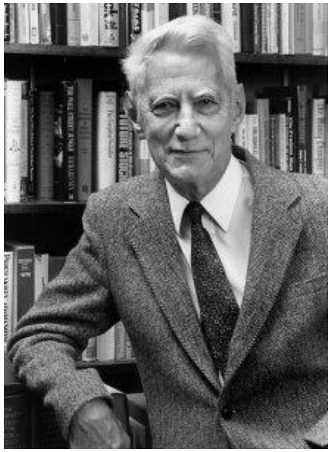
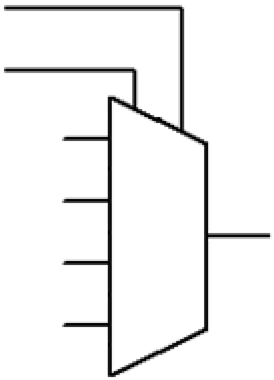
Shannon decomposition



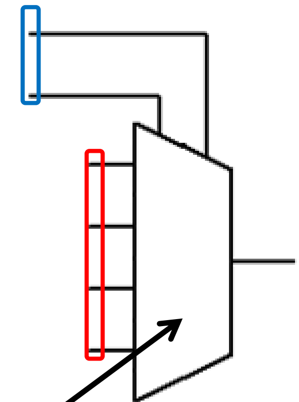
Claude Shannon
mathematician / electrical engineer
(1916 –2001)

(Ex 8.6)



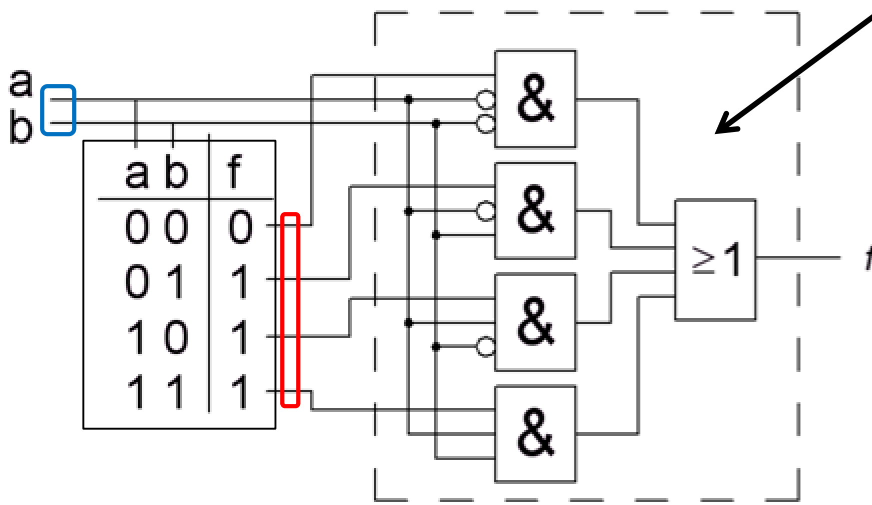
Show how a 4-to-1 multiplexer can be used as a "function generator" for example to generate the OR function.

(Ex 8.6)

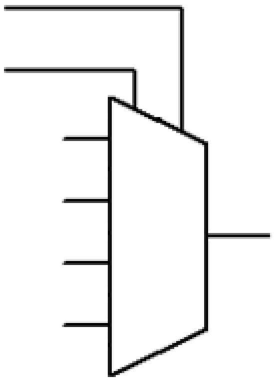


Show how a 4-to-1 multiplexer can be used as a "function generator" for example to generate the OR function.

Multiplexer as function generator

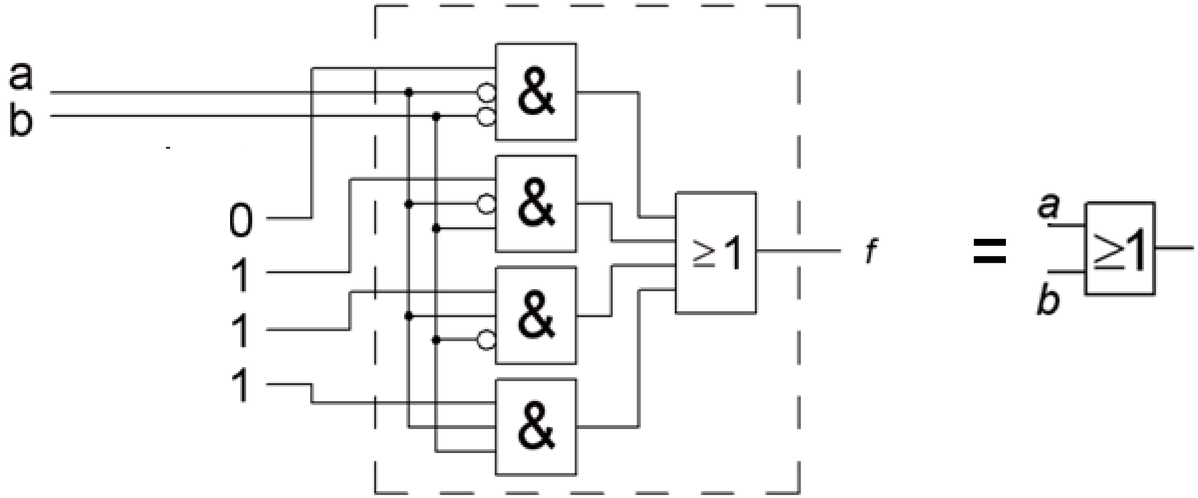


(Ex 8.6)



Show how a 4-to-1 multiplexer can be used as a "function generator" for example to generate the OR function.

Multiplexer as function generator

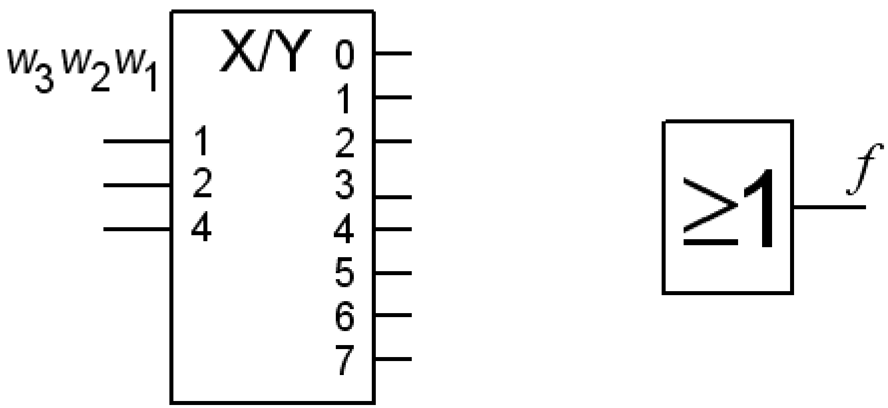


BV 6.1

Show how the function

$$f(w_1, w_2, w_3) = \sum m(0, 2, 3, 4, 5, 7)$$

can be implemented using a 3-to-8 **decoder** and an OR gate.

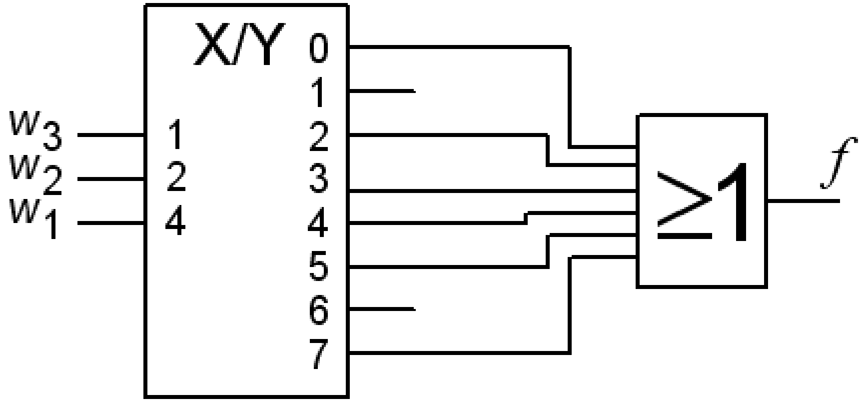


BV 6.1

Show how the function

$$f(w_1, w_2, w_3) = \sum m(0, 2, 3, 4, 5, 7)$$

can be implemented using a 3-to-8 **decoder** and an OR gate.

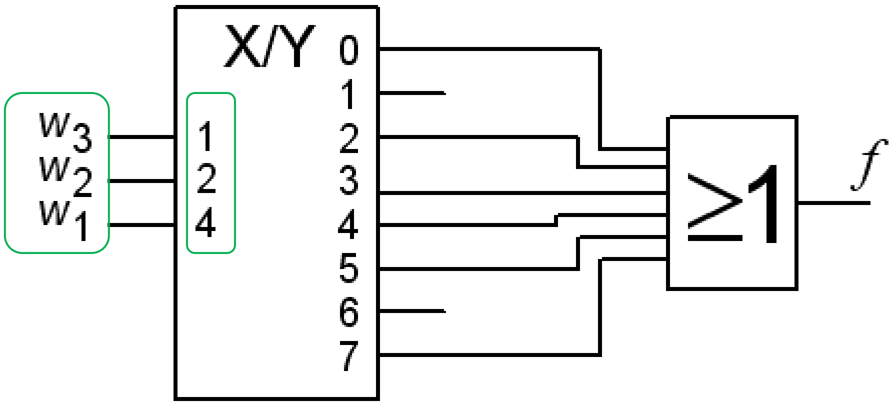


BV 6.1

Show how the function

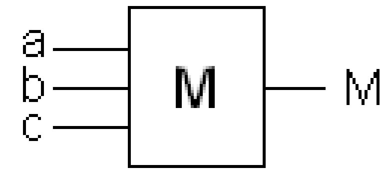
$$f(w_1, w_2, w_3) = \sum m(0, 2, 3, 4, 5, 7)$$

can be implemented using a 3-to-8 **decoder** and an OR gate.



- The correct order is important!

Ex 8.7



A majority gate outputs the same value as the majority of the inputs. The gate can for example be used in fault-tolerant logic, or in image processing circuits.

a) Set up the gate's truth table and minimize the function with Karnaugh map. Realize the function with AND-OR gates.

b) Realize the majority gate with an 8:1 MUX.

c) Use Shannon decomposition and realize the majority gate with a 2:1 MUX and gates.

d) Realize the majority gate with only 2:1 MUXes.

(8.7a)

With AND OR gates

	a	b	c	M	
0	0	0	0	0	
1	0	0	1	0	
2	0	1	0	0	
3	0	1	1	1	$\bar{a}bc$
4	1	0	0	0	
5	1	0	1	1	$a\bar{b}c$
6	1	1	0	1	$ab\bar{c}$
7	1	1	1	1	abc

(8.7a)

	a	b	c	M	
0	0	0	0	0	
1	0	0	1	0	
2	0	1	0	0	
3	0	1	1	1	$\bar{a}bc$
4	1	0	0	0	
5	1	0	1	1	$a\bar{b}c$
6	1	1	0	1	$ab\bar{c}$
7	1	1	1	1	abc

With AND OR gates

M

a \ bc	00	01	11	10
0	^{0}0	^{1}0	^{3}1	^{2}0
1	^{4}0	^{5}1	^{7}1	^{6}1

(8.7a)

	a	b	c	M	
0	0	0	0	0	
1	0	0	1	0	
2	0	1	0	0	
3	0	1	1	1	$\bar{a}bc$
4	1	0	0	0	
5	1	0	1	1	$a\bar{b}c$
6	1	1	0	1	$ab\bar{c}$
7	1	1	1	1	abc

With AND OR gates

M

a \ bc	00	01	11	10
0	0 (0)	1 (0)	3 (1)	2 (0)
1	4 (0)	5 (1)	7 (1)	6 (1)

$$M = ac + ab + bc$$

(8.7a)

	a	b	c	M
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

With AND OR gates

$\bar{a}bc$

$a\bar{b}c$

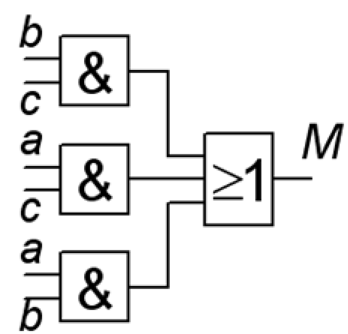
$ab\bar{c}$

abc

M

a \ bc	00	01	11	10
0	0: 0	1: 0	3: 1	2: 0
1	4: 0	5: 1	7: 1	6: 1

$$M = ac + ab + bc$$



8.7b

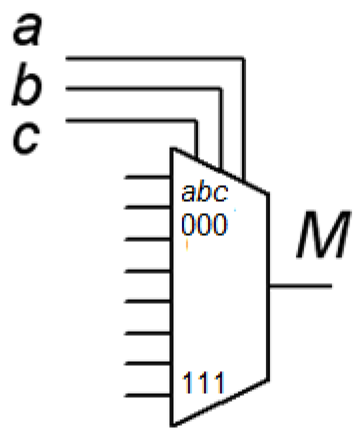
	a	b	c	M
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

