Digital Logic Design

Combinational Logic Part (3)

Half Adder

Design an Adder for 1-bit numbers?

1. Specification:

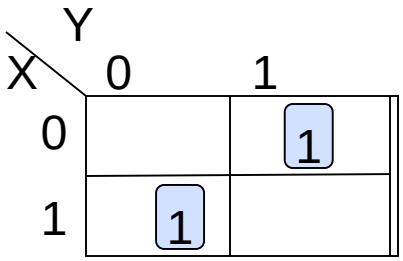
2 inputs (X,Y)

2 outputs (C,S)

2. Formulation:

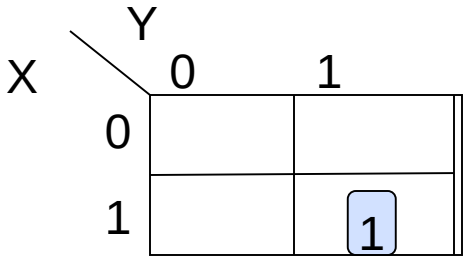
X	Y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

3. Optimization/Circuit



S = YX'+XY'

= X ⊕ Y



C = XY

Half Adder

Design an Adder for 1-bit numbers?

1. Specification:

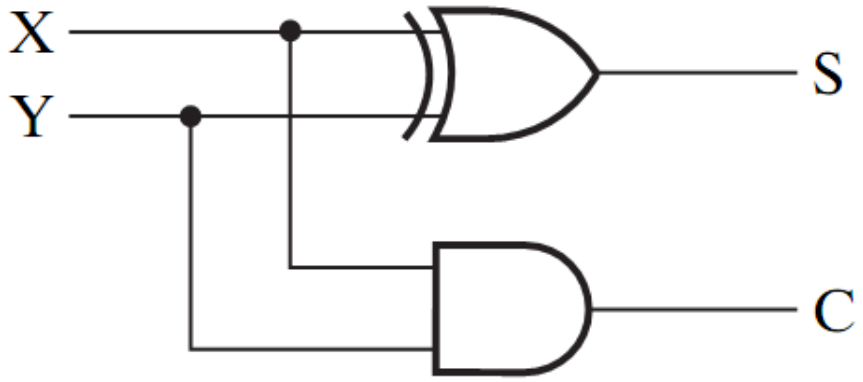
2 inputs (X,Y)

2 outputs (C,S)

2. Formulation:

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

3. Optimization/Circuit

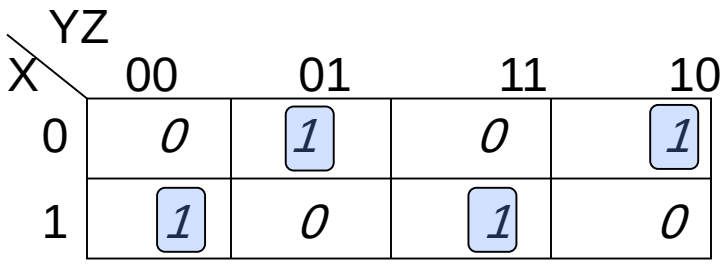


Full Adder

A combinational circuit that adds 3 input bits to generate a Sum bit and a Carry bit

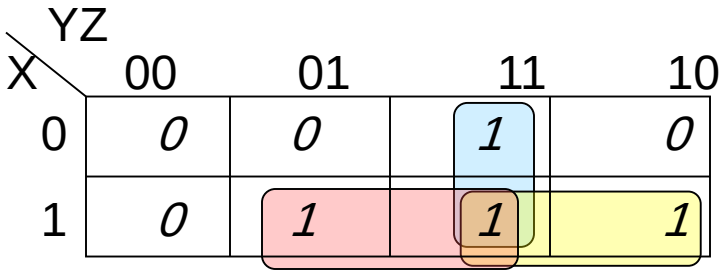
X	Y	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Sum



S = X'Y'Z + X'YZ' + XY'Z' +XYZ

Carry



C = XY + YZ + XZ

Full Adder = can be implemented with 2 Half Adders

Manipulating the Equations:

$$S = X'Y'Z + X'YZ' + XY'Z' + XYZ$$

$$= (X'Y' + XY)Z + (X'Y + XY')Z'$$

$$= (X \oplus Y)'Z + (X \oplus Y)Z'$$

$$= (X \oplus Y) \oplus Z$$

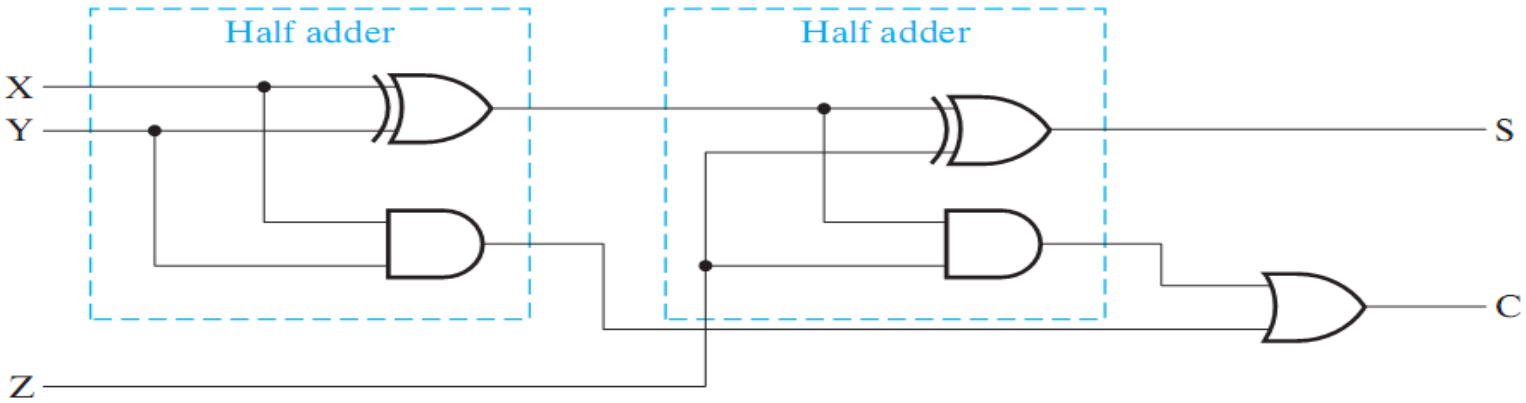
$$C = XY + XZ + YZ = XY + XZ(Y + Y') + YZ(X + X')$$

$$= XY + XYZ + XY'Z + X'YZ + XYZ$$

$$= XY(1 + Z) + Z(XY' + X'Y)$$

$$= XY + Z(X \oplus Y)$$

Think of Z as a carry in



Bigger Adders

- How to build an adder for n-bit numbers?
  - Example: 4-Bit Adder
    - Inputs ?

