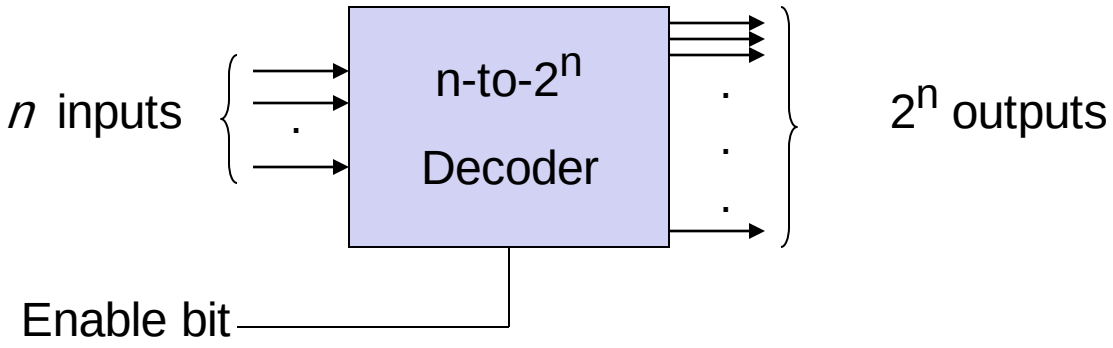
Load a
Add b
Store c
.
.

Home automation:



Decoder with Enable

- A decoder can have an additional input signal called the enable which enables or disables the output generated by the decoder



2-to-4 Decoder

A 2-to-4 Decoder

- 2 inputs (A_1, A_0)
- $2^2 = 4$ outputs (D_3, D_2, D_1, D_0)

2-to-4 Decoder

A 2-to-4 Decoder

2 inputs (A_1, A_0)

$2^2 = 4$ outputs (D_3, D_2, D_1, D_0)

Truth Table

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

2-to-4 Decoder

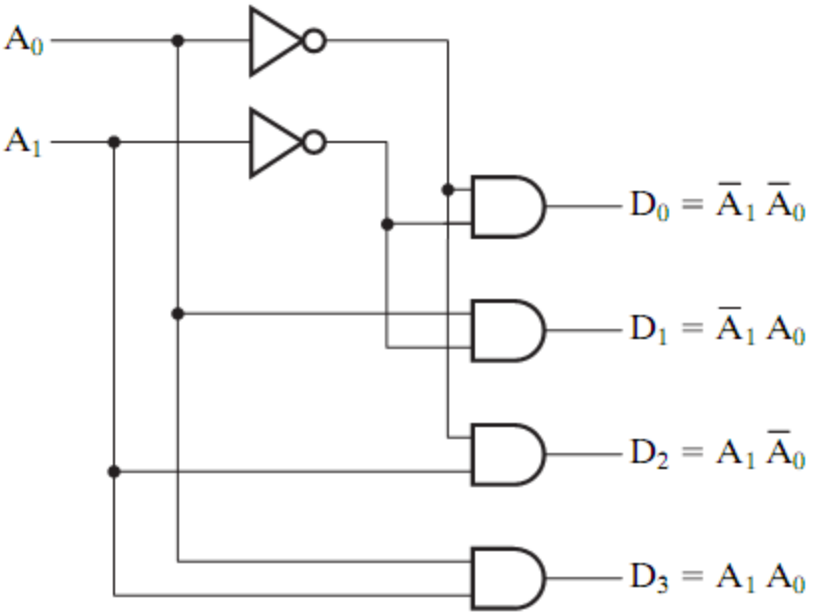
A 2-to-4 Decoder

2 inputs (A_1, A_0)

$2^2 = 4$ outputs (D_3, D_2, D_1, D_0)