With 8-to-1 mux …

8.7b

	a	b	c	M
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

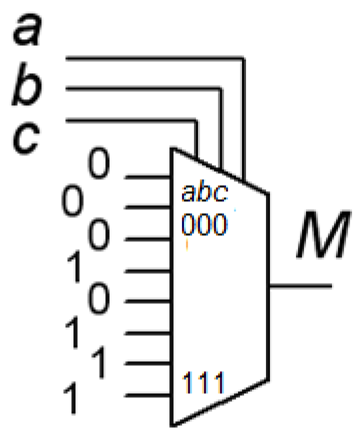
With 8-to-1 mux …



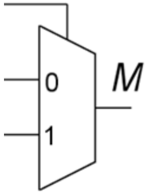
8.7b

	a	b	c	M
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

With 8-to-1 mux …



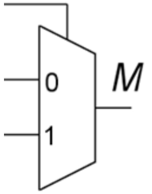
8.7c



Shannon decomposition. ***2-to-1 mux and gates.***

	a	b	c	M	
0	0	0	0	0	
1	0	0	1	0	
2	0	1	0	0	
3	0	1	1	1	$\bar{a}bc$
4	1	0	0	0	
5	1	0	1	1	$a\bar{b}c$
6	1	1	0	1	$ab\bar{c}$
7	1	1	1	1	abc

8.7c



	a	b	c	M
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

Shannon decomposition. ***2-to-1 mux and gates.***

$\bar{a}bc$

$a\bar{b}c$

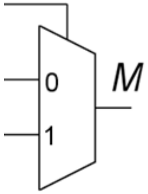
$ab\bar{c}$

abc

$$M = \bar{a}bc + a\bar{b}c + ab\bar{c} + abc =$$

$$= \bar{a}(bc) + a(\bar{b}c + b\bar{c} + bc) =$$

8.7c



	a	b	c	M
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

Shannon decomposition. ***2-to-1 mux and gates.***

$\bar{a}bc$

$a\bar{b}c$

$ab\bar{c}$

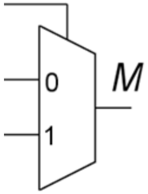
abc

$$M = \bar{a}bc + a\bar{b}c + ab\bar{c} + abc =$$

$$= \bar{a}(bc) + a\boxed{(\bar{b}c + b\bar{c} + bc)} =$$

b	c	?	
0	0	0	
0	1	1	$\bar{b}c$
1	0	1	$b\bar{c}$
1	1	1	bc

8.7c



Shannon decomposition. ***2-to-1 mux and gates.***

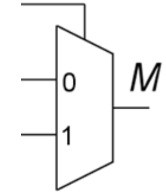
	a	b	c	M	
0	0	0	0	0	
1	0	0	1	0	
2	0	1	0	0	
3	0	1	1	1	$\bar{a}bc$
4	1	0	0	0	
5	1	0	1	1	$a\bar{b}c$
6	1	1	0	1	$ab\bar{c}$
7	1	1	1	1	abc

$$M = \bar{a}bc + a\bar{b}c + ab\bar{c} + abc =$$
$$= \bar{a}(bc) + a(\bar{b}c + b\bar{c} + bc) =$$
$$= \bar{a}(bc) + a(b + c)$$

b	c		
0	0	0	
0	1	1	$\bar{b}c$
1	0	1	$b\bar{c}$
1	1	1	bc

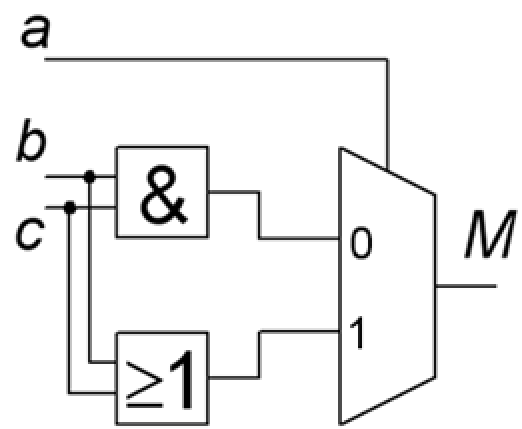
OR

8.7c



	a	b	c	M	
0	0	0	0	0	
1	0	0	1	0	
2	0	1	0	0	
3	0	1	1	1	$\bar{a}bc$
4	1	0	0	0	
5	1	0	1	1	$a\bar{b}c$
6	1	1	0	1	$ab\bar{c}$
7	1	1	1	1	abc

Shannon decomposition. ***2-to-1 mux and gates.***

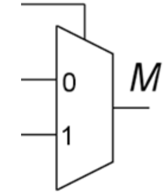


$$M = \bar{a}bc + a\bar{b}c + ab\bar{c} + abc =$$
$$= \bar{a}(bc) + a(\bar{b}c + b\bar{c} + bc) =$$
$$= \bar{a}(bc) + a(b + c)$$

b	c		
0	0	0	
0	1	1	$\bar{b}c$
1	0	1	$b\bar{c}$
1	1	1	bc

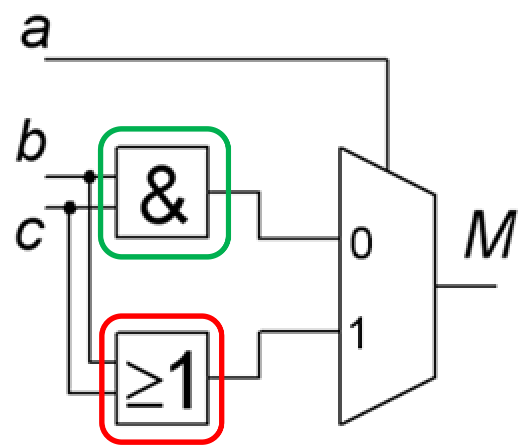
OR

8.7c



	a	b	c	M		
0	0	0	0	0		
1	0	0	1	0		
2	0	1	0	0	AND	
3	0	1	1	1		$\overline{a}bc$
4	1	0	0	0		
5	1	0	1	1		$a\overline{b}c$
6	1	1	0	1	OR	$ab\overline{c}$
7	1	1	1	1		abc

Shannon dekomposition. ***2-to-1 mux and gates.***



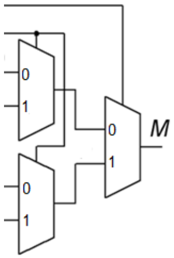
Another way – *a* divides the truth table in two halves. Then solve two simpler nets.

$$M = \overline{a}(bc) + a(b+c)$$

8.7d

	a	b	c	M
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

Shannon decomposition. ***Only 2-to-1 muxes.***



$\bar{a}bc$

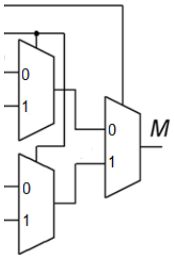
$a\bar{b}c$

$ab\bar{c}$

abc

8.7d

Shannon decomposition. ***Only 2-to-1 muxes.***



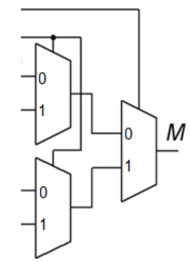
	a	b	c	M	
0	0	0	0	0	
1	0	0	1	0	
2	0	1	0	0	
3	0	1	1	1	$\bar{a}bc$
4	1	0	0	0	
5	1	0	1	1	$a\bar{b}c$
6	1	1	0	1	$ab\bar{c}$
7	1	1	1	1	abc

$$M = \overline{a}\,(bc) + a(b+c) \quad g = bc \quad h = b + c$$

$$g = \overline{b}\,(0) + b(c) \quad = \overline{b}\cdot 0 + b\cdot c$$

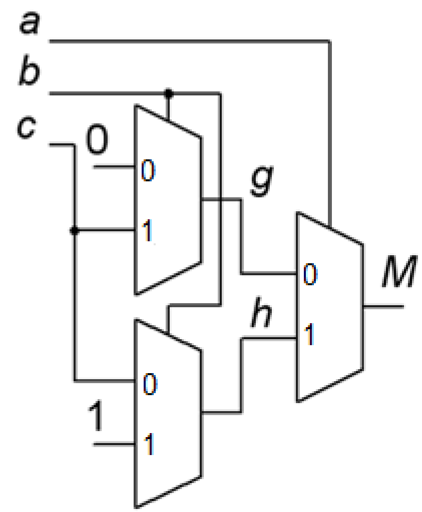
$$h = b + c \quad = b + (b + \overline{b})c = \overline{b}\,c + b + bc = \overline{b}\,c + b(1+c) = \overline{b}\cdot c + b\cdot 1$$

8.7d



	a	b	c	M	
0	0	0	0	0	
1	0	0	1	0	
2	0	1	0	0	
3	0	1	1	1	$\bar{a}bc$
4	1	0	0	0	
5	1	0	1	1	$a\bar{b}c$
6	1	1	0	1	$ab\bar{c}$
7	1	1	1	1	abc

Shannon decomposition. ***Only 2-to-1 muxes.***

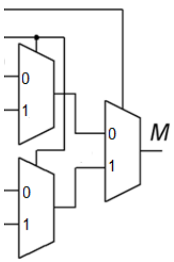


$$M = \bar{a}(bc) + a(b+c) \quad g = bc \quad h = b+c$$

$$g = \bar{b}(0) + b(c) \quad = \bar{b}\cdot 0 + b\cdot c$$

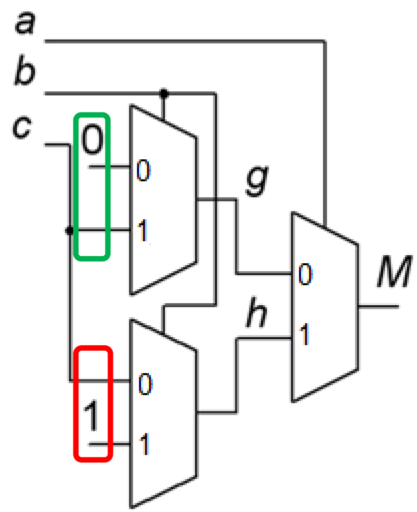
$$h = b + c \quad = b + (b+\bar{b})c = \bar{b}c + b + bc = \bar{b}c + b(1+c) = \bar{b}\cdot c + b\cdot 1$$

8.7d



	a	b	c	M	
0	0	0	0	0	
1	0	0	1	0	
2	0	1	0	0	
3	0	1	1	1	$\bar{a}bc$
4	1	0	0	0	
5	1	0	1	1	$a\bar{b}c$
6	1	1	0	1	$ab\bar{c}$
7	1	1	1	1	abc

Shannon decomposition. ***Only 2-to-1 muxes.***



$$M = \bar{a}(bc) + a(b+c) \quad g = bc \quad h = b + c$$

$$g = \bar{b}(0) + b(c) \quad = \bar{b}\cdot\boxed{0} + b\cdot\boxed{c}$$

$$h = b + c \quad = b + (b+\bar{b})c = \bar{b}\,c + b + bc = \bar{b}c + b(1+c) = \bar{b}\cdot\boxed{c} + b\cdot\boxed{1}$$

BV 6.5

For the function

$$f(w_1, w_2, w_3) = \sum m(0, 2, 3, 6)$$

use Shannon's expansion to derive an implementation using a 2-to-1 multiplexer and any necessary gates.

BV 6.5

For the function

$$f(w_1, w_2, w_3) = \sum m(0, 2, 3, 6)$$

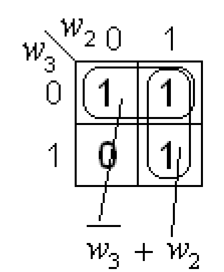
use Shannon's expansion to derive an implementation using a 2-to-1 multiplexer and any necessary gates.

$$f(w_1, w_2, w_3) = \sum m(000, 010, 011, 110) =$$

$$= \overline{w}_1 \overline{w}_2 \overline{w}_3 + \overline{w}_1 w_2 \overline{w}_3 + \overline{w}_1 w_2 w_3 + w_1 w_2 \overline{w}_3 =$$

$$= \overline{w}_1 \boxed{(\overline{w}_2 \overline{w}_3 + w_2 \overline{w}_3 + w_2 w_3)} + w_1 (w_2 \overline{w}_3) =$$

$$(\overline{w}_2 \overline{w}_3 + w_2 \overline{w}_3 + w_2 w_3)$$



BV 6.5

For the function

$$f(w_1, w_2, w_3) = \sum m(0, 2, 3, 6)$$

use Shannon's expansion to derive an implementation using a 2-to-1 multiplexer and any necessary gates.