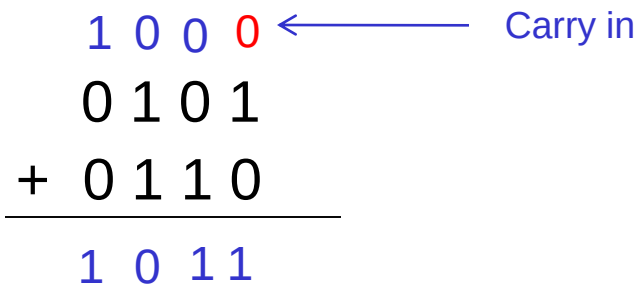
      9 inputs
    - Outputs ?

      5 outputs
    - What is the size of the truth table?

      512 rows!
    - How many functions to optimize?

      5 functions

Binary Parallel Adder



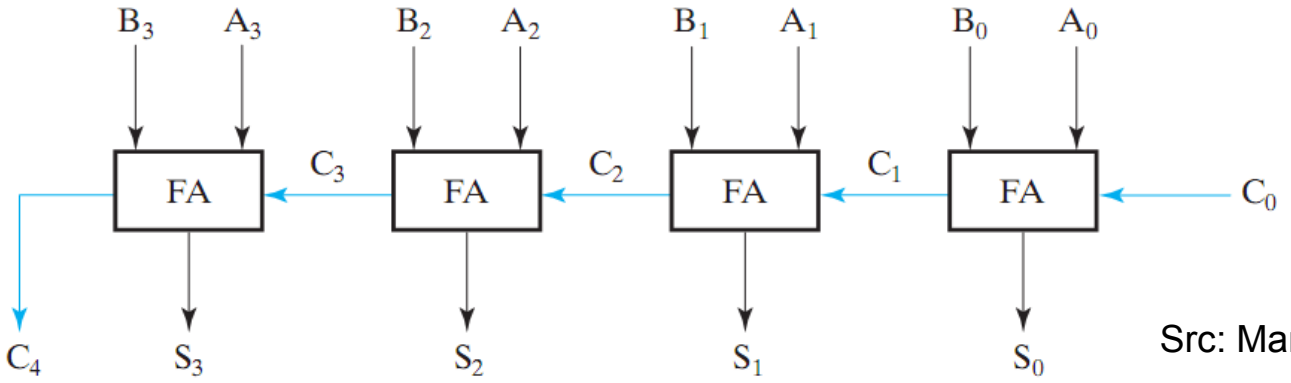
To add n-bit numbers:

- Use n Full-Adders in parallel
- The carries propagates as in addition by hand
- This is an example of a **hierarchical design**
  - The circuit is broken into small blocks

Binary Parallel Adder

To add n-bit numbers:

- Use n Full-Adders in parallel
- The carries propagates from stage to stage (delay)

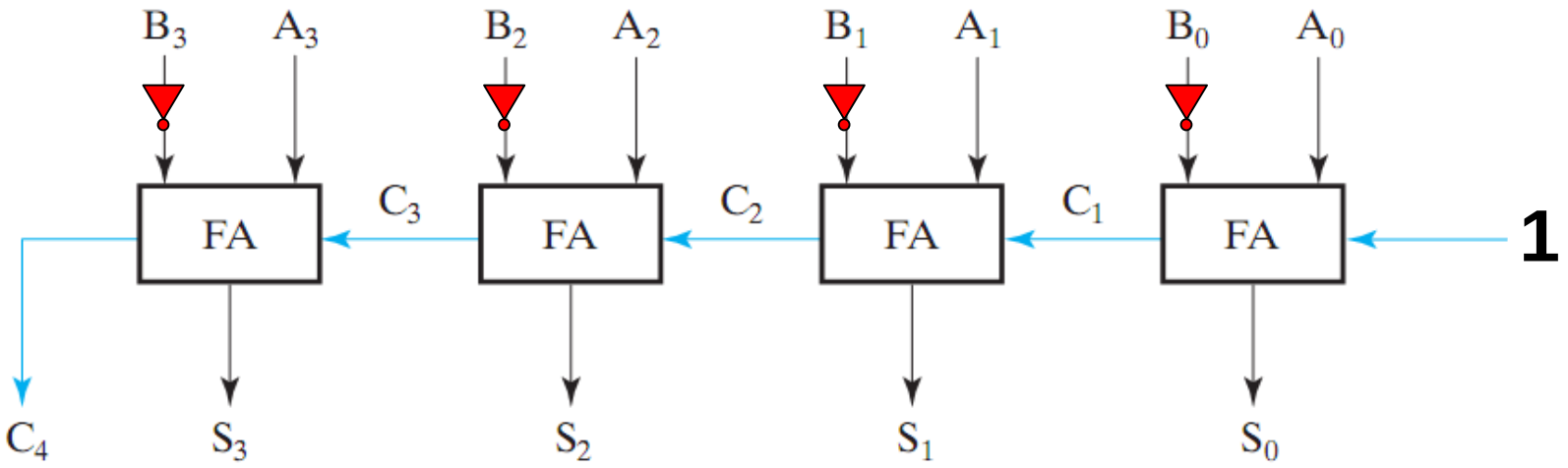


Src: Mano's Book

This adder is called ***ripple carry adder***

Subtraction (2's Complement)

How to build a subtractor using 2's complement?



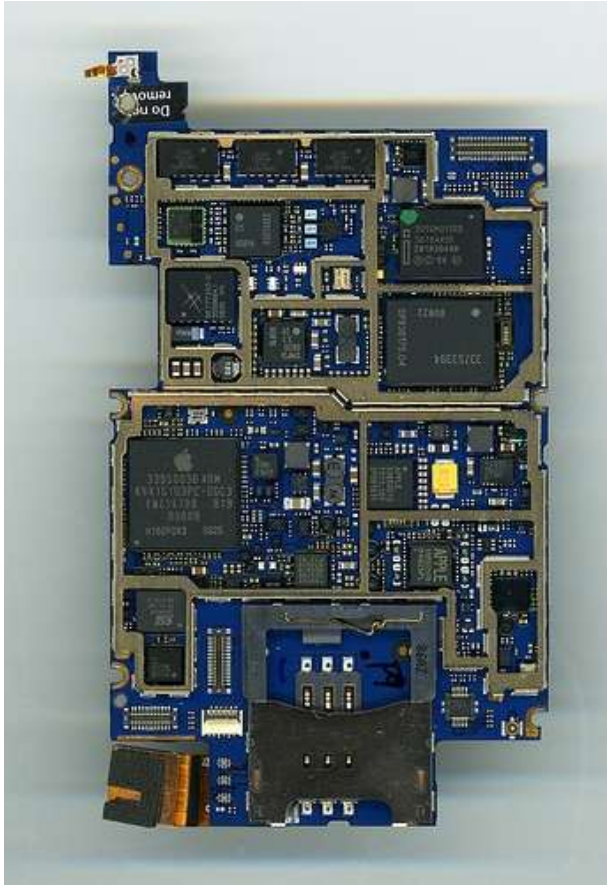
Src: Mano's Book

S = A + (-B)