Truth Table

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



Src: Mano's book

2-to-4 Decoder with Enable

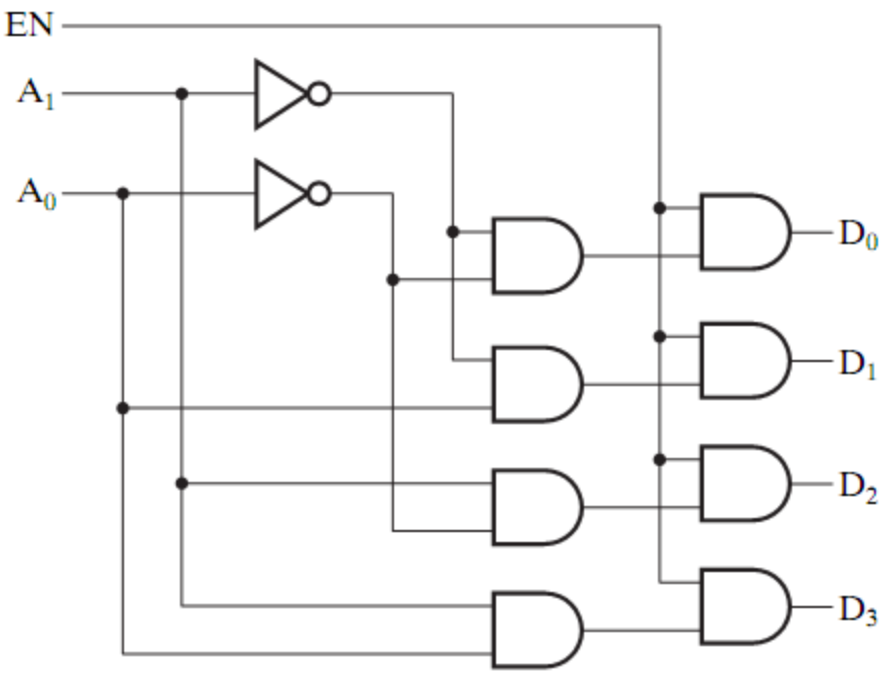
Truth Table

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

2-to-4 Decoder with Enable

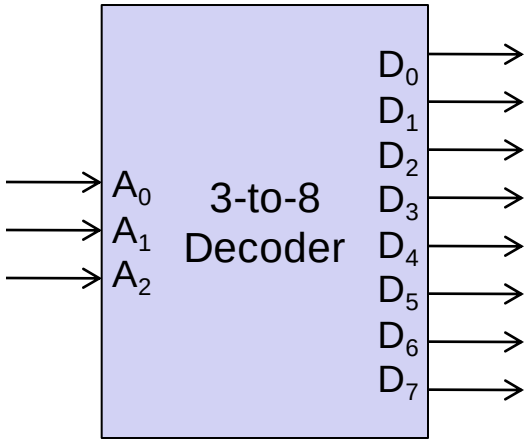
Truth Table

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

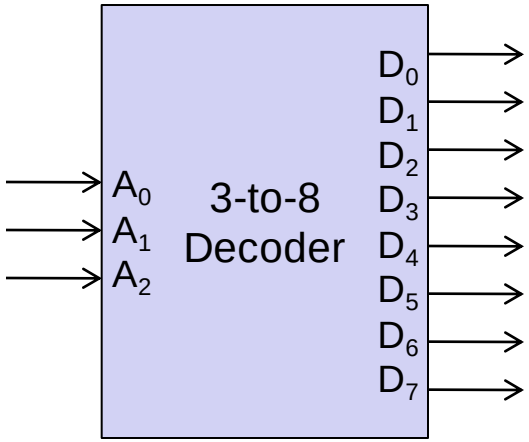


Src: Mano's book

3-to-8 Decoder

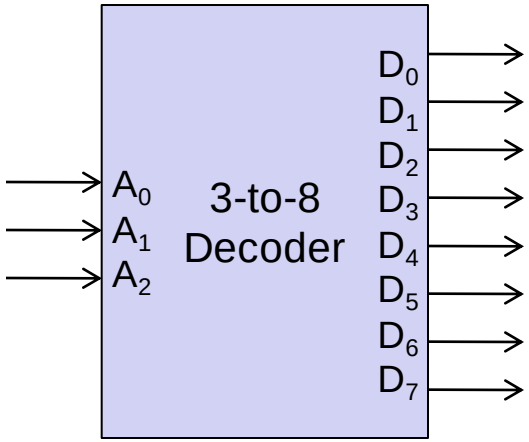


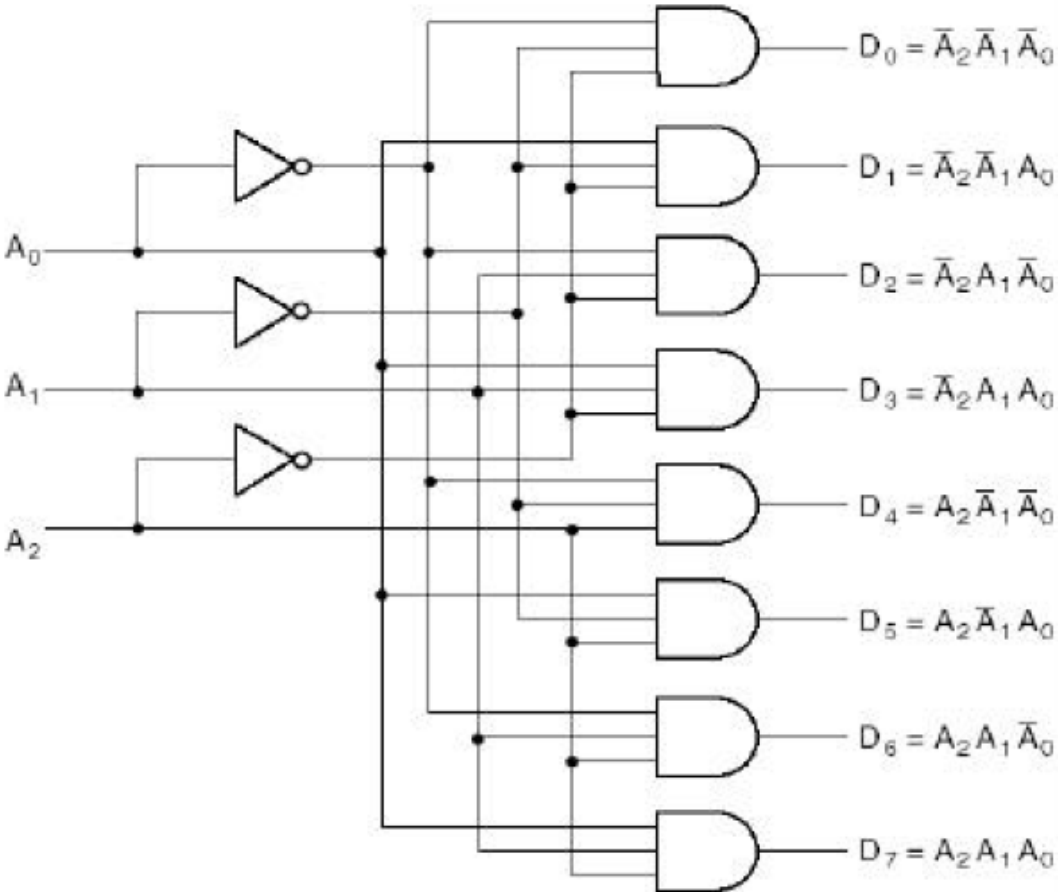
3-to-8 Decoder



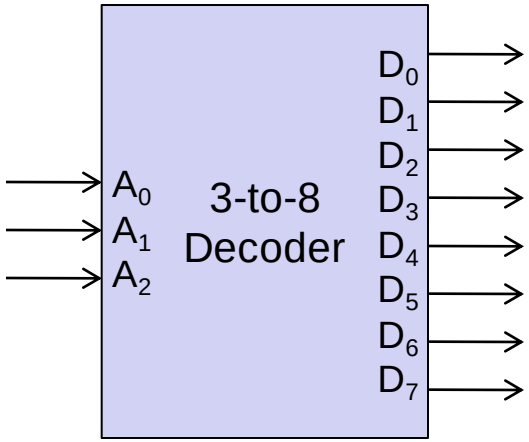
A_2	A_1	A_0	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

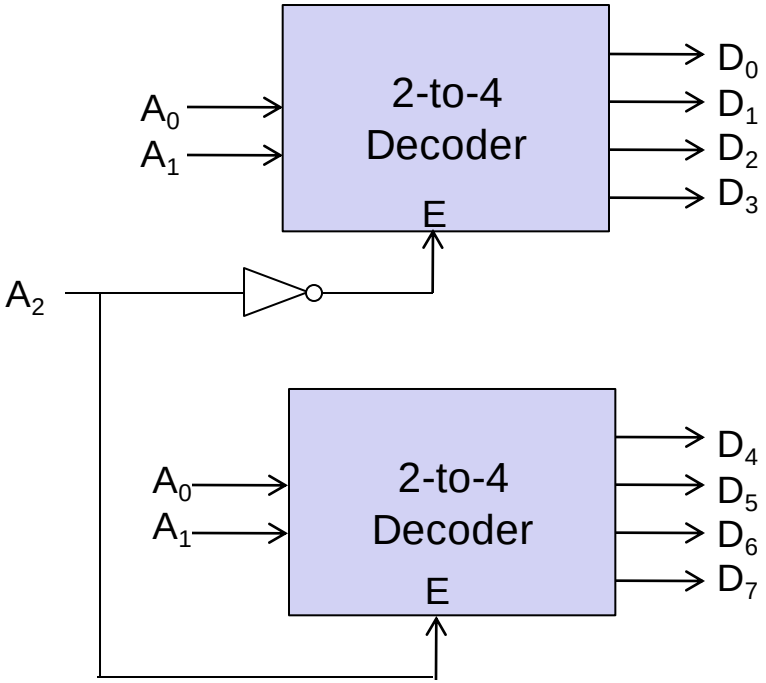
3-to-8 Decoder





3-to-8 Decoder (using 2 2-to-4 decoders)