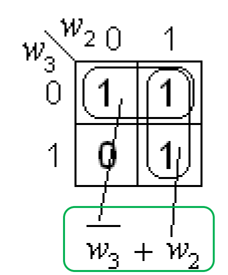
$$f(w_1, w_2, w_3) = \sum m(000, 010, 011, 110) =$$

$$= \overline{w}_1 \overline{w}_2 \overline{w}_3 + \overline{w}_1 w_2 \overline{w}_3 + \overline{w}_1 w_2 w_3 + w_1 w_2 \overline{w}_3 =$$

$$= \overline{w}_1 (\overline{w}_2 \overline{w}_3 + w_2 \overline{w}_3 + w_2 w_3) + w_1 (w_2 \overline{w}_3) =$$

$$= \overline{w}_1 \boxed{(w_2 + \overline{w}_3)} + w_1 (w_2 \overline{w}_3)$$

$$(\overline{w}_2 \overline{w}_3 + w_2 \overline{w}_3 + w_2 w_3)$$



BV 6.5

For the function

$$f(w_1, w_2, w_3) = \sum m(0, 2, 3, 6)$$

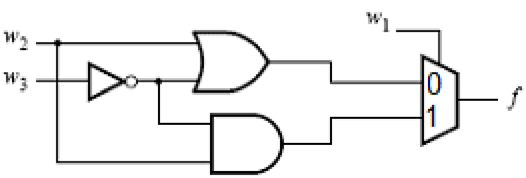
use Shannon's expansion to derive an implementation using a 2-to-1 multiplexer and any necessary gates.

$$f(w_1, w_2, w_3) = \sum m(000, 010, 011, 110) =$$

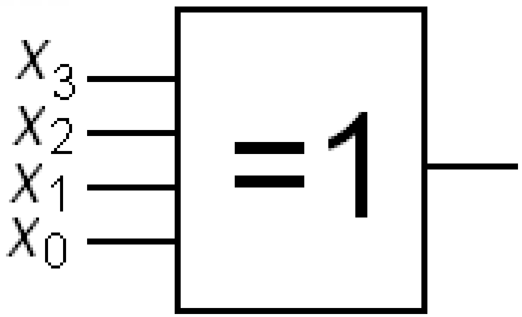
$$= \overline{w}_1 \overline{w}_2 \overline{w}_3 + \overline{w}_1 w_2 \overline{w}_3 + \overline{w}_1 w_2 w_3 + w_1 w_2 \overline{w}_3 =$$

$$= \overline{w}_1 (\overline{w}_2 \overline{w}_3 + w_2 \overline{w}_3 + w_2 w_3) + w_1 (w_2 \overline{w}_3) =$$

$$= \overline{w}_1 (w_2 + \overline{w}_3) + w_1 (w_2 \overline{w}_3)$$



(Ex 8.9)



XOR

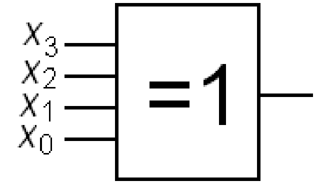
x_3x_2 \ x_1x_0	00	01	11	10
00	$^{0}0$	$^{1}1$	$^{3}0$	$^{2}1$
01	$^{4}1$	$^{5}0$	$^{7}1$	$^{6}0$
11	$^{12}0$	$^{13}1$	$^{15}0$	$^{14}1$
10	$^{8}1$	$^{9}0$	$^{11}1$	$^{10}0$

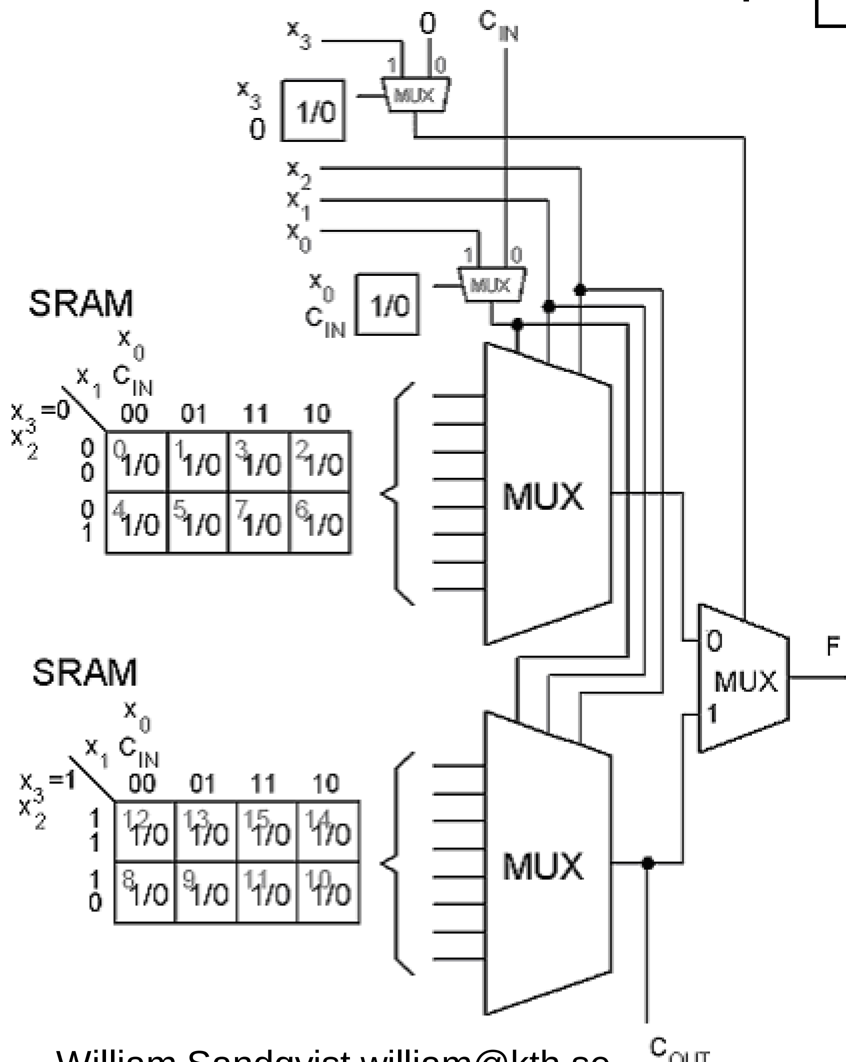
Show how a four-input XOR gate (XOR, odd parity function) is realized in an FPGA circuit. Show the contents of the SRAM cells (LUT, Lookup Table)

(8.9)

XOR

x_3x_2 \ x_1x_0	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0

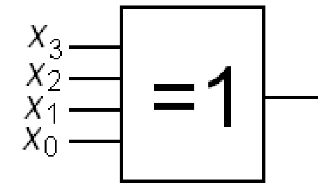




X_3=0

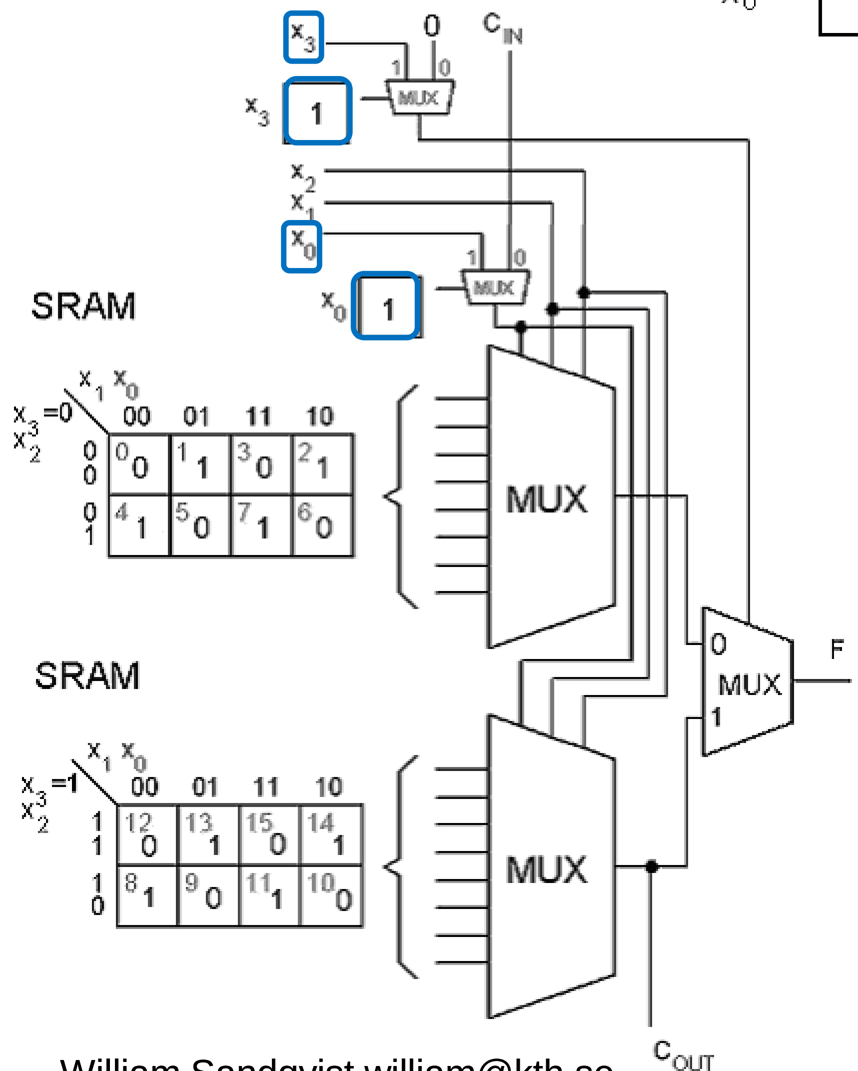
X_3=1

(8.9)

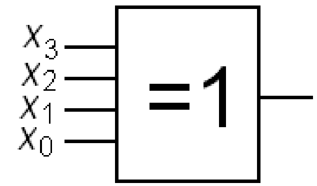


XOR

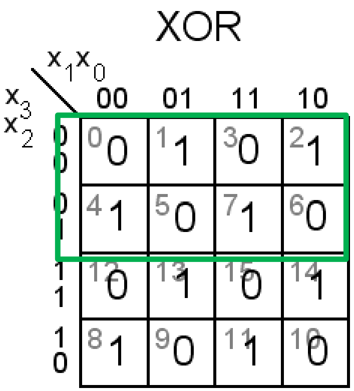
x_3x_2 \ x_1x_0	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0



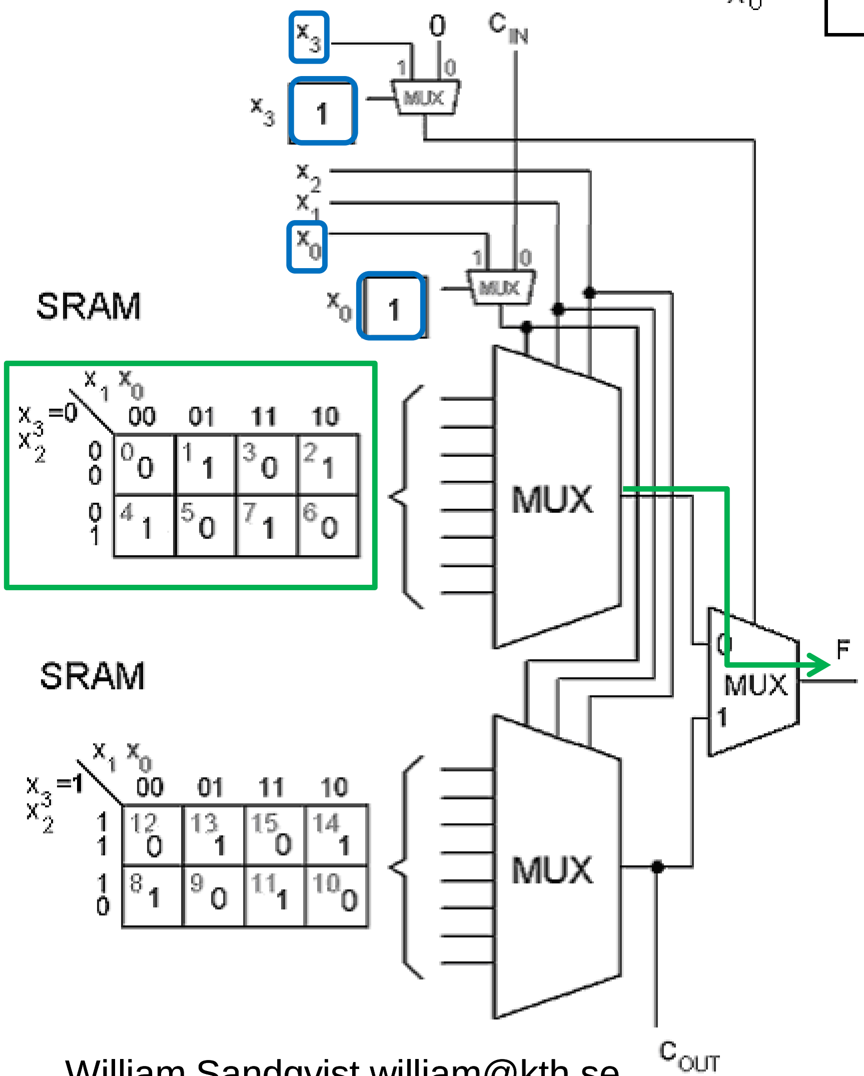
(8.9)



x3
x2
x1
x0
=1

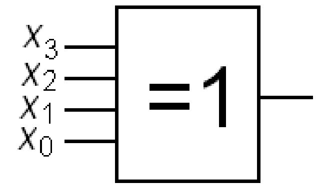


XOR
x1x0
x3x2
00 01 11 10
00: 0 1 0 1
01: 1 0 1 0
11: 0 1 0 1
10: 1 0 1 0



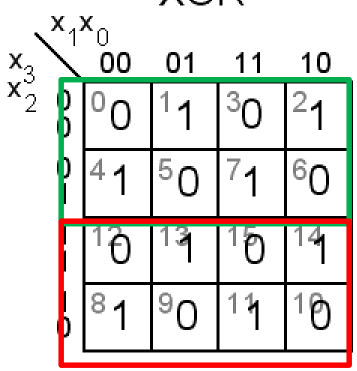
x3
0
C IN
1
0
MUX
x3
1
x2
x1
x0
x0
1
SRAM
x1 x0
x3=0
x3 x2
00 01 11 10
00: 0 1 0 1
01: 1 0 1 0
MUX
SRAM
x1 x0
x3=1
x3 x2
00 01 11 10
11: 0 1 0 1
10: 1 0 1 0
MUX
0
MUX
1
F
C OUT

(8.9)