Functional Blocks

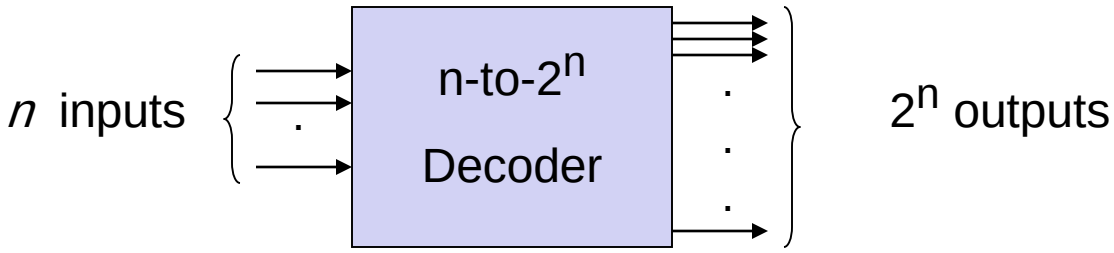
- Digital systems consists of many components (blocks)
- Useful blocks needed in many designs
  - Arithmetic blocks
  - Decoders
  - Encoders
  - Multiplexers

Examples of MSI devices



iPhone motherboard (torontophonerepair.com)

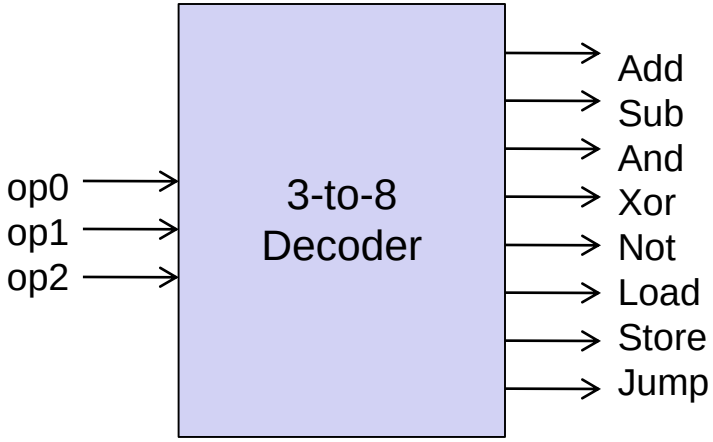
Decoder



- 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

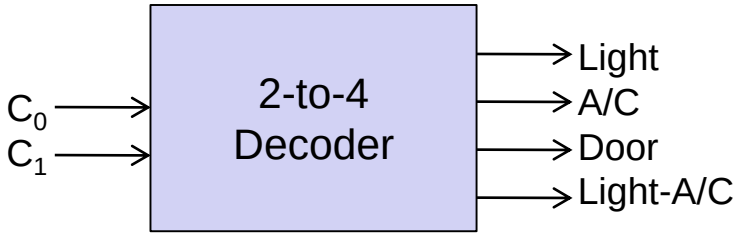
Decoder (Uses)

Decode a 3-bit op-codes:



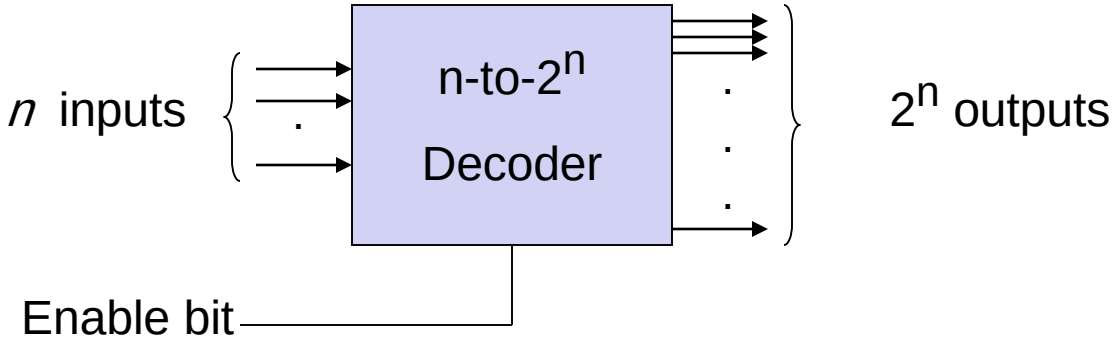
Load a
Add b
Store c
.
.

