Shannon’s Expansion

Shannon's Expansion Theorem

- Any Boolean function $f(w_1, w_2, \dots, w_n)$can be written in the form

$$f(w_1, w_2, \dots, w_n) = w_1' \cdot f(0, w_2, \dots, w_n) + w_1 \cdot f(1, w_2, \dots, w_n)$$

- Any of the n variables can be used.

Proof

It suffices to prove the theorem holds for all possible values of one of the variables.

Since this is a binary function each variable can only be 1 or 0.

Therefore we need only look at two cases.

Proof

Let w_1= 0, then

$f(w_1, w_2, \dots, w_n)$ = 1· $f(0, w_2, \dots, w_n)$ + 0· $f(1, w_2, \dots, w_n)$

= $f(0, w_2, \dots, w_n)$

Let w_1= 1, then

$f(w_1, w_2, \dots, w_n)$ = 0· $f(0, w_2, \dots, w_n)$ + 1· $f(0, w_2, \dots, w_n)$

= $f(1, w_2, \dots, w_n)$

The proof holds for any arbitrary x.

Example: 3 input majority circuit

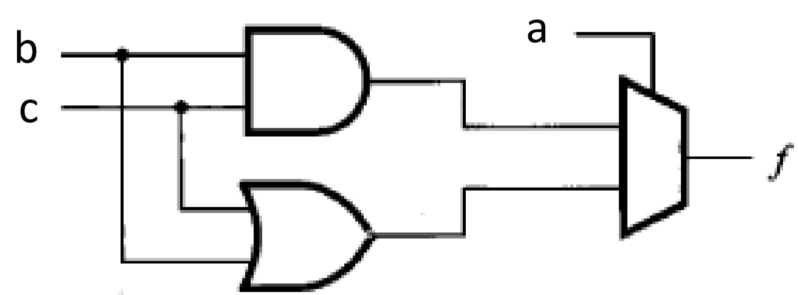
Given $f(a, b, c) = ab+ac+bc$

a	b	c	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

a’

a

a	f
0	bc
1	b + c

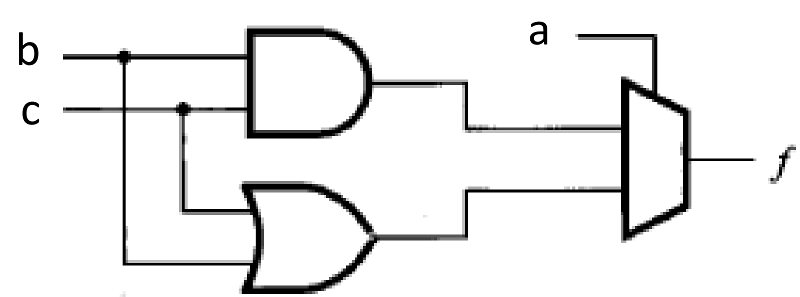


Shannon's Expansion Theorem

Given f(a, b, c) = ab+ac+bc

Expanding this function in terms of a gives

$$f = a'(0 \cdot b + 0 \cdot c + bc) + \ a(1 \cdot b + 1 \cdot c + bc)$$
$$f = a'(bc) + a(b + c)$$



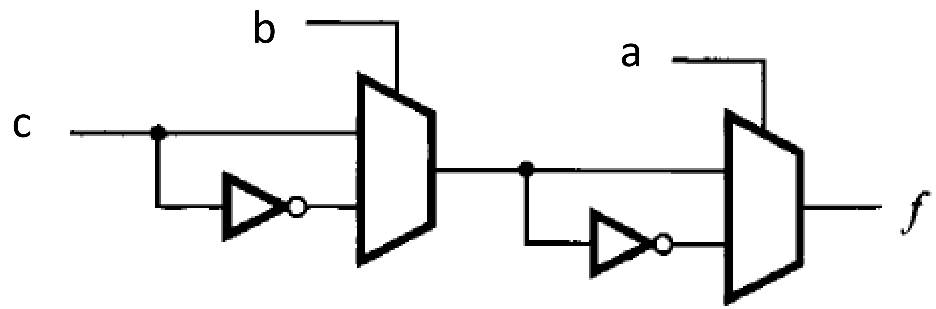
Example: 3 input XOR

a	b	c	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

a’

a

a	f
0	b XOR c
1	(b XOR c)’