Decoder-Based Combinational Circuits

- A Decoder generates all the minterms
- A boolean function can be expressed as a sum of minterms
- Any boolean function can be implemented using a decoder and an OR gate.
- Note: The Boolean function must be represented in terms of its minterms and not its minimized form

Decoder-Based Combinational Circuits (Example)

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

$S = \sum m\ (1,2,4,7)$

$C = \sum m\ (3,5,6,7)$

3 inputs and 8 possible minterms
3-to-8 decoder can be used for implementing this circuit

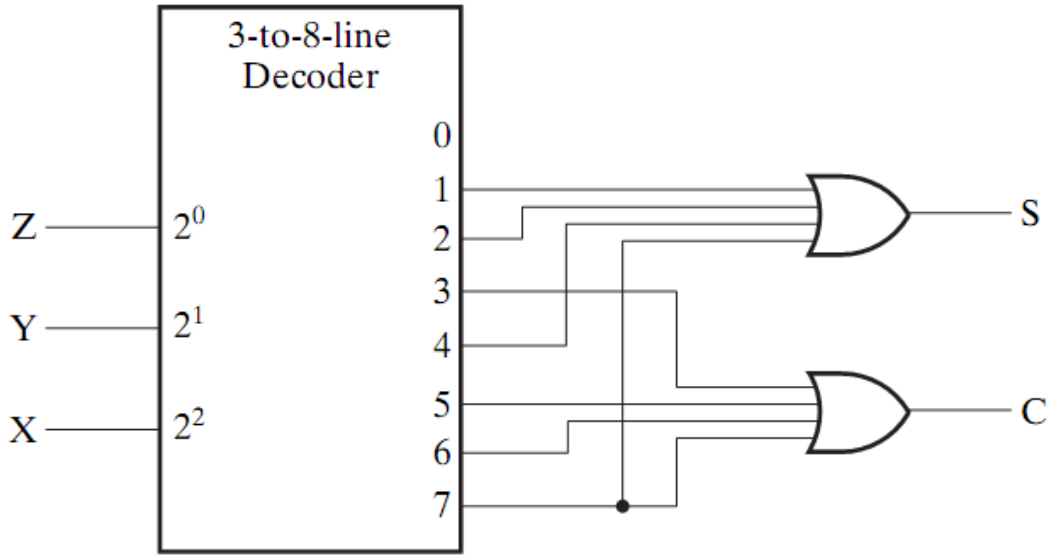
Decoder-Based Combinational Circuits (Example)

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

$S = \sum m\ (1,2,4,7)$

$C = \sum m\ (3,5,6,7)$

3 inputs and 8 possible minterms
3-to-8 decoder can be used for implementing this circuit

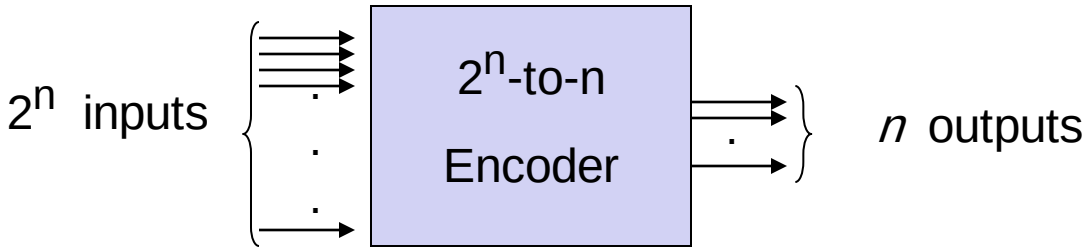


Src: Mano's book

Decoder-Based Combinational Circuits (Summary)

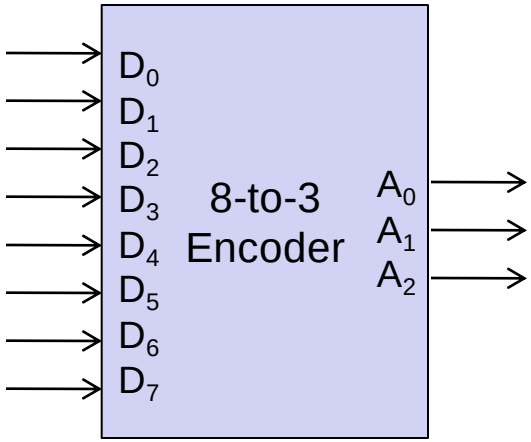
- Good if:
  - Many output functions with same inputs
  - Each output has few minterms
- Hint:
  - Check if the function complement has fewer minterms and use NOR instead of OR.

Encoder



- **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.

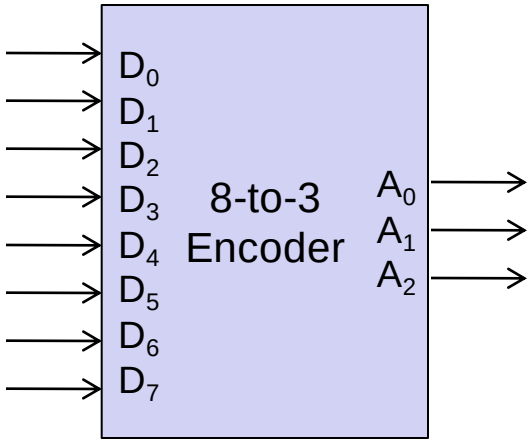
8-to-3 Encoder



Description:

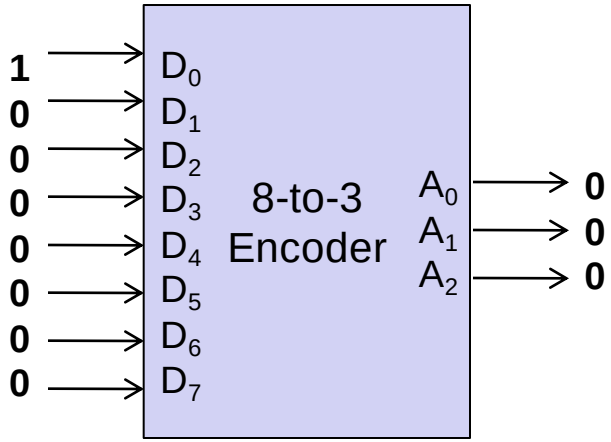
- 2^3 = 8 inputs, 3 outputs
- one input =1, others = 0's
- Each input generate unique binary code

8-to-3 Encoder (truth table)