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



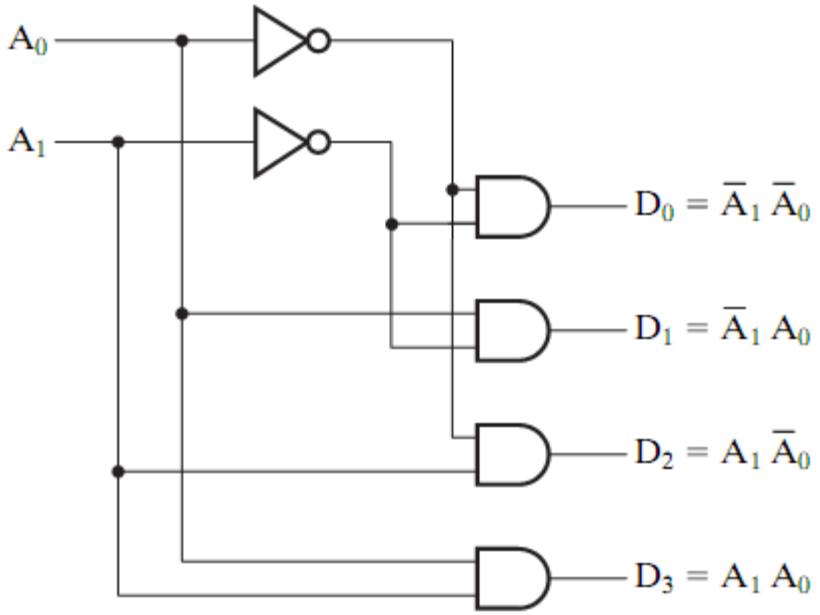
2-to-4 Decoder

A 2-to-4 Decoder

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

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

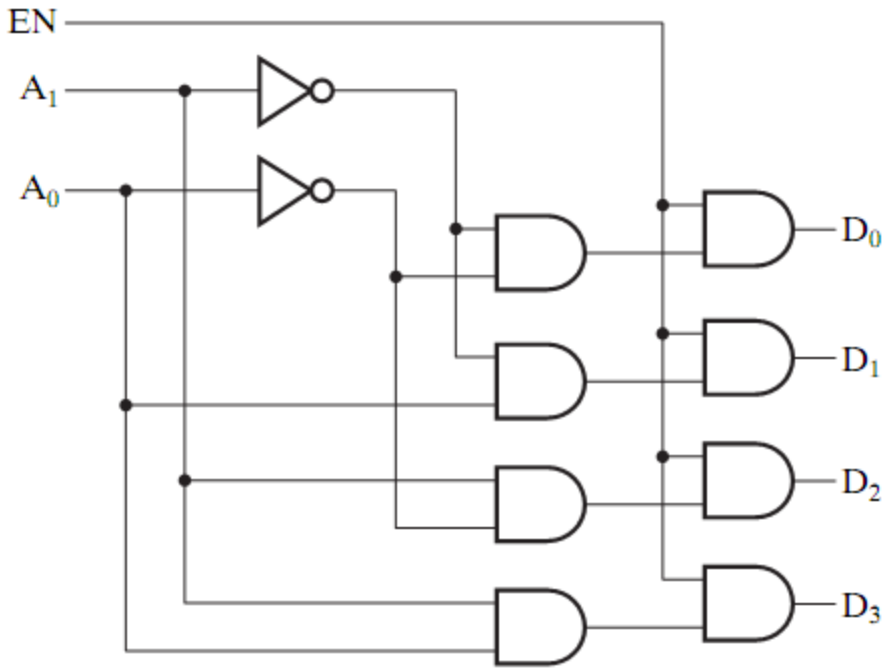


Src: Mano's book

2-to-4 Decoder with Enable

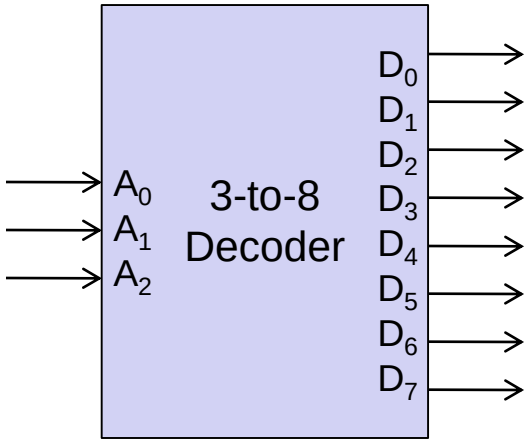
Truth Table

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



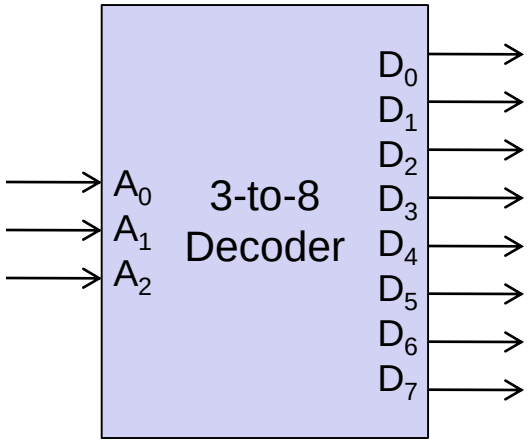
Src: Mano's book

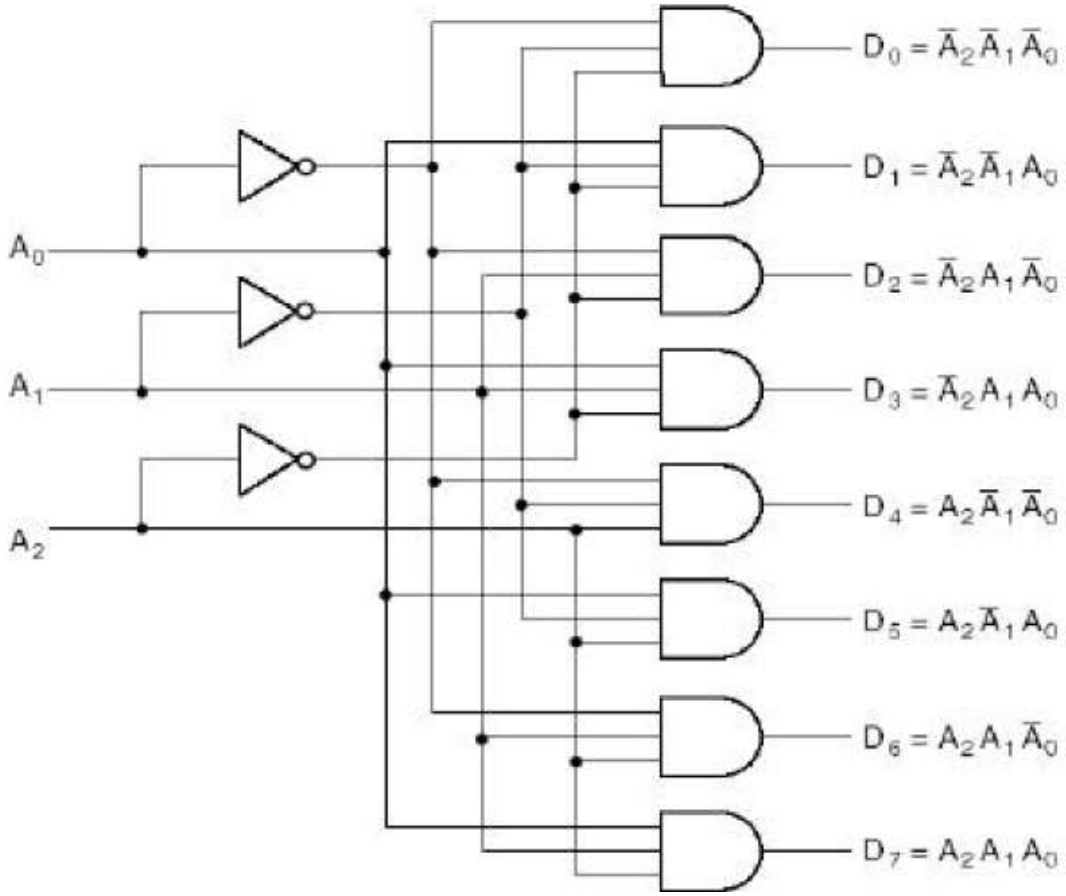
3-to-8 Decoder



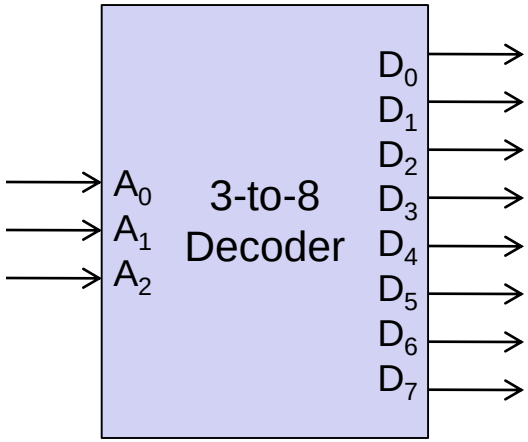
A_2	A_1	A_0	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