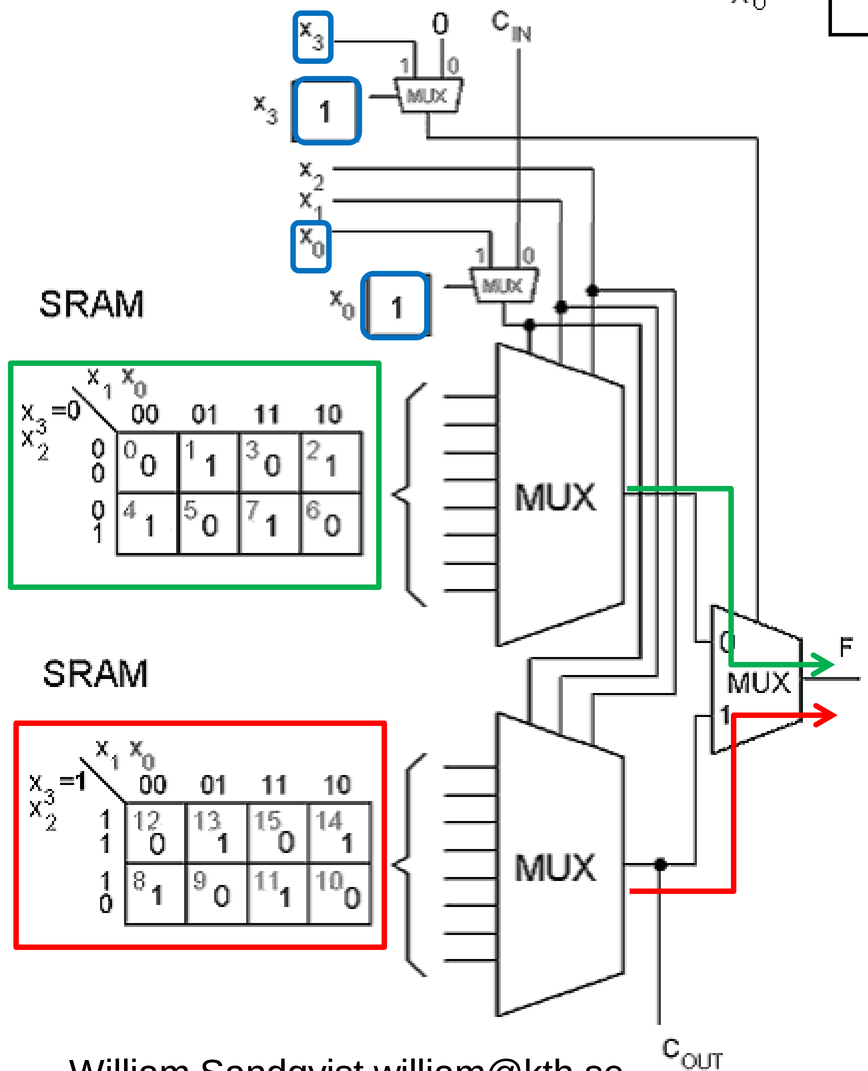


x_3
x_2
x_1
x_0
=1

XOR

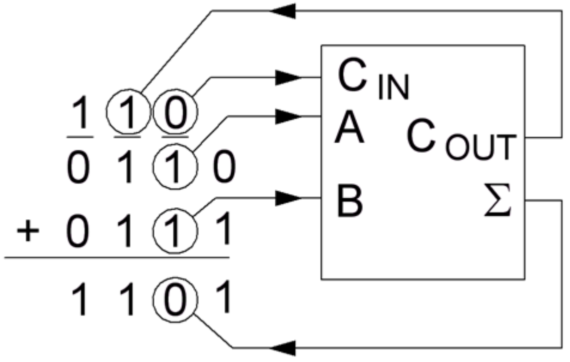


$x_1 x_0$
$x_3 x_2$
00
01
11
10



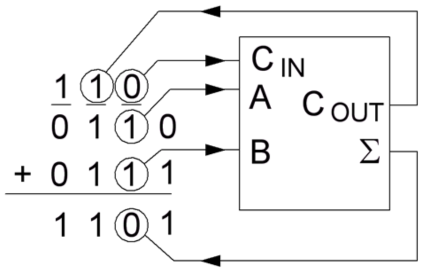
SRAM
SRAM
MUX
MUX
MUX
x_3
x_2
x_1
x_0
C_{IN}
C_{OUT}
F
x_3=0
x_3=1

(Ex 8.8)

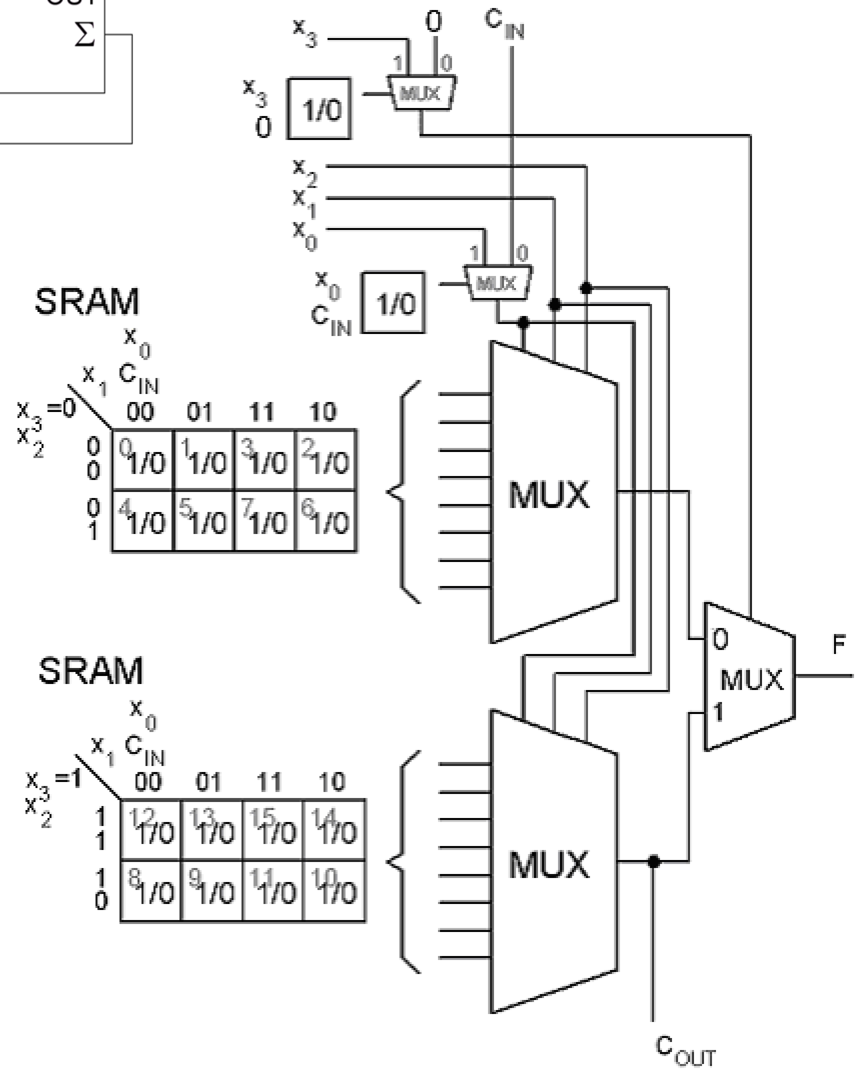


Set up full adder truth table. Show how a full adder is implemented in an FPGA chip. Logic elements of an FPGA is able to cascade COUT and CIN between "neighbors." Show the contents of the SRAM cells (LUT, Lookup Table).

(8.8)

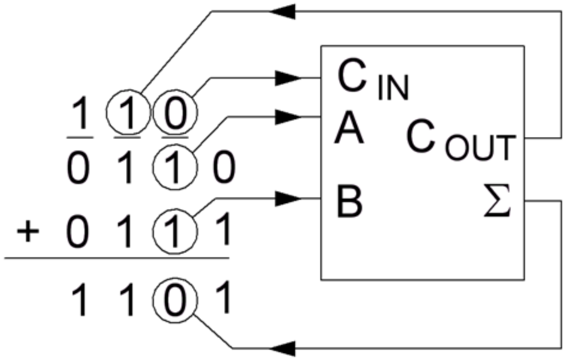


1 1 0
0 1 1 0
+ 0 1 1 1
1 1 0 1
C IN
A
C OUT
B
Σ

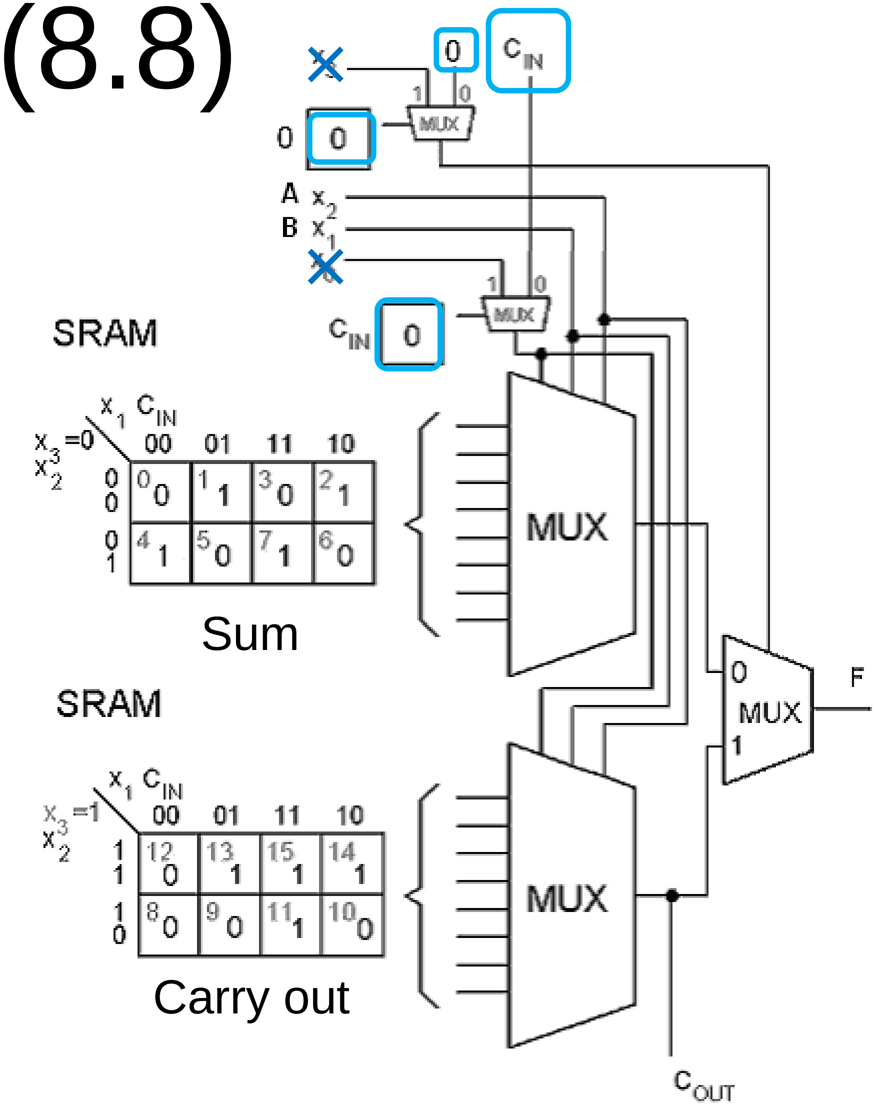


x3
0
CIN
MUX
1/0
x2
x1
x0
SRAM
x3=0
00 01 11 10
MUX
F
x3=1
COUT

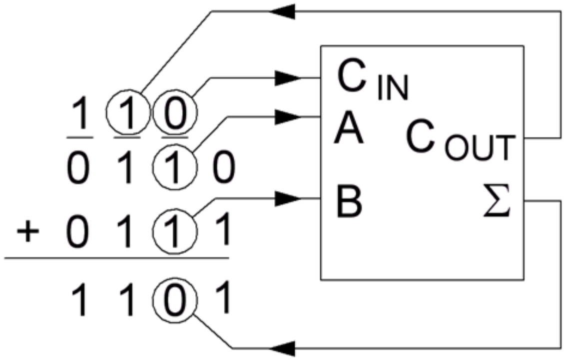
(8.8)