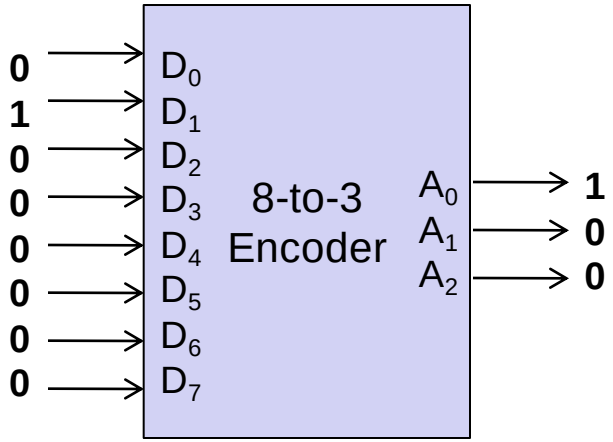
inputs								outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

8-to-3 Encoder (truth table)



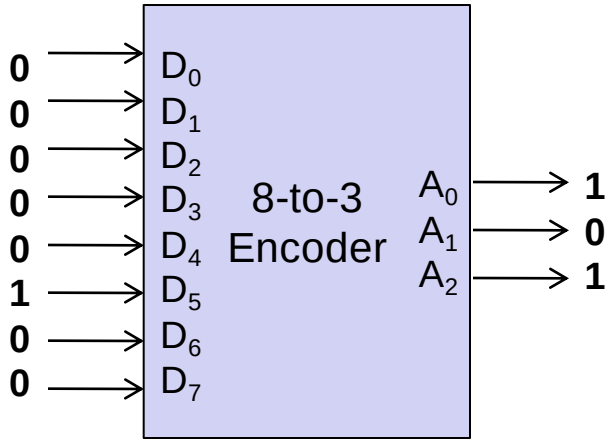
inputs								outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

8-to-3 Encoder (truth table)



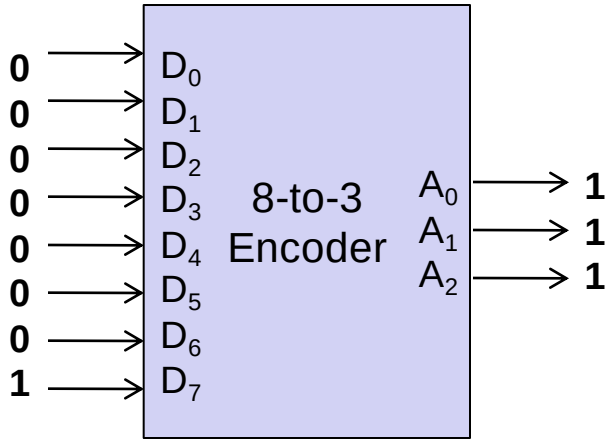
inputs								outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

8-to-3 Encoder (truth table)



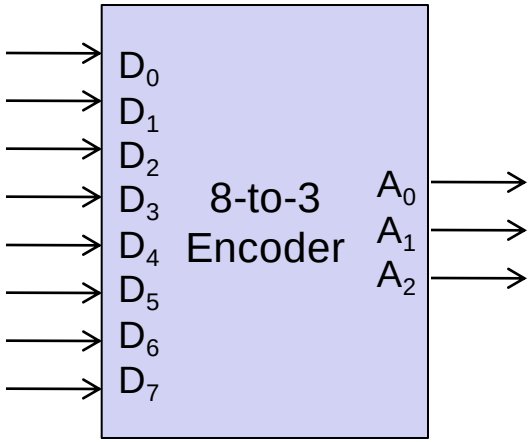
inputs								outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

8-to-3 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

8-to-3 Encoder (equations)



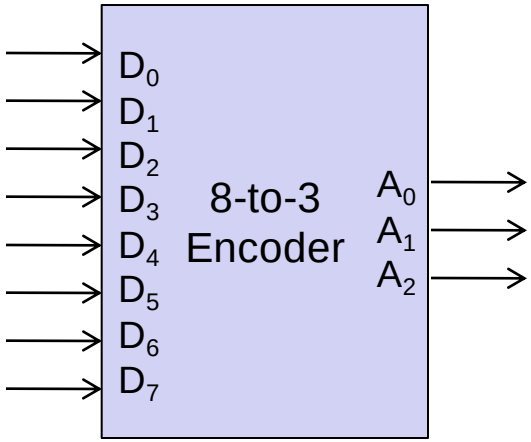
inputs								outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

Output equations:

A_0 = ?
A_1 = ?
A_2 = ?

Note: This truth table is not complete! Why?

8-to-3 Encoder (equations)



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

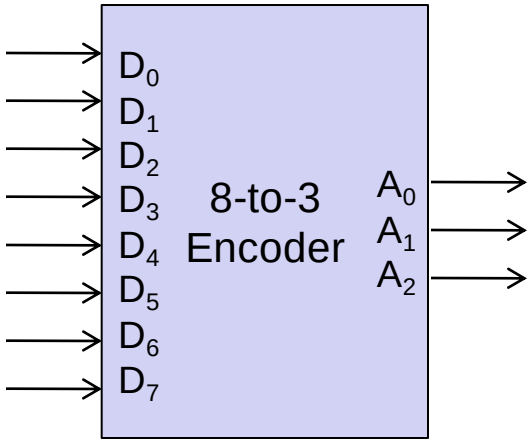
Output equations:

$A_0 = D_1 + D_3 + D_5 + D_7$

$A_1 = ?$

$A_2 = ?$

8-to-3 Encoder (equations)



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

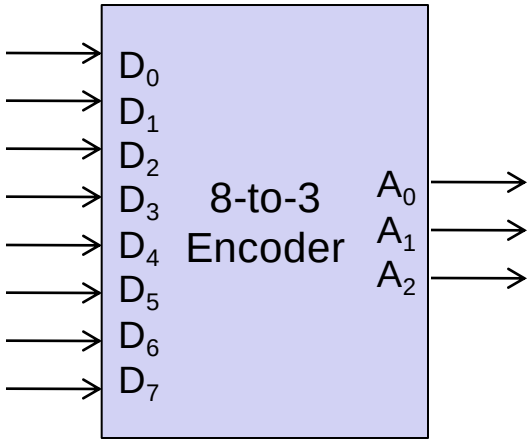
Output equations:

$A_0 = D_1 + D_3 + D_5 + D_7$

$A_1 = D_2 + D_3 + D_6 + D_7$

$A_2 = ?$

8-to-3 Encoder (equations)