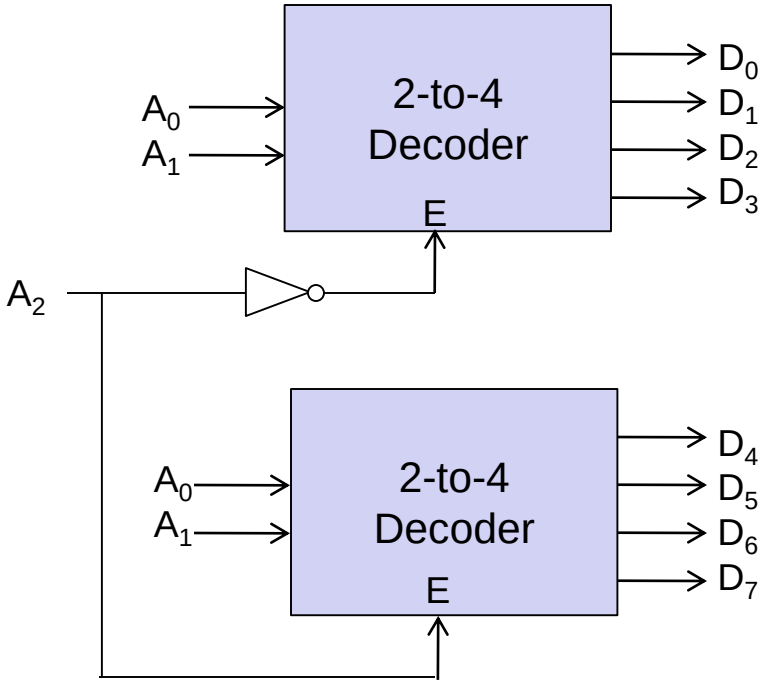
3-to-8 Decoder





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 as minterms (not minimized form)

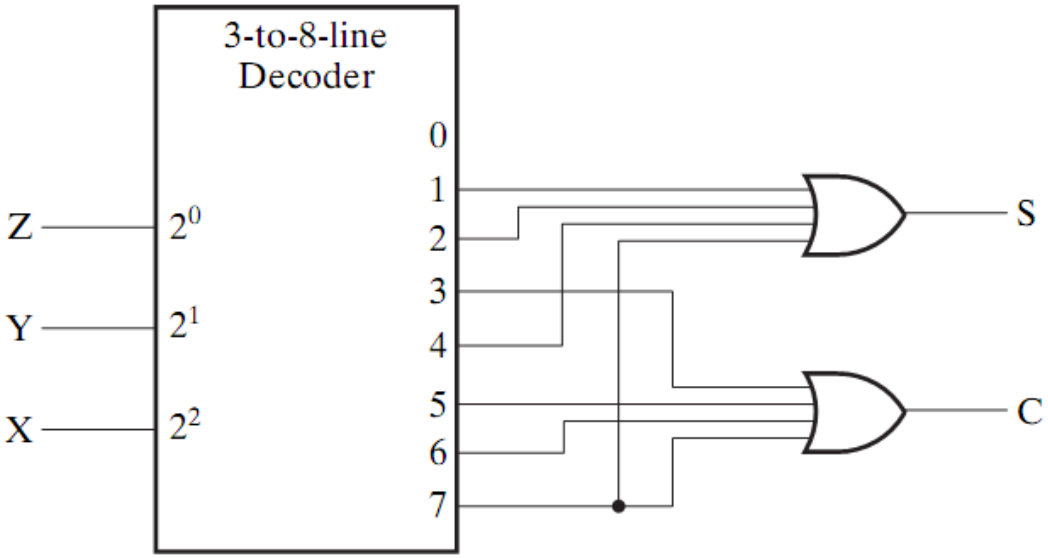
Decoder-Based Combinational Circuits (Example)

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

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

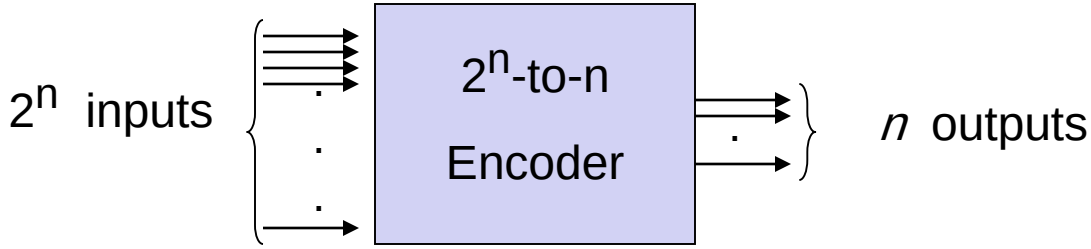
$C = \sum m\ (3,5,6,7)$

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



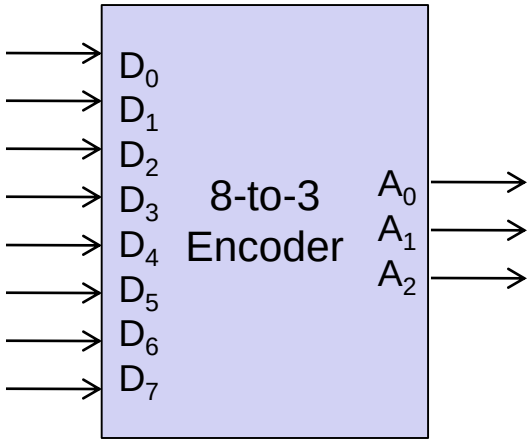
Src: Mano's book

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.