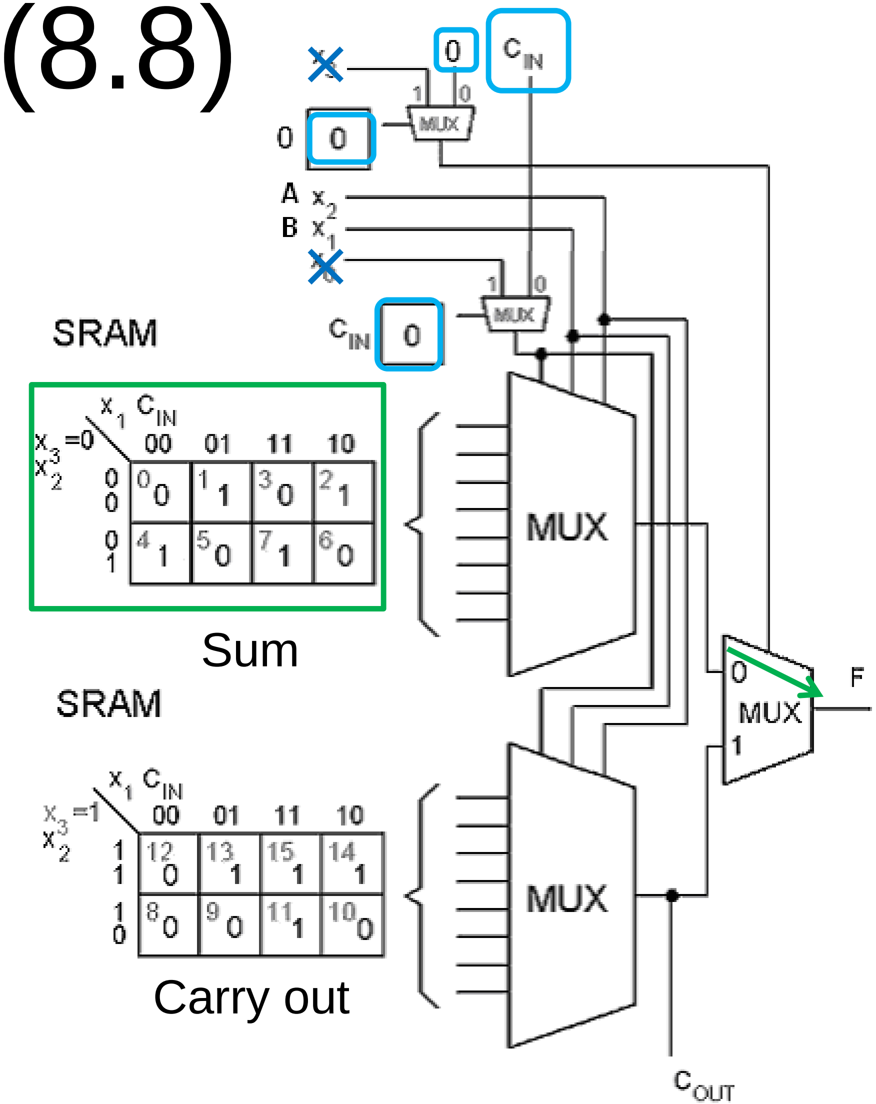
x_2 A	x_1 B	x_0 C_{IN}	Σ	C_{OUT}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



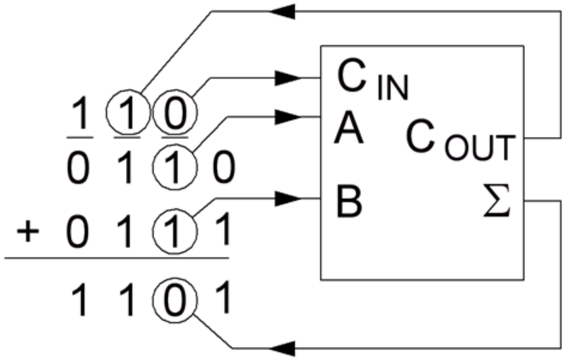
(8.8)



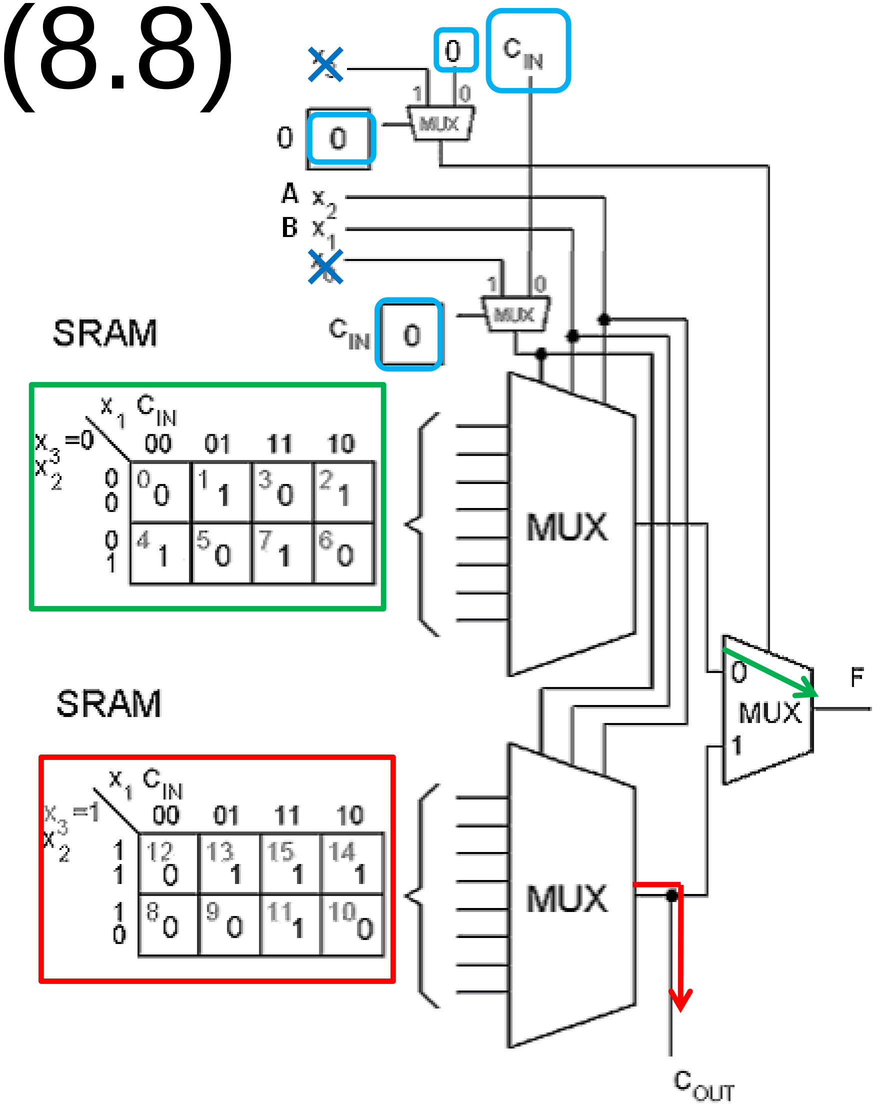
x_2	x_1	x_0		
A	B	C_{IN}	Σ	C_{OUT}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



(8.8)

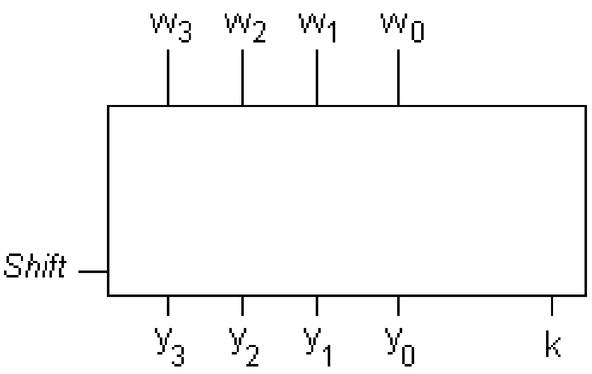


x_2	x_1	x_0		
A	B	C_{IN}	Σ	C_{OUT}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

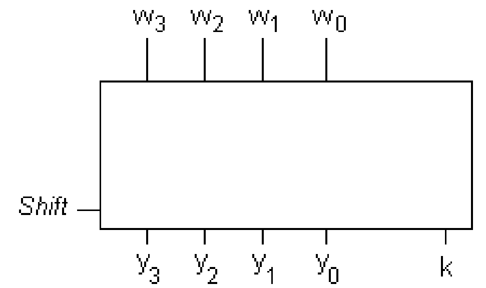


(BV ex 6.31)

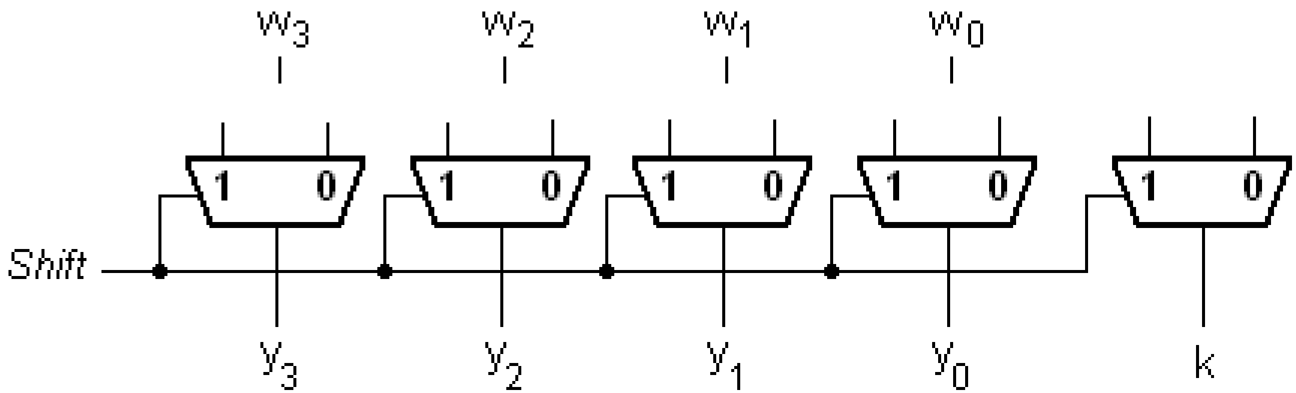
In digital systems it is often necessary to have circuits that can shift the bits of a vector one or more bit positions to the left or right.
Design a circuit that can shift a four-bit vector $W = w_3w_2w_1w_0$ one bit position to the right when a control signal *Shift* is equal to 1.
Let the outputs of the circuit be a four-bit vector $Y = y_3y_2y_1y_0$ and a signal k, such that if *Shift* = 1 then $y_3 = 0$, $y_2 = w_3$, $y_1 = w_2$, $y_0 = w_1$, and $k = w_0$. If *Shift* = 0 then $Y = W$ and $k = 0$.



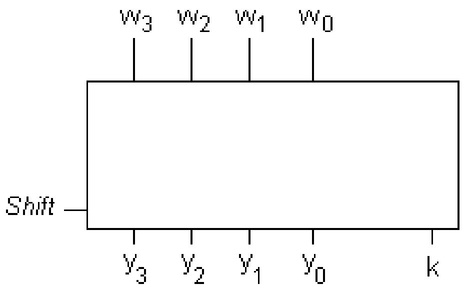
(BV ex 6.31)



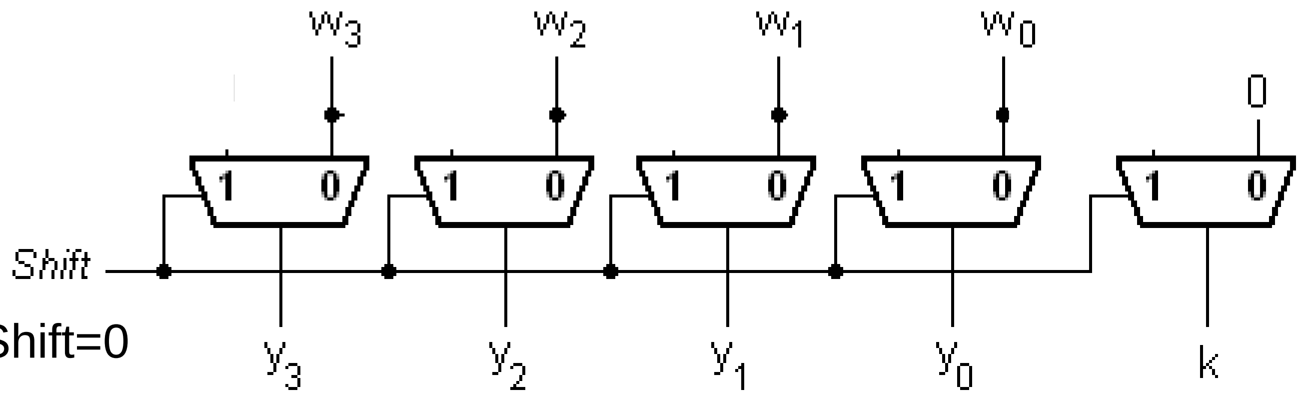
We uses MUXes:



(BV ex 6.31)



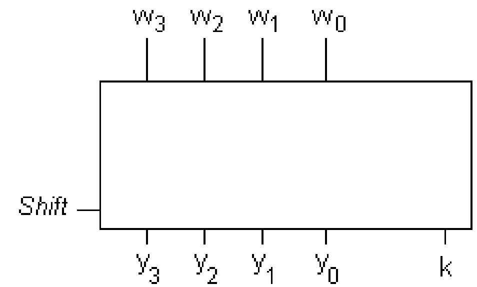
We uses MUXes:



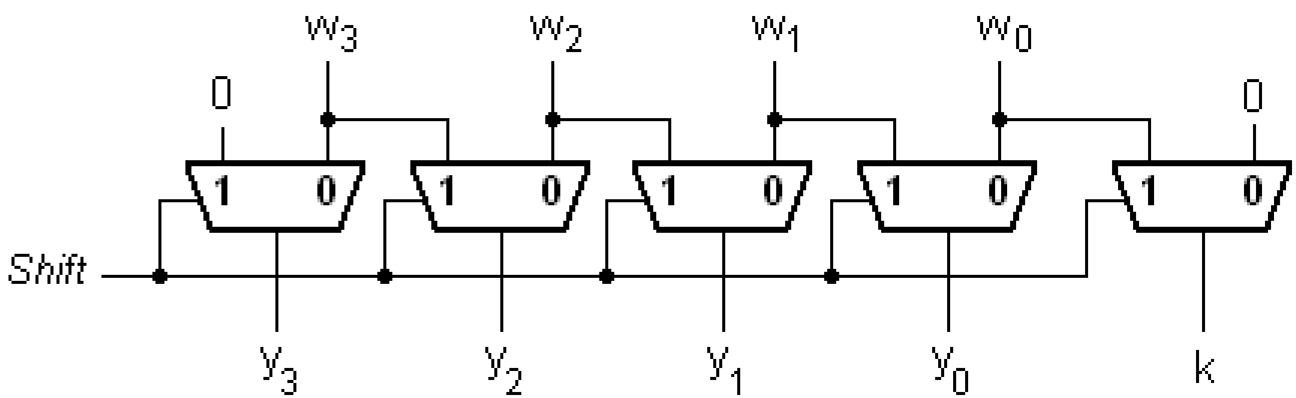
Shift=0

If Shift = 1 then $y_3 = 0$, $y_2 = w_3$, $y_1 = w_2$, $y_0 = w_1$, and $k = w_0$.
If *Shift* = 0 then $y_3 = w_3$, $y_2 = w_2$, $y_1 = w_1$, $y_0 = w_0$, and $k = 0$.