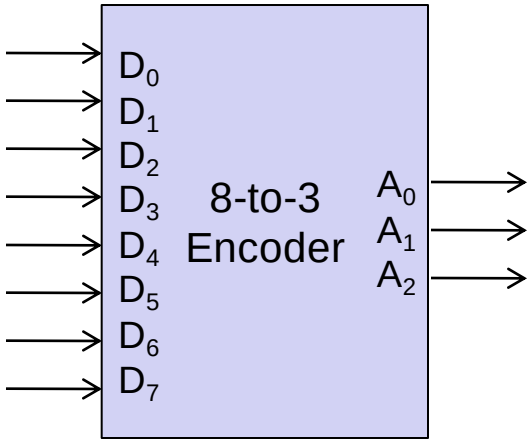


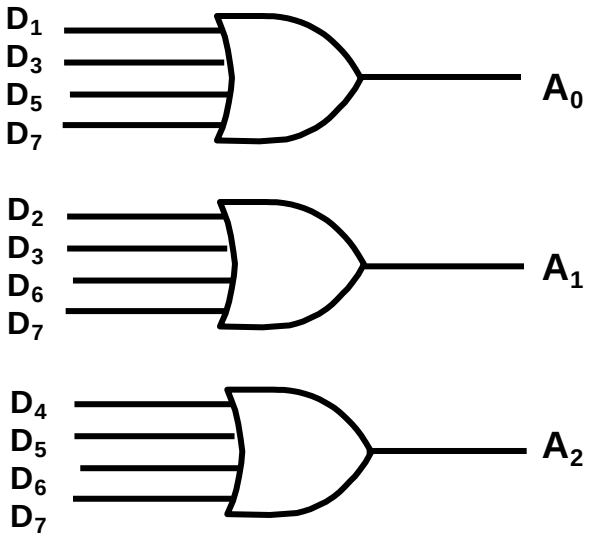
inputs								outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

Output equations:

$A_0 = D_1 + D_3 + D_5 + D_7$
$A_1 = D_2 + D_3 + D_6 + D_7$
$A_2 = D_4 + D_5 + D_6 + D_7$

8-to-3 Encoder (circuit)





Output equations:

$A_0 = D_1 + D_3 + D_5 + D_7$
$A_1 = D_2 + D_3 + D_6 + D_7$
$A_2 = D_4 + D_5 + D_6 + D_7$

8-to-3 Encoder (limitations)

Two Limitations:

1. Two or more inputs = 1
   - Example: $D_3 = D_6 = 1$
   - $A_2A_1A_0 = 111$ ✗
2. All inputs = 0
   - Same as $D_0 = 1$

Output equations:

$A_0 = D_1 + D_3 + D_5 + D_7$
$A_1 = D_2 + D_3 + D_6 + D_7$
$A_2 = D_4 + D_5 + D_6 + D_7$

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

Priority Encoder

- Address the previous two limitations

1. Two or more inputs = 1

   Consider the bit with highest priority

2. All inputs = 0

   Add another output v to indicate this combination

4-to-2 Priority Encoder

Description:

- $2^2 = 4$ inputs, 2 + 1 outputs
- Two or more 1's take highest priority

4-to-2 Priority Encoder

Description:

- $2^2 = 4$ inputs, 2 + 1 outputs
- Two or more 1's take highest priority

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

This is a condensed truth table!
It has only 5 rows instead of 16!

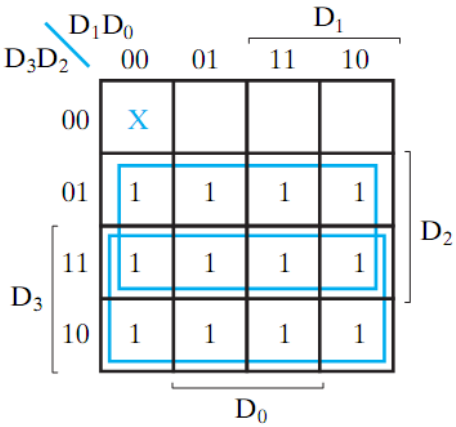
Row 3 = 2 combinations
Row 4 = 4 combinations
Row 5 = 8 combinations

4-to-2 Priority Encoder

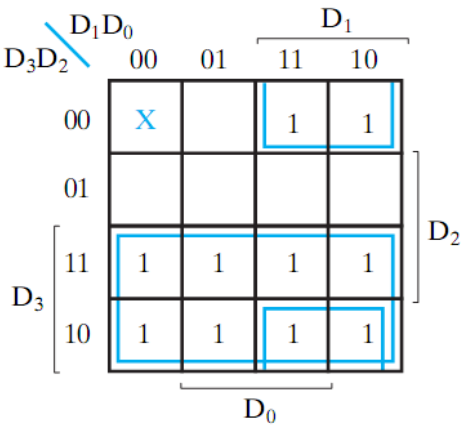
Description:

- $2^2 = 4$ inputs, 2 + 1 outputs
- Two or more 1's take highest priority

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



$A_1 = D_2 + D_3$



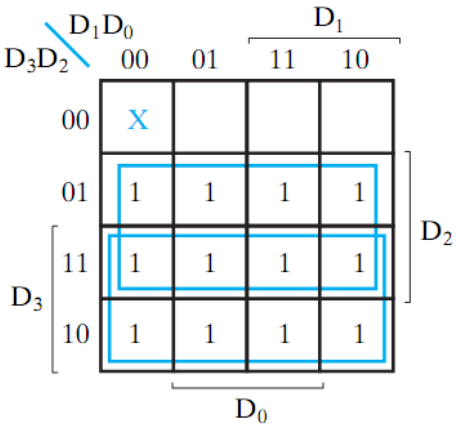
$A_0 = D_3 + D_1\overline{D}_2$

4-to-2 Priority Encoder

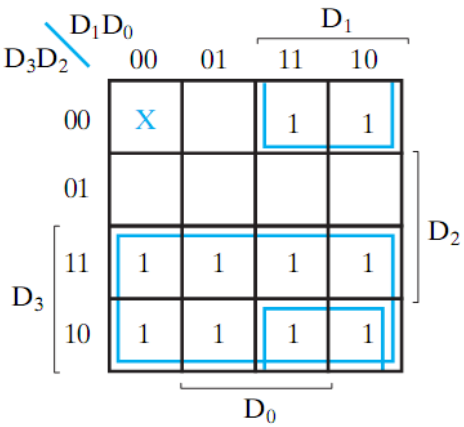
Description:

- $2^2 = 4$ inputs, 2 + 1 outputs
- Two or more 1's take highest priority

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



$A_1 = D_2 + D_3$



$A_0 = D_3 + D_1\overline{D}_2$

Equations:

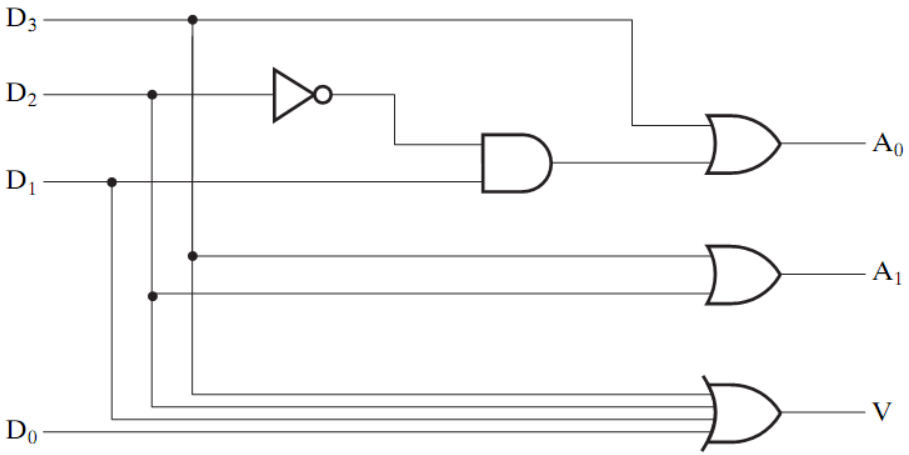
$$A_0 = D_3 + D_1 D_2'$$
$$A_1 = D_2 + D_3$$
$$V = D_0 + D_1 + D_2 + D_3$$