Example: 3 input XOR

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

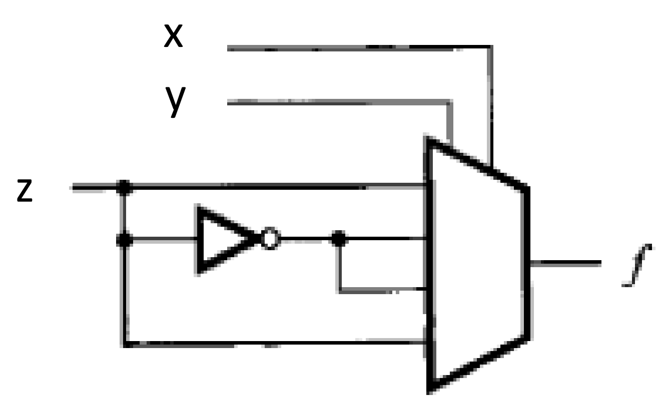
z

z’

z’

z

xy	f
00	z
01	z’
10	z’
11	z



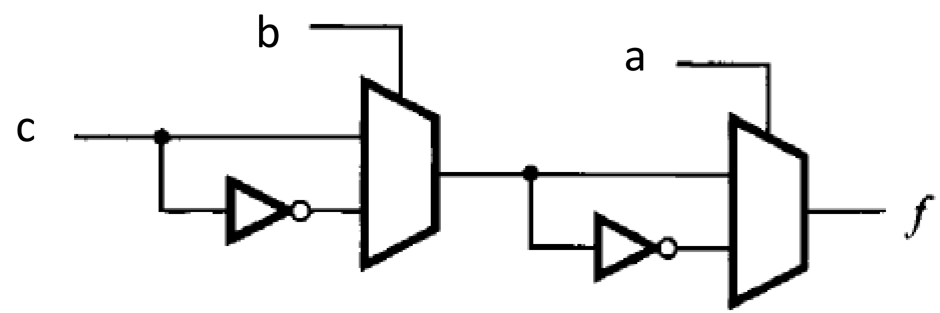
Shannon's Expansion Theorem

3 input XOR function

$f = a \oplus b \oplus c$

$= a'(0 \oplus b \oplus c) + a(1 \oplus b \oplus c)$

$= a'(b \oplus c) + a(b \oplus c)'$



Example

$f = a'c + bc'$

Expansion on a

$f = a'(b + c) + a(bc')$

Expansion on b

$f = b'(a'c) + b(a' + c')$

Expansion on c

$f = c'(b) + c(a')$

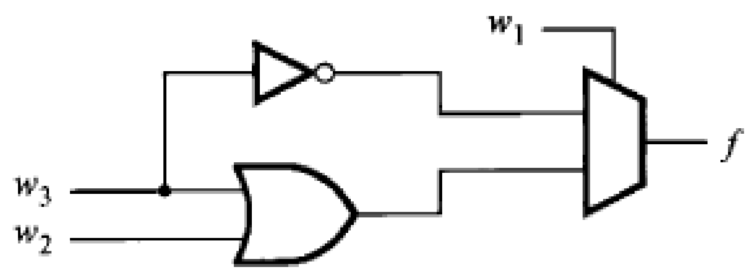
The point?
Some expansions are more efficient than others.

More problems

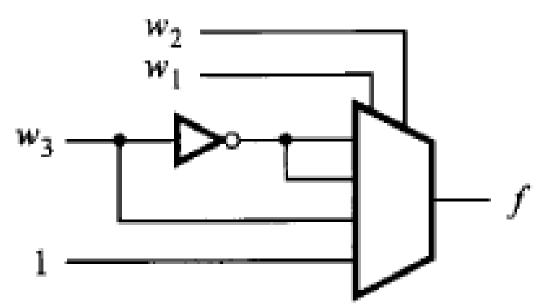
$f = a'c' + ab + ac$

Implement using a 2-to-1 multiplexer and any other necessary gates.

Implement the same function using a 4-to-1 multiplexer.



(a) Using a 2-to-1 multiplexer



(b) Using a 4-to-1 multiplexer