(BV ex 6.31)



We uses MUXes:

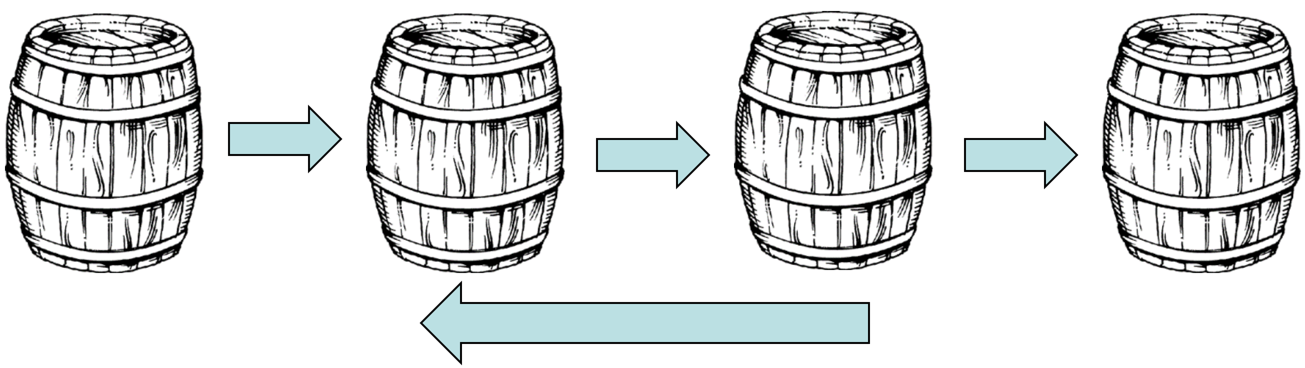


If Shift = 1 then $y_3 = 0$, $y_2 = w_3$, $y_1 = w_2$, $y_0 = w_1$, and $k = w_0$.
If *Shift* = 0 then $y_3 = w_3$, $y_2 = w_2$, $y_1 = w_1$, $y_0 = w_0$, and $k = 0$.

BV ex. 6.32
Barrel shifter

The shifter in Example 6.31 shifts the bits of an input vector by one bit position to the right. It fills the empty bit on the left side with 0. If the bits that are shifted out are placed into the empty position on the left, then the circuit effectively rotates the bits of the input vector by a specified number of bit positions. Such a circuit is called a *barrel shifter*.

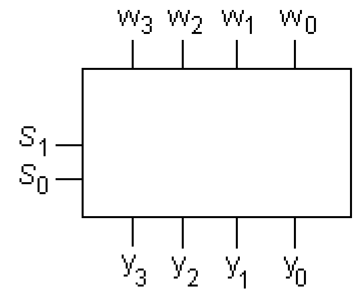
Design a four-bit barrel shifter that rotates the bits by 0, 1, 2, or 3 bit positions as determined by the valuation of two control signals s_1 and s_0.



A barrel shifter is used to speed up floating point operations.

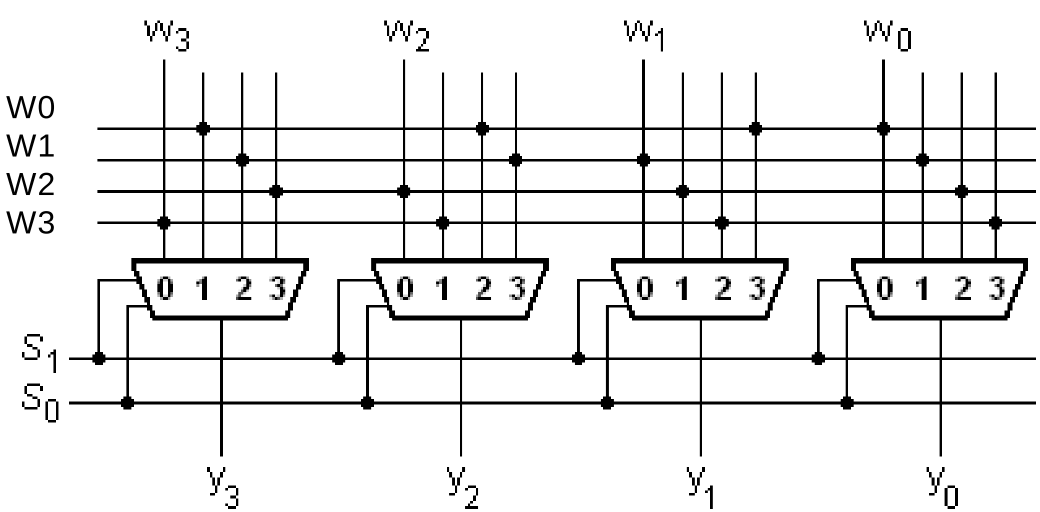
Barrel shifter

BV ex. 6.32

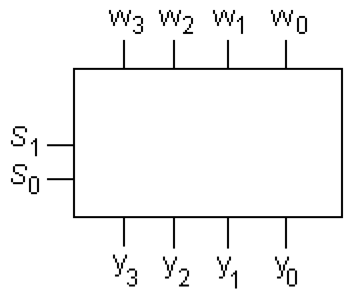


Truth table:

	s_1	s_0	y_3	y_2	y_1	y_0
0	0	0	w_3	w_2	w_1	w_0
1	0	1	w_0	w_3	w_2	w_1
2	1	0	w_1	w_0	w_3	w_2
3	1	1	w_2	w_1	w_0	w_3

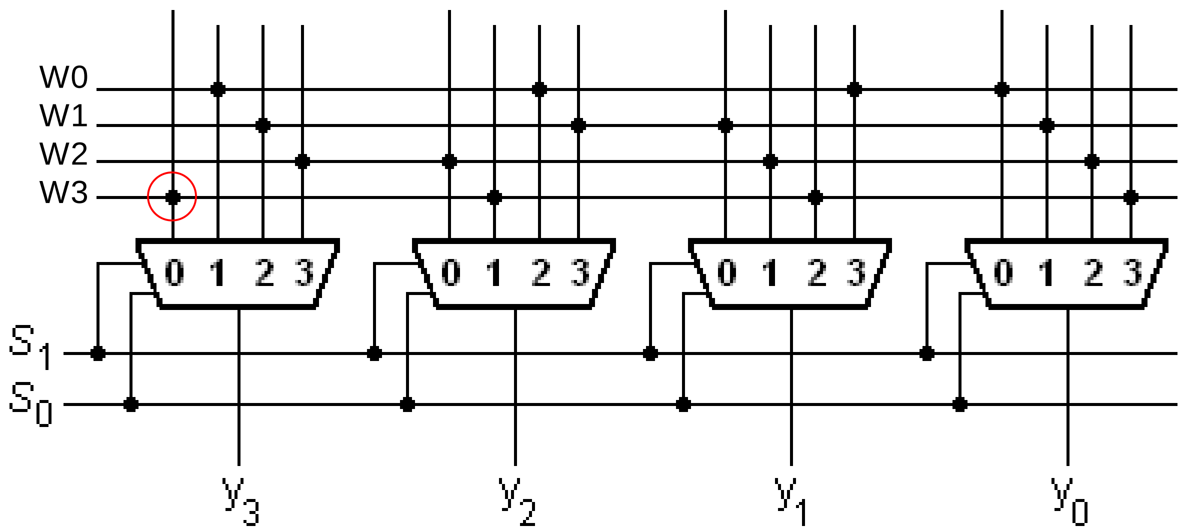


BV ex. 6.32

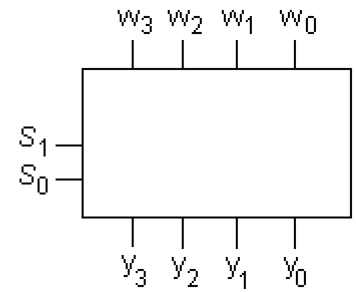


Truth table:

	s_1	s_0	y_3	y_2	y_1	y_0
0	0	0	w_3	w_2	w_1	w_0
1	0	1	w_0	w_3	w_2	w_1
2	1	0	w_1	w_0	w_3	w_2
3	1	1	w_2	w_1	w_0	w_3

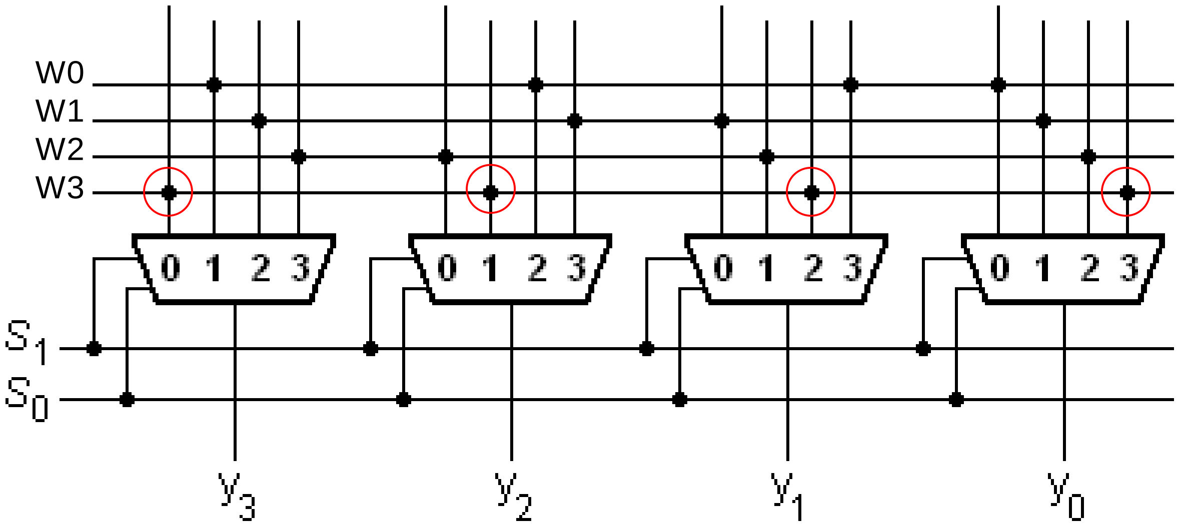


BV ex. 6.32



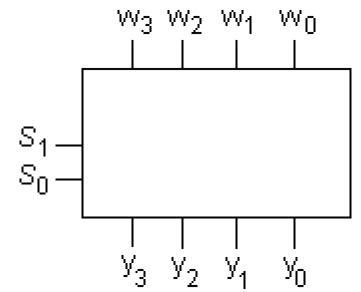
Truth table:

	s_1	s_0	y_3	y_2	y_1	y_0
0	0	0	w_3	w_2	w_1	w_0
1	0	1	w_0	w_3	w_2	w_1
2	1	0	w_1	w_0	w_3	w_2
3	1	1	w_2	w_1	w_0	w_3



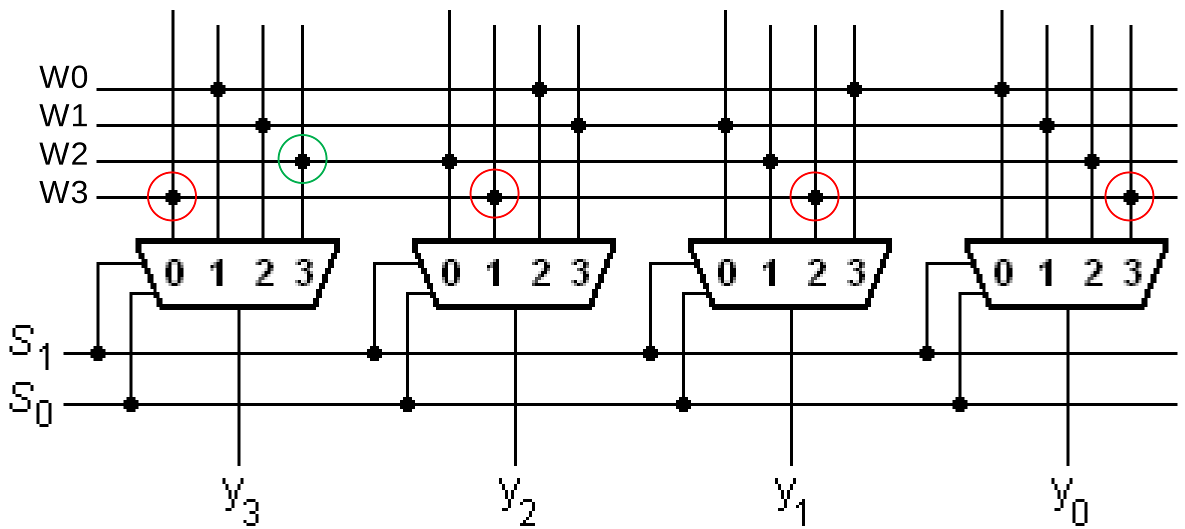
William Sandqvist william@kth.se

BV ex. 6.32

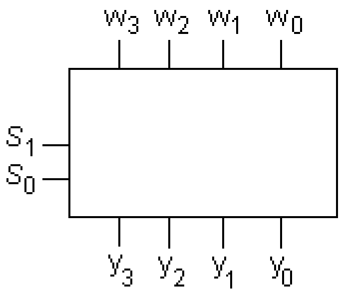


Truth table:

	s_1	s_0	y_3	y_2	y_1	y_0
0	0	0	w_3	w_2	w_1	w_0
1	0	1	w_0	w_3	w_2	w_1
2	1	0	w_1	w_0	w_3	w_2
3	1	1	w_2	w_1	w_0	w_3