4-to-2 Priority Encoder

Description:

- $2^2 = 4$ inputs, 2 + 1 outputs
- Two or more 1's take highest priority

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

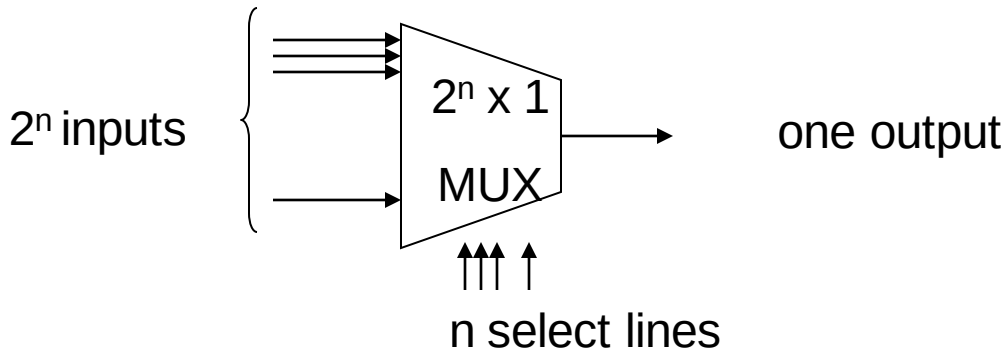


Equations:

$$A_0 = D_3 + D_1 D_2'$$
$$A_1 = D_2 + D_3$$
$$V = D_0 + D_1 + D_2 + D_3$$

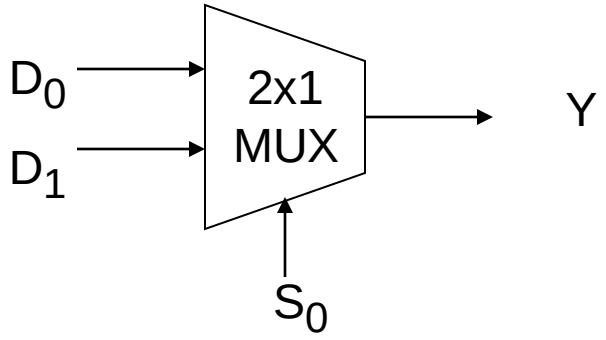
Multiplexers

- Is a combinational circuit
- Has a single output
- Directs one of 2^n input to the output
- Input to output direction is done based on a set of n select bits



2x1 MUX

- A 2x1 multiplexer (MUX) will have 2 input lines and 1 select line



- $Y=D_0$ for $S_0=0$, and $Y=D_1$ for $S_0=1$
- Minimizing will result in: $Y = S_0'.D_0 + S_0.D_1$

4x1 MUX

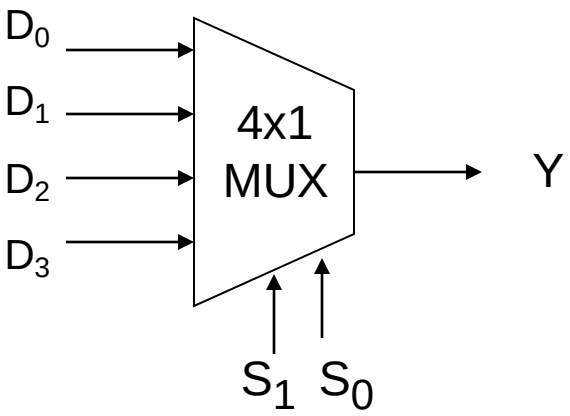
- A 4x1 MUX will have 4 input lines (D_0, D_1, D_2, D_3) and 1 output Y with 2 Select Lines (S_0, S_1)
- The output for different select values is defined as:

  $S_0S_1 = 00$, $Y = D_0$

  $S_0S_1 = 01$, $Y = D_1$

  $S_0S_1 = 10$, $Y = D_2$

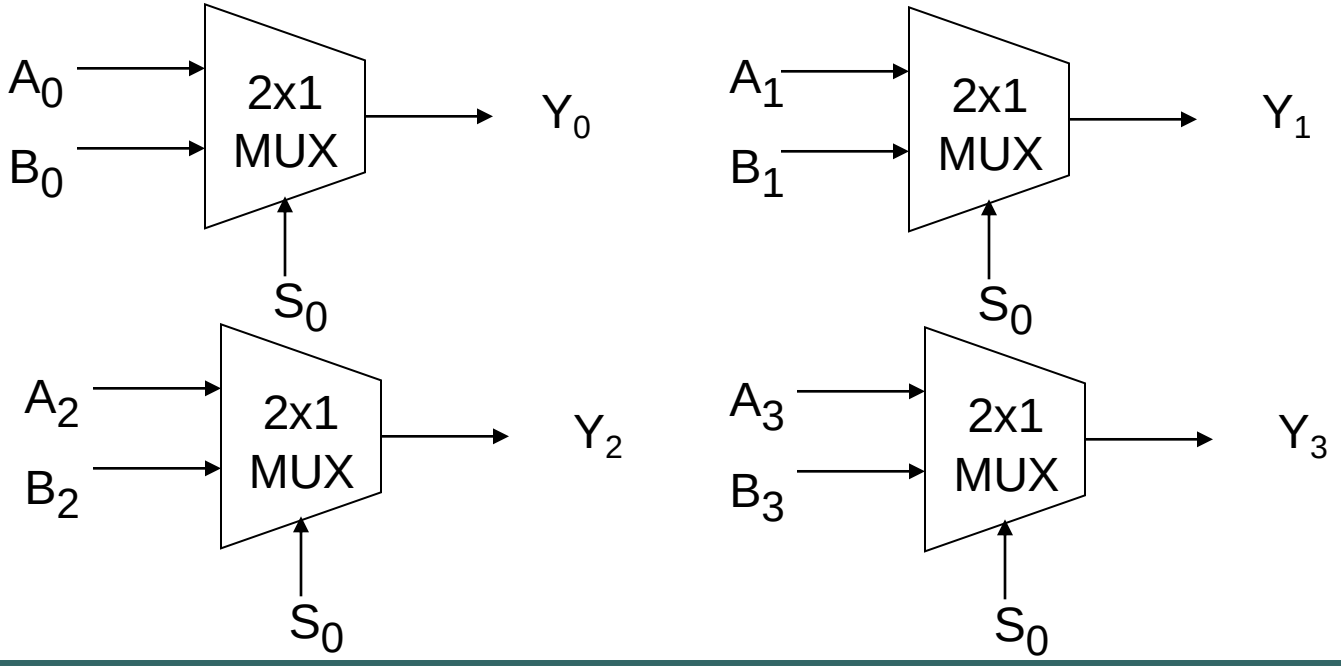
  $S_0S_1 = 11$, $Y = D_3$



- $Y = \overline{S_1}\,\overline{S_0}D_0 + \overline{S_1}S_0D_1 + S_1\overline{S_0}D_2 + S_1S_0D_3$
- The output Y depends on the minterms of the Select lines

