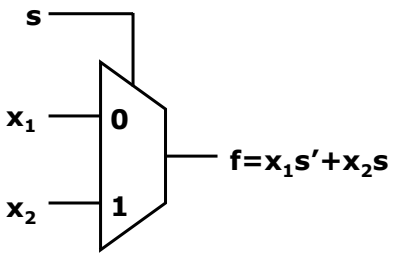
ECE380 Digital Logic

Combinatorial Circuit Building Blocks:

Multiplexers

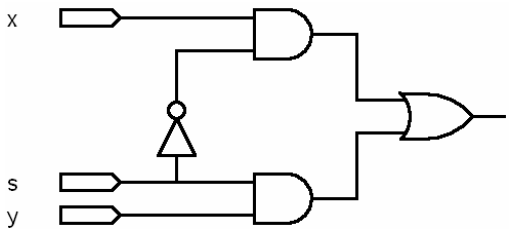
Multiplexers

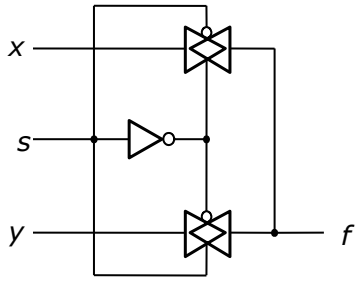
- A multiplexer (MUX) circuit has
  - A number of data inputs
  - One (or more) select inputs
  - One output
- It passes the signal value on one of its data inputs to its output based on the value(s) of the select signal(s)



s	$f(s,x_1,x_2)$
0	x_1
1	x_2

Multiplexer implementations

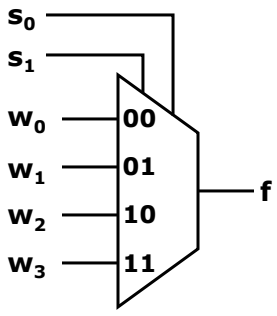




The preferred implementation

4-input multiplexer

- A 4-input multiplexer 'selects' one of four data inputs to be output based on the values of 2 select lines

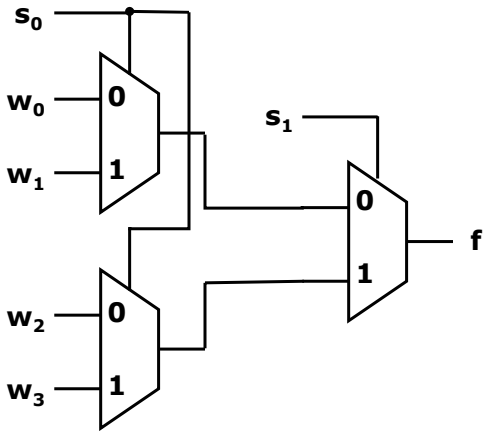


s_1	s_0	f
0	0	w_0
0	1	w_1
1	0	w_2
1	1	w_3

$$f = s_1's_0'w_0 + s_1's_0w_1 + s_1s_0'w_2 + s_1s_0w_3$$

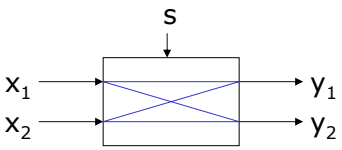
Building a 4-input MUX

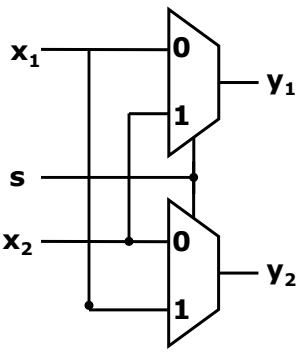
- A 4-input multiplexer can be constructed using 2-input multiplexers



MUX application (a 2x2 crossbar)

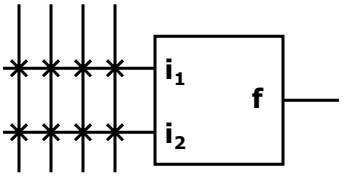
- A circuit with *n* inputs and *k* outputs whose function is to provide a capability to connect any input to any output is called a ***nxk crossbar switch***
  - With 2 inputs and 2 outputs, it is called a 2x2 crossbar
  - Useful in applications where it is necessary to connect one set of wires to another set of wires, where the connection pattern changes from time to time
  - Telephone switching networks are an example