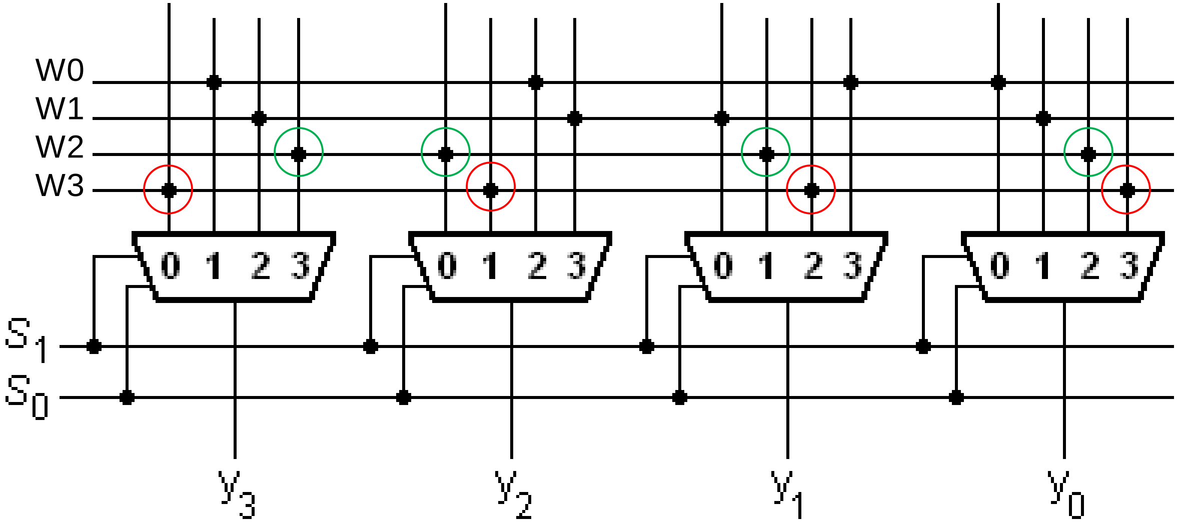


BV ex. 6.32



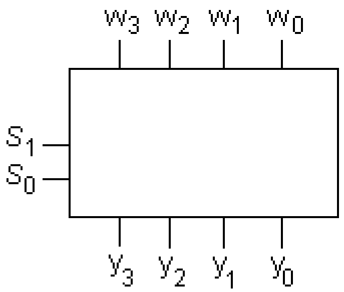
Truth table:

	s_1	s_0	y_3	y_2	y_1	y_0
0	0	0	w_3	w_2	w_1	w_0
1	0	1	w_0	w_3	w_2	w_1
2	1	0	w_1	w_0	w_3	w_2
3	1	1	w_2	w_1	w_0	w_3



William Sandqvist william@kth.se

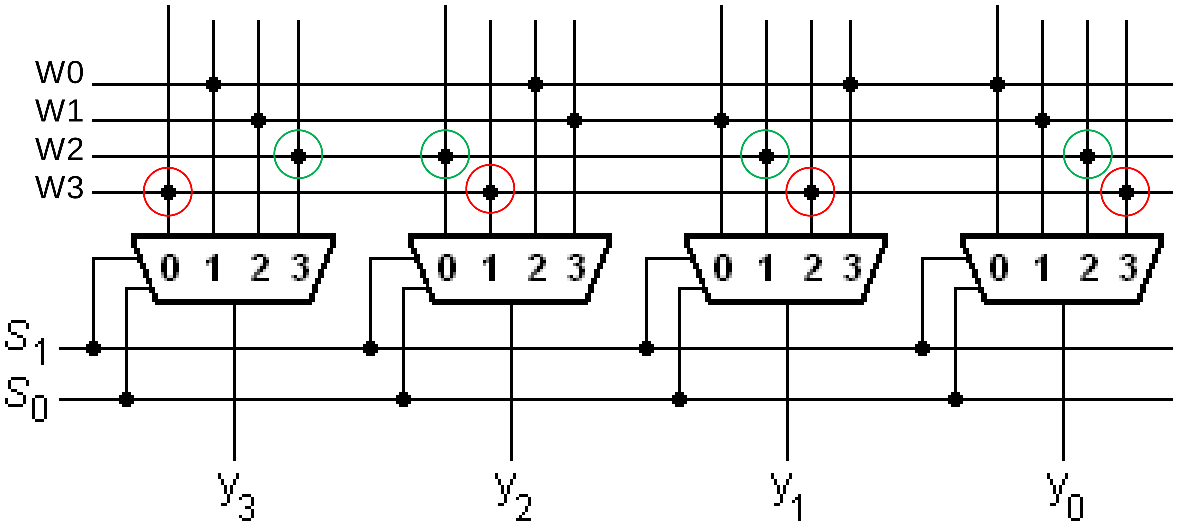
BV ex. 6.32



Truth table:

	s_1	s_0	y_3	y_2	y_1	y_0
0	0	0	w_3	w_2	w_1	w_0
1	0	1	w_0	w_3	w_2	w_1
2	1	0	w_1	w_0	w_3	w_2
3	1	1	w_2	w_1	w_0	w_3

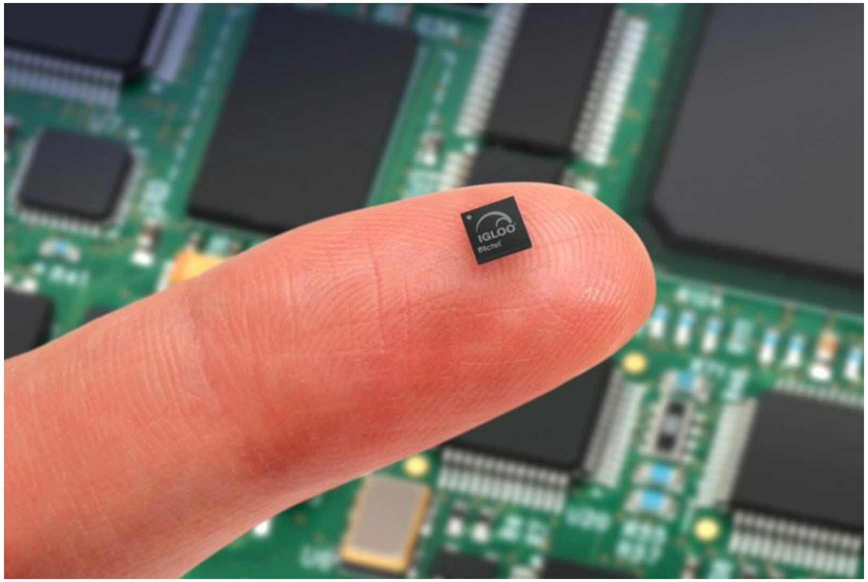
And so on ...



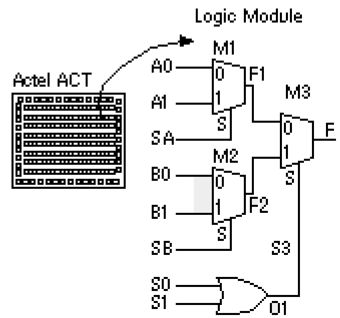
= Lowcost FPGA

Key Benefits

- Lowest FPGA unit cost starts at **$0.49**
- Ultra-low power in Flash*Freeze mode, as low as 2 µW
- Nonvolatile FPGA eliminates unnecessary parts from BOM
- Single-chip and ultra-low-power products simplify board design
- Variety of cost-optimized packages reduce assembly costs
- Low-power FPGAs reduce thermal management and cooling needs



BV 6.16



Actel Corporation manufactures an FPGA family called Act 1, which uses multiplexer based logic blocks. Show how the function

$$f = w_2 \overline{w_3} + w_1 w_3 + \overline{w_2} w_3$$

can be implemented using only ACT 1 logic blocks.

BV 6.16

$$f = w_2 \overline{w_3} + w_1 w_3 + \overline{w_2} w_3$$

BV 6.16

$$f = w_2\,\overline{w_3} + w_1\,w_3 + \overline{w_2}\,w_3$$

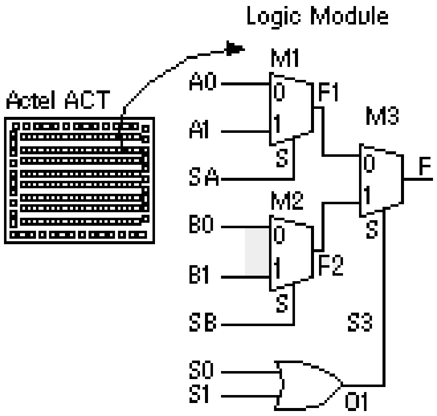
$$f = \overline{w_3}\,(w_2) + w_3\,(w_1 + \overline{w_2})$$

$$\overline{w_3}\,(w_2) = \overline{w_3}\,(\overline{w_2}\cdot 0 + w_2\cdot 1)$$

$$w_3\,(w_1 + \overline{w_2}) = w_3\left(w_1(w_2 + \overline{w_2}) + \overline{w_2}\right) = w_3\left(w_2\,w_1 + \overline{w_2}\,w_1 + \overline{w_2}\right) =$$

$$= w_3\left(w_2\,w_1 + \overline{w_2}\,(w_1 + 1)\right) = w_3\left(\overline{w_2}\cdot 1 + w_2\cdot w_1\right)$$

$$f = \overline{w_3}\,(\overline{w_2}\cdot 0 + w_2\cdot 1) + w_3\left(\overline{w_2}\cdot 1 + w_2\cdot w_1\right)$$



BV 6.16

$$f = w_2\,\overline{w_3} + w_1\,w_3 + \overline{w_2}\,w_3$$

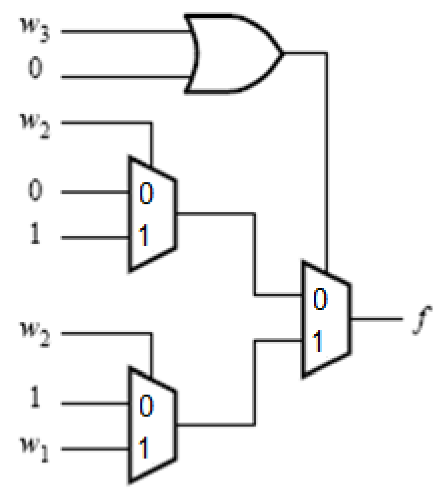
$$f = \overline{w_3}\,(w_2) + w_3\,(w_1 + \overline{w_2})$$

$$\overline{w_3}\,(w_2) = \overline{w_3}\,(\overline{w_2}\cdot 0 + w_2\cdot 1)$$

$$w_3\,(w_1 + \overline{w_2}) = w_3\left(w_1(w_2 + \overline{w_2}) + \overline{w_2}\right) = w_3\left(w_2\,w_1 + \overline{w_2}\,w_1 + \overline{w_2}\right) =$$

$$= w_3\left(w_2\,w_1 + \overline{w_2}\,(w_1 + 1)\right) = w_3\left(\overline{w_2}\cdot 1 + w_2\cdot w_1\right)$$

$$f = \overline{w_3}\,(\overline{w_2}\cdot 0 + w_2\cdot 1) + w_3\left(\overline{w_2}\cdot 1 + w_2\cdot w_1\right)$$



BV 6.16

$$f = w_2\,\overline{w_3} + w_1\,w_3 + \overline{w_2}\,w_3$$

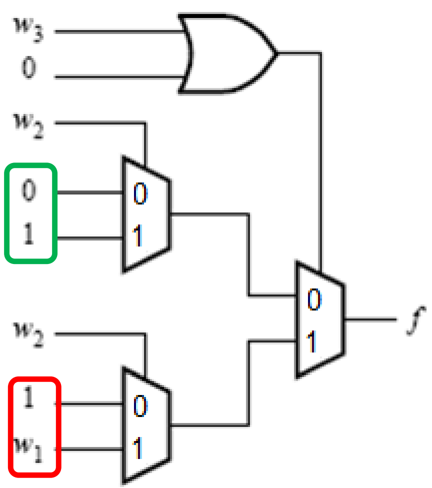
$$f = \overline{w_3}\,(w_2) + w_3\,(w_1 + \overline{w_2})$$

$$\overline{w_3}\,(w_2) = \overline{w_3}\,(\overline{w_2}\cdot 0 + w_2\cdot 1)$$

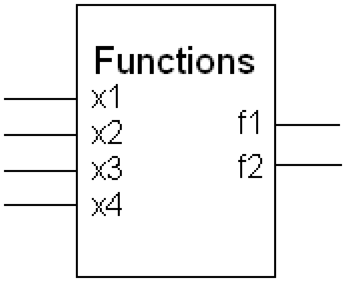
$$w_3\,(w_1 + \overline{w_2}) = w_3\left(w_1(w_2 + \overline{w_2}) + \overline{w_2}\right) = w_3\left(w_2\,w_1 + \overline{w_2}\,w_1 + \overline{w_2}\right) =$$

$$= w_3\left(w_2\,w_1 + \overline{w_2}\,(w_1 + 1)\right) = w_3\left(\overline{w_2}\cdot 1 + w_2\cdot w_1\right)$$

$$f = \overline{w_3}\,(\overline{w_2}\cdot\boxed{0} + w_2\,\boxed{1}) + w_3\left(\overline{w_2}\,\boxed{1} + w_2\cdot\boxed{w_1}\right)$$



VHDL BV 2.51a



Write VHDL code to describe the following functions

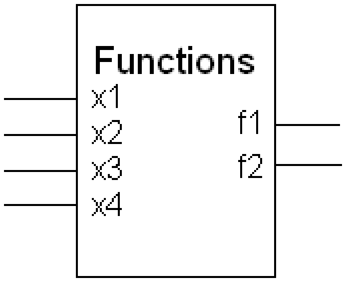
$$f_1 = x_1 \overline{x_3} + x_2 \overline{x_3} + \overline{x_3}\,\overline{x_4} + x_1 x_2 + x_1 \overline{x_4}$$

$$f_2 = (x_1 + \overline{x_3}) \cdot (x_1 + x_2 + \overline{x_4}) \cdot (x_2 + \overline{x_3} + \overline{x_4})$$

VHDL code is written with a text editor and saved in a file with the extension `.vhd`. The code always consists of two sections ENTITY and ARCHITECTURE.