Quad 2x1 MUX

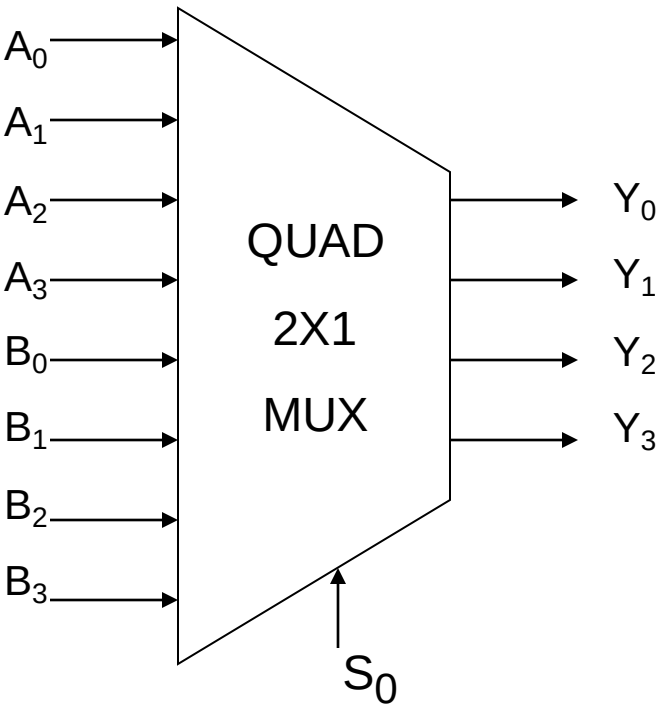
- Does multiplexing of two 4-bit numbers.
- Has a 4-bit output and a single select line
- Is built using four 2x1 MUXes



Quad 2x1 MUX

$Y = A$ If $S_0 = 0$

$Y = B$ if $S_0=1$



MUX-based Design

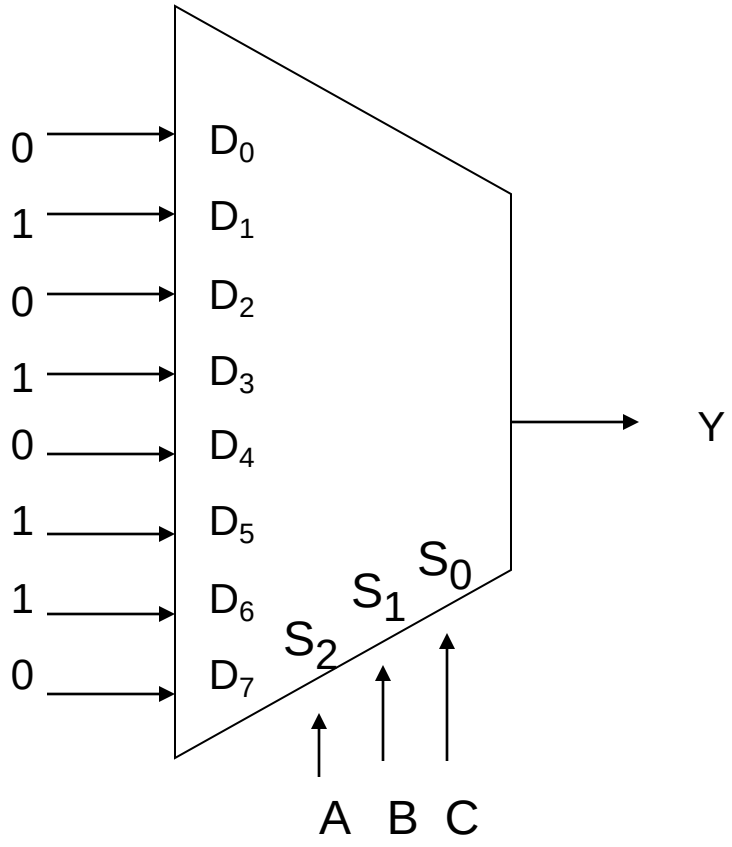
- A MUX can be used to implement any function expressed using its minterms

Example: Implement $F(A,B,C)=\sum(1,3,5,6)$ using MUXes

Solution1:

We can use a MUX with the number of select lines equal to the number of input variables of the function. Since this function has 3 input variables, it will require 3 select lines, i.e. an 8x1 MUX

MUX-based Design (n-Select lines)



F(A,B,C)=∑(1,3,5,6)

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

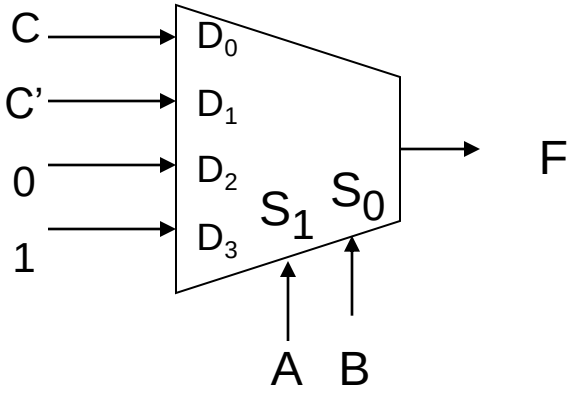
MUX-based Design (n-1 Select lines)

- Implement the function F(A,B,C) =∑(1,3,5,6)
- We will use 2 select lines instead of the 3 required for the three input variables
  - A => S1, B=> S0
- The third variable C and its complement will serve as two of the inputs to the MUX

MUX-based Design (n-1 Select lines)

A	B	C	F	
0	0	0	0	F = C
0	0	1	1	
0	1	0	1	F = C’
0	1	1	0	
1	0	0	0	F = 0
1	0	1	0	
1	1	0	1	F = 1
1	1	1	1	

F(A,B,C)=∑(1,3,5,6)



