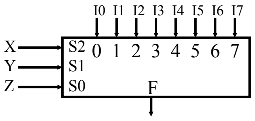
Multiplexing and Multiplexer

- Multiplexers are circuits which select one of many inputs
- In here, we assume that we have one-bit inputs
  (in general, each input may have more than one bit)

- Suppose we have eight inputs: I0, I1, I2, I3, I4, I5, I6, I7
- We want one of them to be output based on selection signals
- 3 bits of selection signals to decide which input goes to output
- Note the order of selection signals
  – X is MSB and Z is LSB

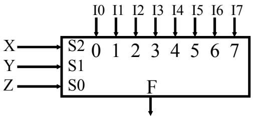
8-to-1 Multiplexer



X Y Z	F
0 0 0	I0
0 0 1	I1
0 1 0	I2
0 1 1	I3
1 0 0	I4
1 0 1	I5
1 1 0	I6
1 1 1	I7

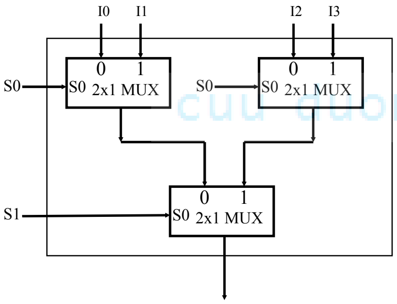
Multiplexer Implementation

- We can write a logic expression for output F as follows
  F = X' Y' Z' I0 + X' Y' Z I1 + X' Y Z' I2 + X' Y Z I3
  + X Y' Z' I4 + X Y' Z I5 + X Y Z' I6 + X Y Z I7
- This circuit can be implemented using
  – eight 4-input AND gates and one 8-input OR gates



X Y Z	F
0 0 0	I0
0 0 1	I1
0 1 0	I2
0 1 1	I3
1 0 0	I4
1 0 1	I5
1 1 0	I6
1 1 1	I7

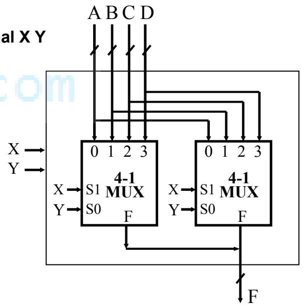
Implementing 4-to-1 MUX using 2-to-1 MUXs



Making a 2-bit 4-to-1 Multiplexer

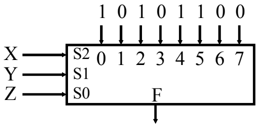
- Four *2-bit* inputs A, B, C, D
- One *2-bit* output F
- Two bits of selection signal X Y

X	Y	F
0	0	A
0	1	B
1	0	C
1	1	D



Synthesis of Logic Functions using Multiplexers

- Multiplexers can be directly used to implement a function
- Easiest way is to use function inputs as selection signals
- Input to multiplexer is a set of 1s and 0s depending on the function to be implemented
- We use a 8-to-1 multiplexer to implement function F
- Three select signals are X, Y, and Z, and output is F
- Eight inputs to multiplexer are 1 0 1 0 1 1 0 0
- Depending on the input signals
  – multiplexer will select proper output

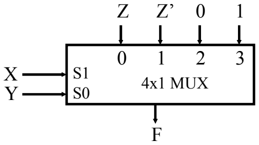


X Y Z	F
0 0 0	1
0 0 1	0
0 1 0	1
0 1 1	0
1 0 0	1
1 0 1	1
1 1 0	0
1 1 1	0

Implementing 3-variable functions with 4x1 MUX

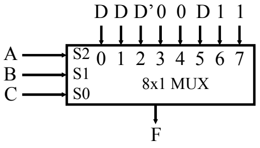
- Divide the outputs into 4 groups based on X and Y.
- Write the outputs as a function of Z
- There are only 4 possibilities: F=Z, F=Z', F=0, F=1

X Y Z	F	
0 0 0	0	F=Z
0 0 1	1	
0 1 0	1	F=Z'
0 1 1	0	
1 0 0	0	F=0
1 0 1	0	
1 1 0	1	F=1
1 1 1	1	



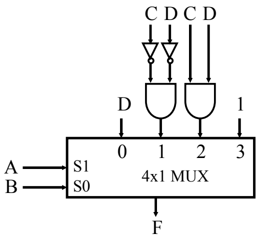
Implementing 4-variable functions with 8x1 MUX

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	



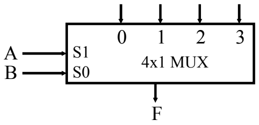
Implementing 4-variable functions with 4x1 MUX