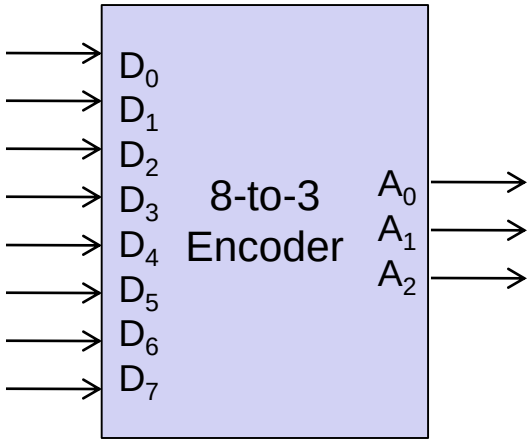
8-to-3 Encoder



Description:

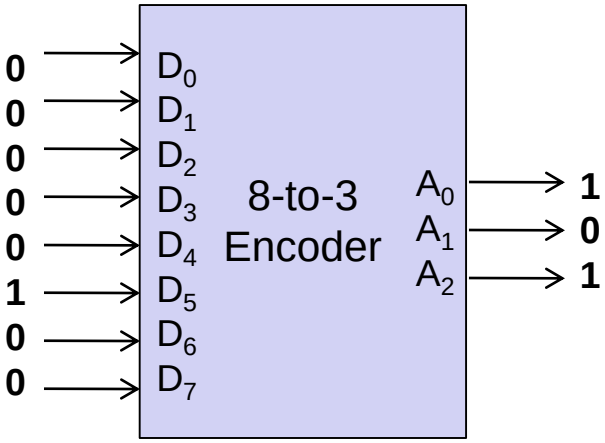
- 2^3 = 8 inputs, 3 outputs
- one input =1, others = 0's
- Each input generate unique binary code

8-to-3 Encoder (truth table)



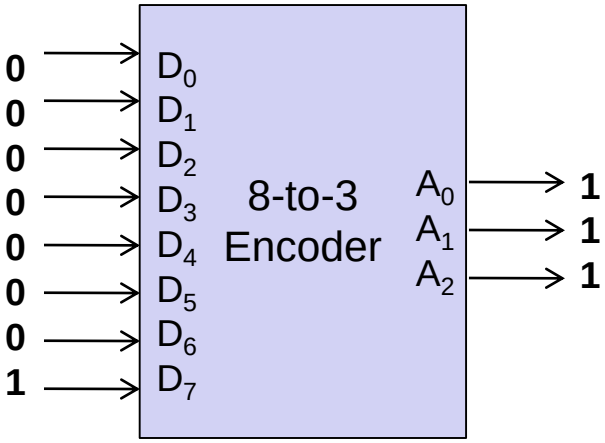
inputs								outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

8-to-3 Encoder (truth table)



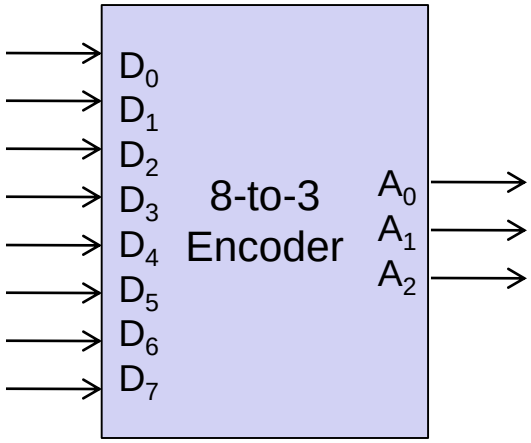
inputs								outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

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 (equations)



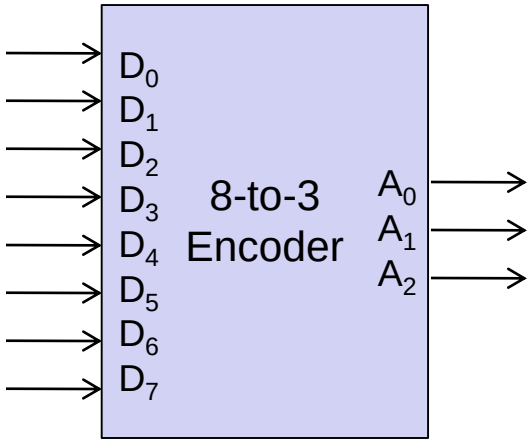
Output equations:

A_0 = ?
A_1 = ?
A_2 = ?

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

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

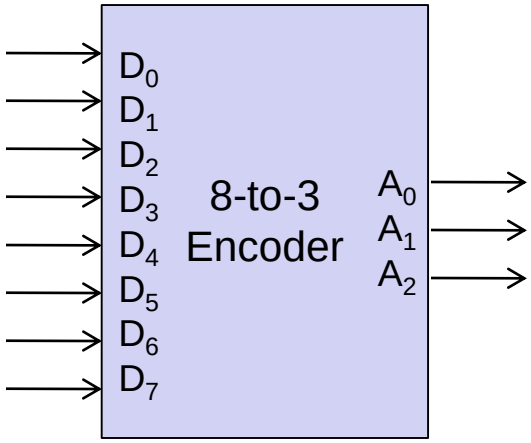
Output equations:

$A_0 = D_1 + D_3 + D_5 + D_7$

$A_1 = ?$

$A_2 = ?$

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

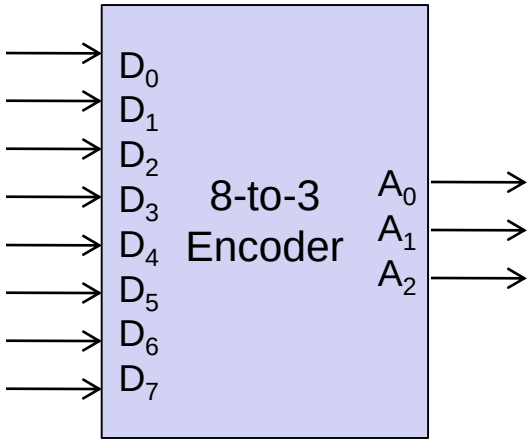
Output equations:

$A_0 = D_1 + D_3 + D_5 + D_7$

$A_1 = D_2 + D_3 + D_6 + D_7$

$A_2 = ?$

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

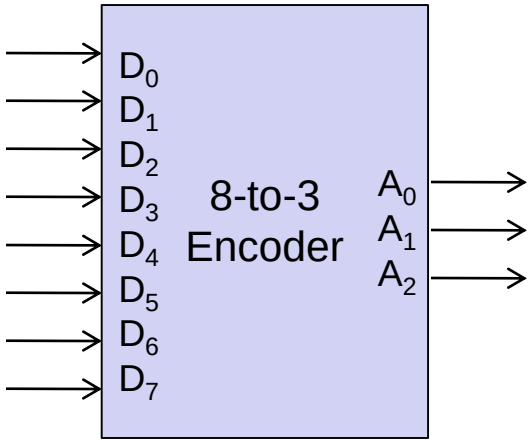
Output equations:

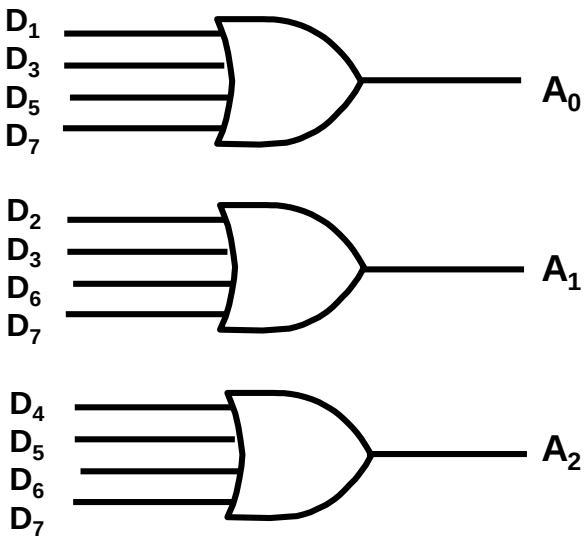
$A_0 = D_1 + D_3 + D_5 + D_7$

$A_1 = D_2 + D_3 + D_6 + D_7$

$A_2 = D_4 + D_5 + D_6 + D_7$

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 $D_0 = 1$

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

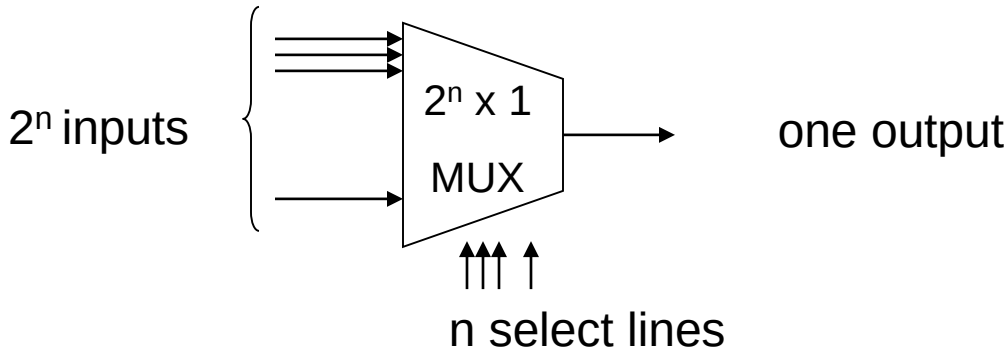
   Consider the bit with highest priority

2. All inputs = 0

   Add another output v to indicate this combination

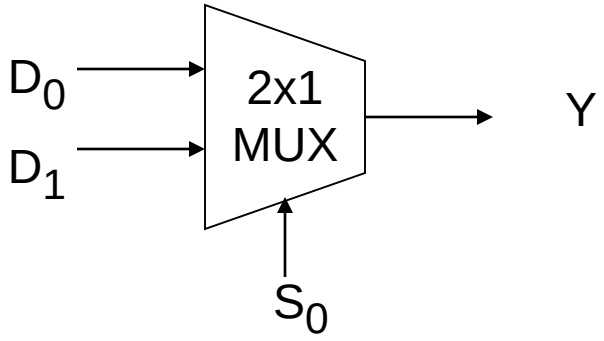
Multiplexers

- A combinational circuit
- Has a single output
- Directs one of 2^n input to the output
- Choosing which input is done using n select lines



2x1 MUX

- A 2x1 multiplexer (MUX) has 2 inputs, 1 output and 1 select line



- $Y=D_0$ for $S_0=0$, and $Y=D_1$ for $S_0=1$
- Minimizing will result in: $Y = S_0'.D_0 + S_0.D_1$
- **Exercise**: Draw the circuit?

4x1 MUX

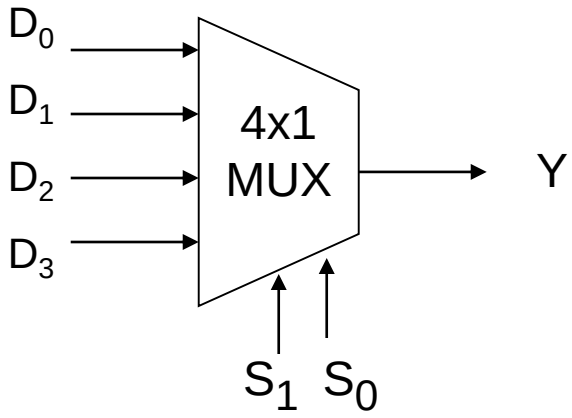
- A 4x1 MUX has 4 input lines (D_0, D_1, D_2, D_3) , 1 output Y, and 2 Select Lines (S_0, S_1)
- The output for different select values is defined as:

  $S_0S_1 = 00$, $Y = D_0$

  $S_0S_1 = 01$, $Y = D_1$

  $S_0S_1 = 10$, $Y = D_2$

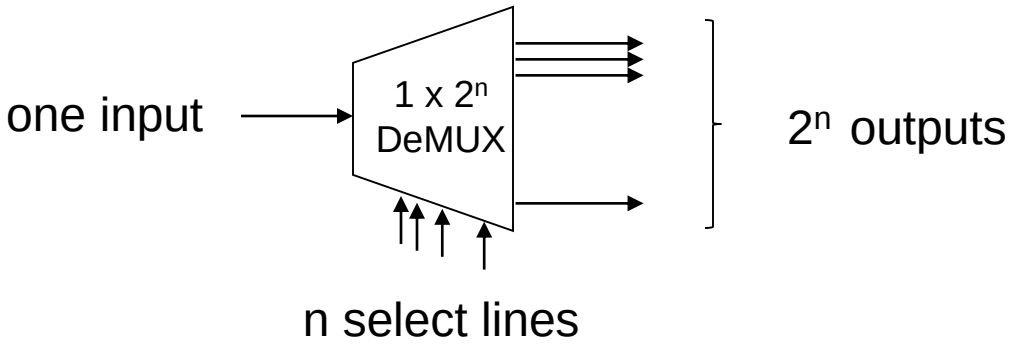
  $S_0S_1 = 11$, $Y = D_3$



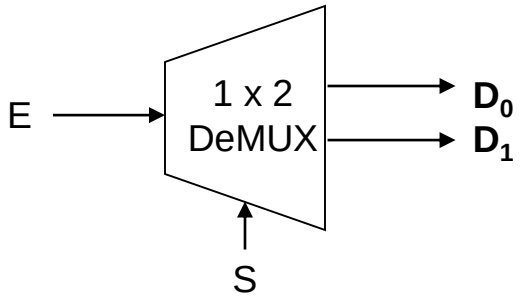
- $Y = \overline{S_1}\,\overline{S_0}D_0 + \overline{S_1}S_0D_1 + S_1\overline{S_0}D_2 + S_1S_0D_3$
- The output Y depends on the minterms of the Select lines
- **Exercise**: Draw the circuit?