Another Example

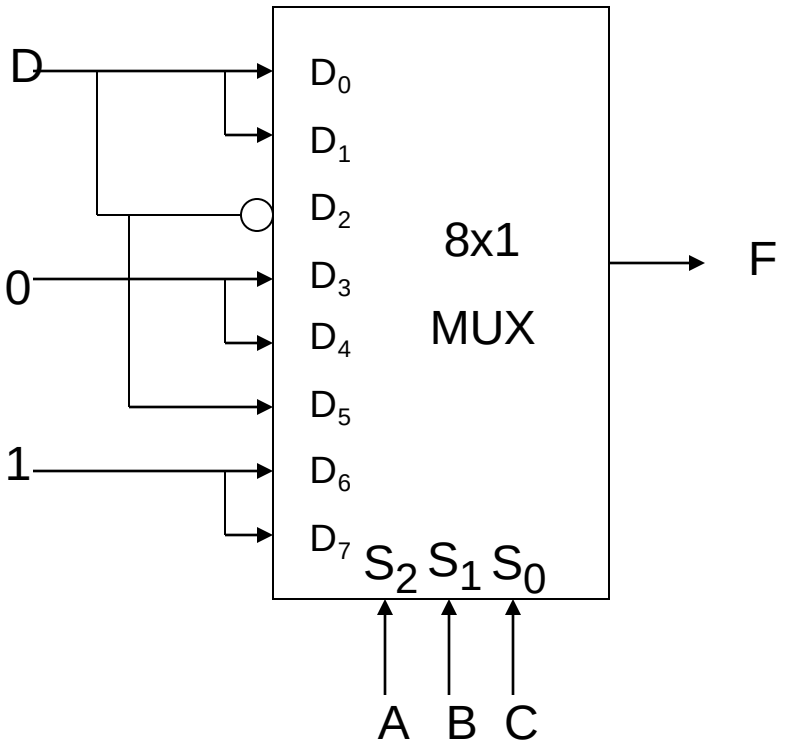
Implement the function

$F(A,B,C,D)=\sum(1,3,4,11,12,13,14,15)$

We can implement this function with 3 Select lines => an 8x1 MUX is required

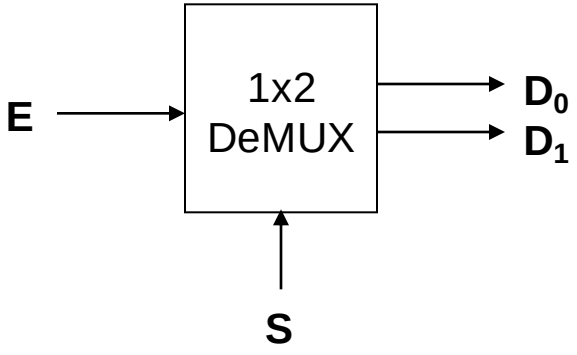
Example

A	B	C	D	F	
0	0	0	0	0	F = D
0	0	0	1	1	
0	0	1	0	0	F = D
0	0	1	1	1	
0	1	0	0	1	F = D’
0	1	0	1	0	
0	1	1	0	0	F = 0
0	1	1	1	0	
1	0	0	0	0	F = 0
1	0	0	1	0	
1	0	1	0	0	F = D
1	0	1	1	1	
1	1	0	0	1	F = 1
1	1	0	1	1	
1	1	1	0	1	F = 1
1	1	1	1	1	

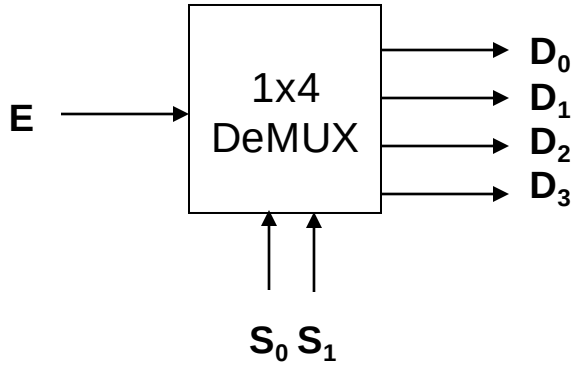


DeMultiplexer

- Performs the inverse of the operation of a MUX
- It has one input line, the input from which is transmitted to one of 2n output lines
- The output lines are selected based on the select inputs



1x4 DeMUX



The circuit has an input E, the outputs are given by:

$D_0 = E$, if $S_0S_1=00$ $\quad$ $D_0 = S_1'S_0' E$

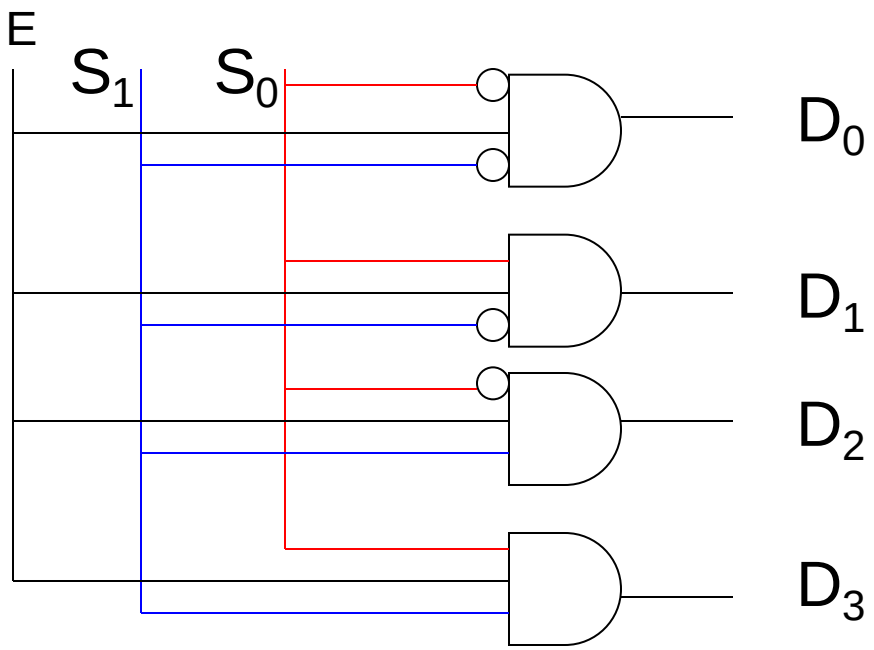
$D_1 = E$, if $S_0S_1=01$ $\quad$ $D_1 = S_1'S_0 E$

$D_2 = E$, if $S_0S_1=10$ $\quad$ $D_2 = S_1S_0' E$

$D_3 = E$, if $S_0S_1=11$ $\quad$ $D_3 = S_1S_0 E$

DeMUX/Decoder

- The circuit diagram of a 1x4 DeMUX resembles a 2x4 Decoder with an Enable input
- For the decoder, the inputs are A_1 and A_0, and the signal E is the Enable
- For the DeMUX, the inputs S_1 and S_0 are the select inputs and E is the actual input



Summary

- Useful Functional Blocks
  - Decoders
  - Encoders
  - Multiplexers
  - DeMultiplexers
- Can be used to build bigger systems