Entity is a description of how the circuit "looks from the outside" (the interface), and Architecture how it "looks like inside."

VHDL BV 2.51a



$$f_1 = x_1\overline{x_3} + x_2\overline{x_3} + \overline{x_3}\,\overline{x_4} + x_1x_2 + x_1\overline{x_4}$$

$$f_2 = (x_1 + \overline{x_3})\cdot(x_1 + x_2 + \overline{x_4})\cdot(x_2 + \overline{x_3} + \overline{x_4})$$

Program code is written with a text editor. So we can only do text comments to the code. A fixed-width font is used (eg. `Courier New`).

Comments begin with

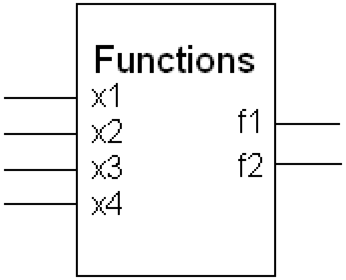
`--`

If you wish, you can "draw" clarification ASCII graphics in the comment lines..

One usually indent text blocks that belong together for greater clarity.

```
--       _____________
--      |             |
--      | Functions   |
--   ->-| x1          |
--   ->-| x2       f1 |->-
--   ->-| x3       f2 |->-
--   ->-| x4          |
--      |_____________|
--
```

VHDL BV 2.51a



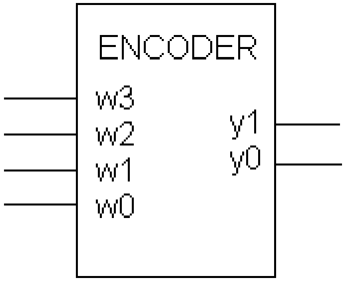
$$f_1 = x_1 \overline{x_3} + x_2 \overline{x_3} + \overline{x_3}\,\overline{x_4} + x_1 x_2 + x_1 \overline{x_4}$$

$$f_2 = (x_1 + \overline{x_3}) \cdot (x_1 + x_2 + \overline{x_4}) \cdot (x_2 + \overline{x_3} + \overline{x_4})$$

```
ENTITY Functions IS
   PORT(x1, x2, x3, x4 :IN  STD_LOGIC;
        f1, f2,        :OUT STD_LOGIC )
END Functions

ARCHITECTURE LogicFunc OF Functions IS
BEGIN
   f1 <= (x1 AND NOT x3)OR(x2 AND NOT x3)OR
         (NOT x3 AND NOT x4)OR(x1 AND x2)OR
         (x1 AND NOT x4);
   f2 <= (x1 OR NOT x3)AND(x1 OR x2 OR NOT x4)AND
         (x2 OR NOT x3 OR NOT x4);
END LogicFunc ;
```

VHDL BV 6.21



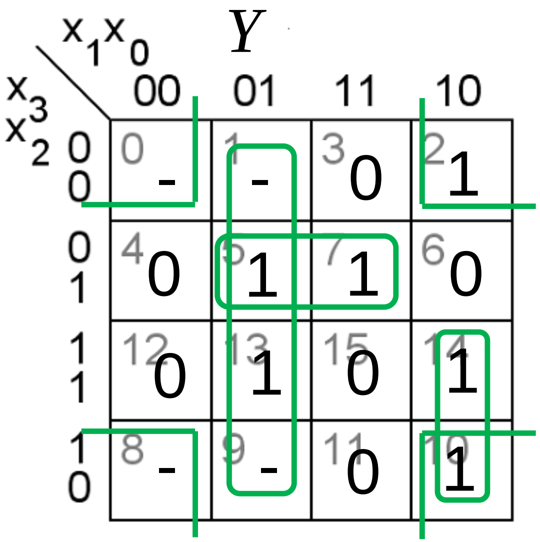
Using a **selected** signal assignement, write VHDL code for a 4-to-2 binary encoder. Only one of w0 …w3 is ”1” at a time.

```
LIBRARY ieee;
USE IEEE.std_logic_1164.all;
ENTITY ENCODER IS
   PORT( w :IN  STD_LOGIC_VECTOR( 3 DOWNTO 0 )   ;
         y :OUT STD_LOGIC_VECTOR( 1 DOWNTO 0 ) );
END ENCODER
ARCHITECTURE Behavior OF ENCODER IS
BEGIN
   WITH w SELECT
      y <= ”00” WHEN ”0001”,
           ”01” WHEN ”0010”,
           ”10” WHEN ”0100”,
           ”11” WHEN OTHERS;
END Behavior ;
```

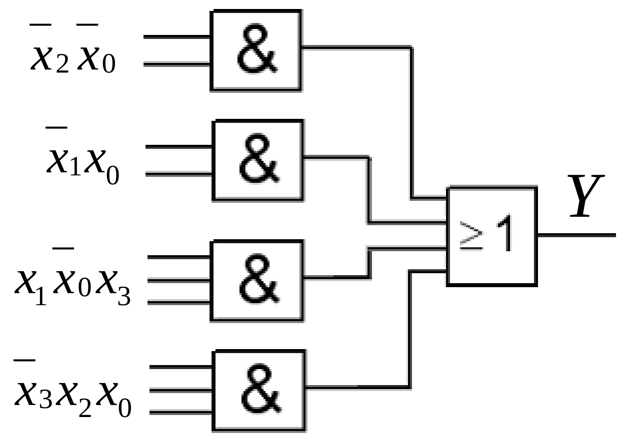
(8.10) Additional if time permits

Y

$x_3x_2 \backslash x_1x_0$	00	01	11	10
00	0: -	1: -	3: 0	2: 1
01	4: 0	5: 1	7: 1	6: 0
11	12: 0	13: 1	15: 0	14: 1
10	8: -	9: -	11: 0	10: 1



$$Y = \bar{x}_2\bar{x}_0 + \bar{x}_1 x_0 + x_1\bar{x}_0 x_3 + \bar{x}_3 x_2 x_0$$



Y

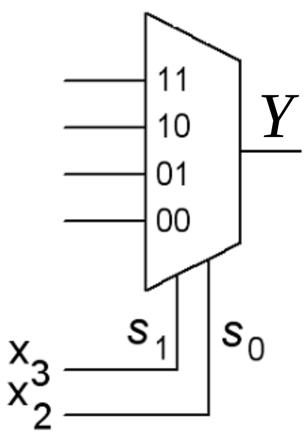
x_3x_2 \ x_1x_0	00	01	11	10
00	[0] -	[1] -	[3] 0	[2] 1
01	[4] 0	[5] 1	[7] 1	[6] 0
11	[12] 0	[13] 1	[15] 0	[14] 1
10	[8] -	[9] -	[11] 0	[10] 1

x_1 \ x_0	0	1
0		
1		

$x_3x_2(0,0) \Rightarrow$

x_1 \ x_0	0	1
0		
1		

$x_3x_2(0,1) \Rightarrow$



x_1 \ x_0	0	1
0		
1		

$x_3x_2(1,1) \Rightarrow$

x_1 \ x_0	0	1
0		
1		

$x_3x_2(1,0) \Rightarrow$

Y

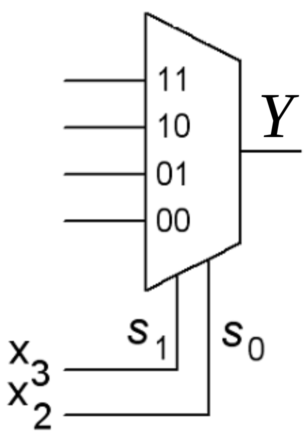
$x_3x_2 \backslash x_1x_0$	00	01	11	10
00	0: -	1: -	3: 0	2: 1
01	4: 0	5: 1	7: 1	6: 0
11	12: 0	13: 1	15: 0	14: 1
10	8: -	9: -	11: 0	10: 1

$x_1 \backslash x_0$	0	1
0	-	-
1	1	0

$x_3x_2(0,0) \Rightarrow$

$x_1 \backslash x_0$	0	1
0	0	1
1	0	1

$x_3x_2(0,1) \Rightarrow$



$x_1 \backslash x_0$	0	1
0	0	1
1	1	0

$x_3x_2(1,1) \Rightarrow$

$x_1 \backslash x_0$	0	1
0	-	-
1	1	0

$x_3x_2(1,0) \Rightarrow$

Y

x_3x_2 \ x_1x_0	00	01	11	10
00	0 -	1 -	3 0	2 1
01	4 0	5 1	7 1	6 0
11	12 0	13 1	15 0	14 1
10	8 -	9 -	11 0	10 1

$$Y = \bar{x}_2\bar{x}_0 + \bar{x}_1 x_0 + x_1\bar{x}_0 x_3 + \bar{x}_3 x_2 x_0$$

x_1 \ x_0	0	1
0	-	-
1	1	0

$x_3x_2(0,0) \Rightarrow$

$Y = \bar{x}_0$

x_1 \ x_0	0	1
0	0	1
1	0	1

$x_3x_2(0,1) \Rightarrow$

$Y = x_0$

x_1 \ x_0	0	1
0	0	1
1	1	0

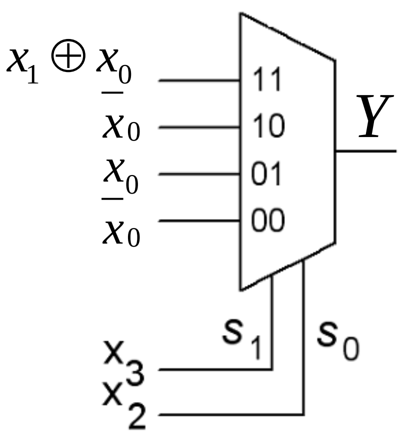
$x_3x_2(1,1) \Rightarrow$

$Y = x_1 \oplus x_0$

x_1 \ x_0	0	1
0	-	-
1	1	0

$x_3x_2(1,0) \Rightarrow$

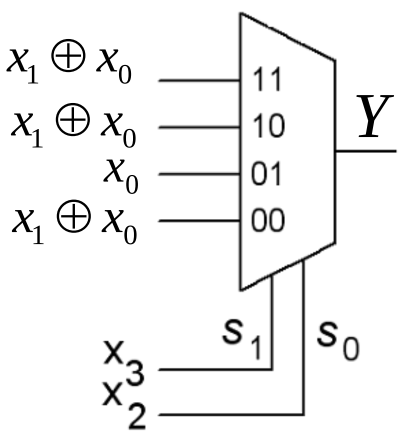
$Y = \bar{x}_0$



William Sandqvist william@kth.se

$x_3x_2 \backslash x_1x_0$	00	01	11	10
00	0: -	1: -	3: 0	2: 1
01	4: 0	5: 1	7: 1	6: 0
11	12: 0	13: 1	15: 0	14: 1
10	8: -	9: -	11: 0	10: 1

$$Y = \bar{x}_2\bar{x}_0 + \bar{x}_1 x_0 + x_1\bar{x}_0 x_3 + \bar{x}_3 x_2 x_0$$



Or …

$x_1 \backslash x_0$	0	1
0	-	-
1	1	0

$x_3x_2(0,0) \Rightarrow$

$Y = x_1 \oplus x_0$

$x_1 \backslash x_0$	0	1
0	0	1
1	0	1

$x_3x_2(0,1) \Rightarrow$

$Y = x_0$

$x_1 \backslash x_0$	0	1
0	0	1
1	1	0

$x_3x_2(1,1) \Rightarrow$

$Y = x_1 \oplus x_0$

$x_1 \backslash x_0$	0	1
0	-	-
1	1	0

$x_3x_2(1,0) \Rightarrow$

$Y = x_1 \oplus x_0$

Or if you don't have acess to the variable x_0 inverted …