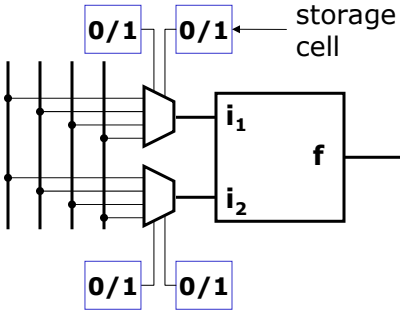


MUX application (prog. switch)

- In programmable devices (PLDs, CPLDs and FPGAs) programmable switches connect wires inside the device
  - These can be implemented with multiplexers



An FPGA logic block with programmable inputs

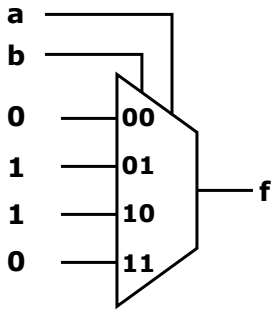


MUX implementation

Logic functions using MUXs

- MUXs can be used to synthesize logic functions
  - The LUT implementations use MUXs to select a (constant) value from a look-up table
- Consider the XOR function

a	*b*	*f*
0	0	0
0	1	1
1	0	1
1	1	0

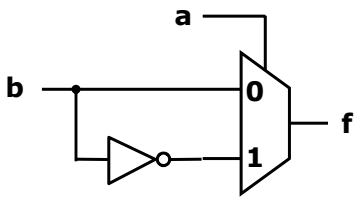


Logic functions using MUXs

- The previous XOR solution is not particularly efficient

a	*b*	*f*	
0	0	0	when a=0, f=b
0	1	1	
1	0	1	when a=1, f=b′
1	1	0	

a	*f*
0	b
1	b′



Logic functions using MUXs

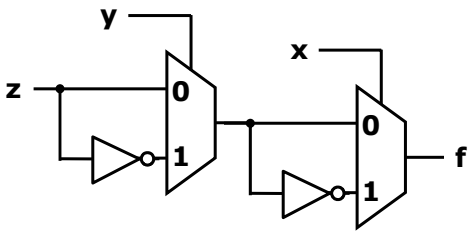
- Implement the following with a 2-input MUX and any additional logic gates

a	*b*	*f*
0	0	1
0	1	1
1	0	0
1	1	1

Logic functions using MUXs

- A 3-input XOR can be implemented with two 2-input MUXs

x	*y*	*z*	*f*	
0	0	0	0	y⊕z
0	0	1	1	
0	1	0	1	
0	1	1	0	
1	0	0	1	(y⊕z)′
1	0	1	0	
1	1	0	0	
1	1	1	1	



Logic functions using MUXs

- Implement the following with 2-input MUXs and any additional logic gates

x	*y*	*z*	*f*
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Shannon's expansion theorem

- Any Boolean function $f(w_1,\ldots,w_n)$ can be written in the form

  $f(w_1,\ldots,w_n)=(w_1)' \cdot f(0,w_2,\ldots,w_n)+(w_1) \cdot f(1,w_2,\ldots,w_n)$
- The expansion can be done using any of the n variables
- If $f(w_1,w_2,w_3)= w_1w_2+w_1w_3+w_2w_3$
  - Expanding this in terms of w_1 gives

    $f(w_1,w_2,w_3)= w_1(w_2+w_3)+(w_1)'(w_2w_3)$

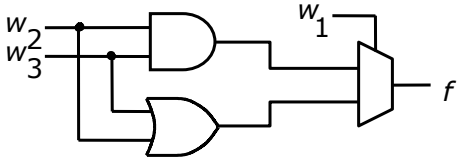
    $w_1(w_2+w_3)$: f when $w_1=1$

    $(w_1)'(w_2w_3)$: f when $w_1=0$

Shannon's expansion example

w_1	w_2	w_3	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

w_1	f
0	w_2w_3
1	w_2+w_3



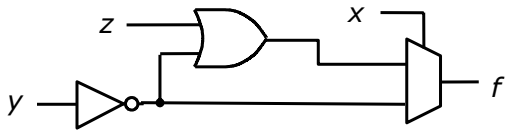
Shannon's expansion example

x	*y*	*z*	*f*
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

f=x'y'z'+x'y'z+x'yz+xy'z'+xy'z

choose x as the expansion variable

f=x'(y'z'+y'z+yz)+x(y'z'+y'z)
f=x'(y'+z)+x(y')



Shannon's expansion example

x	*y*	*z*	*f*
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

f=x'y'z'+x'y'z+x'yz+xy'z'+xy'z

choose z as the expansion variable