DeMultiplexer

- Performs the inverse operation of a MUX
- It has one input and 2^n outputs
- The input is passed to one of the outputs based on the n select line



1x2 DeMUX

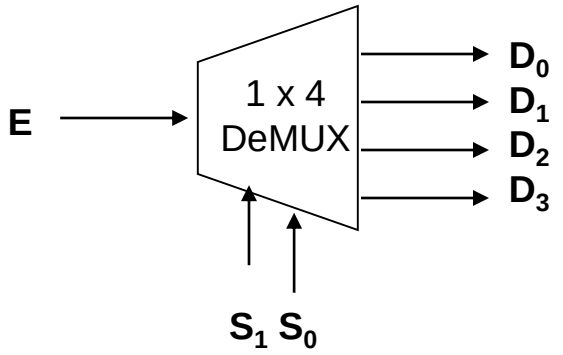


The circuit has an input E, the outputs are given by:

D_0 = E, if S=0 $\quad\quad$ $D_0 = \overline{S}\,E$

D_1 = E, if S=1 $\quad\quad$ $D_1 = S\,E$

1x4 DeMUX



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

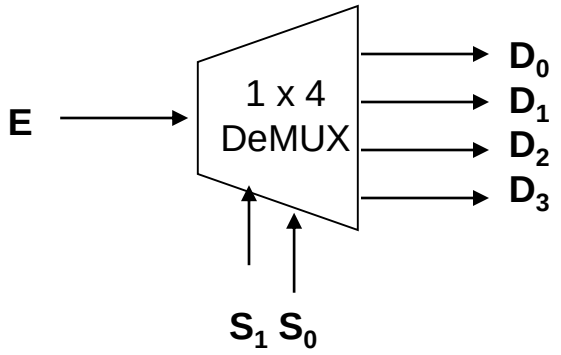
$D_0 = E$, if $S_0S_1=00$ $\quad D_0 = S_1'S_0' E$

$D_1 = E$, 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 vs Decoder

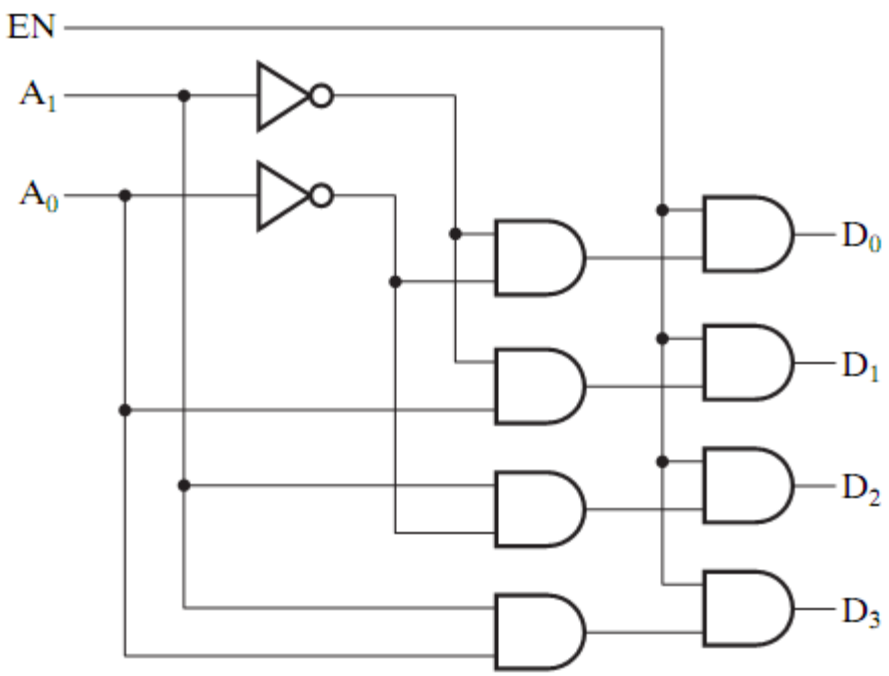


- A 1x4 DeMUX is equivalent to a 2x4 Decoder with an Enable
  - Think of S_1S_0 a the decoder's input
  - Think of E as the decoder's enable
- In general, a DeMux is equivalent to a Decoder with an Enable

DeMUX vs Decoder

2x4 Decoder 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

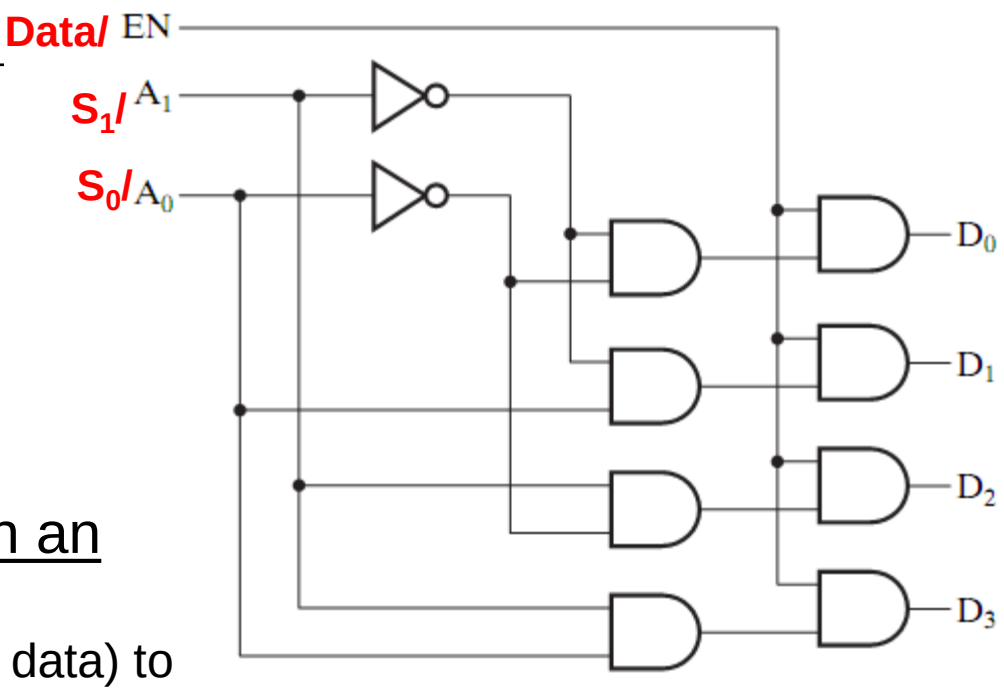


Src: Mano's book

DeMUX vs Decoder

2x4 Decoder 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

To convert a 2x4 Decoder with an Enable to a 1x4 DeMux:

- Assign DeMux's input (actual data) to EN
- Assign DeMux's selection lines (S_1,S_0) to the inputs A_1, A_0