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



Implementing 4-variable functions with 4x1 MUX

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

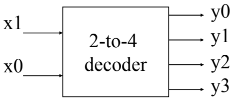


Definition of Decoder

- Suppose we have n input bits (which can represent up to 2^n distinct elements of coded information).
- We need a device that allows us to select which of the 2^n elements, devices, memory locations, etc. is being selected.
- In general:
  - A decoder has n input bits
  - A decoder has 2^n (or less) output bits
  - As a rule, all but one of the outputs is zero (deselected) at any time (called *one-hot encoded*)

2-to-4 Decoder

- The 2-to-4 decoder is a block which decodes the 2-bit binary inputs and produces four outputs
- One output corresponding to the input combination is a one
- Two inputs and four outputs are shown in the figure
- The equations are
  - y0 = x1’. x0’
  - y1 = x1’. x0
  - y2 = x1 . x0’
  - y3 = x1 . x0



- The truth table:

x1	x0	y3	y2	y1	y0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Definition of Encoder

- Encoders perform the inverse function of Decoders.
- An encoder has 2^n (or less) input bits and n output bits
- The output bits generate the binary code corresponding to the input value
- Assuming only one input has a value of 1 at any given time
- Example: An 8-to-3 Encoder

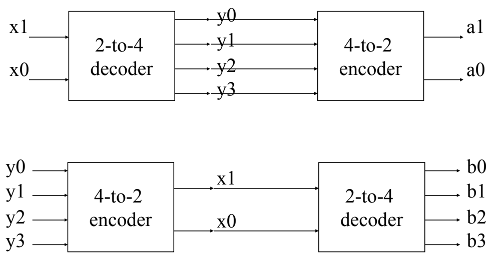
Inputs								Outputs		
D7	D6	D5	D4	D3	D2	D1	D0	A2	A1	A0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

A2=D4+D5+D6+D7

A1=D2+D3+D6+D7

A0=D1+D3+D5+D7

What are the outputs of the following circuits?



Priority Encoders

- Each input signal has a priority level associated with it
- May have more than one 1’s in the input signals
- Outputs indicate the active input that has the highest priority
- Example: 4-to-2 priority encoder
  - Assume w3 has the highest priority and w0 the lowest
  - y1 y0 indicate the active input with highest priority
  - z indicates none of the inputs is equal to 1

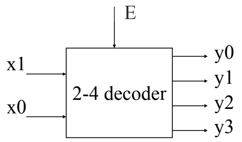
w3	w2	w1	w0	y1	y0	z
0	0	0	0	d	d	0
0	0	0	1	0	0	1
0	0	1	x	0	1	1
0	1	x	x	1	0	1
1	x	x	x	1	1	1

x: both 0 and 1 (irrelevant)

Let i0 = w0 w1’ w2’ w3’
i1 = w1 w2’ w3’
i2 = w2 w3’
i3 = w3
Then y0 = i1 + i3
y1 = i2 + i3

Decoder with Enable

- A 2-to-4 decoder can be designed with an enable signal
- If enable is zero, all outputs are zero
- If enable is 1, then an output corresponding to two inputs is a one, all others are still zero
- The equations are
  - y0 = x1’. x0’. E
  - y1 = x1’. x0 . E
  - y2 = x1 . x0’. E
  - y3 = x1 . x0 . E

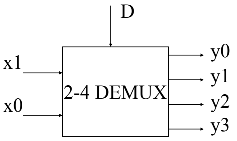


Truth Table for 2-to-4 Decoder with Enable

x1	x0	E	y3	y2	y1	y0
0	0	0	0	0	0	0
0	0	1	0	0	0	1
0	1	0	0	0	0	0
0	1	1	0	0	1	0
1	0	0	0	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	0	0
1	1	1	1	0	0	0

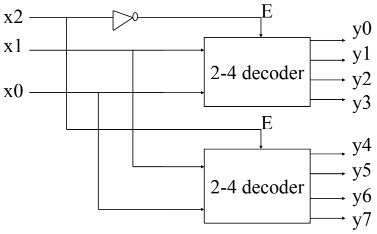
Demultiplexers

- Perform the opposite function of multiplexers
- Placing the value of a single data input onto one of the multiple data outputs
- Same implementation as decoder with enable
- Enable input of decoder serves as the data input for the demultiplexer



3-to-8 decoder using a 2-to-4 decoder with Enable

- The 3-to-8 decoder can be implemented using two 2-to-4 decoders with enable and one NOT gate
- The implementation is as shown


x2
x1
x0
E
2-4 decoder
y0
y1
y2
y3
E
2-4 decoder
y4
y5
y6
y7