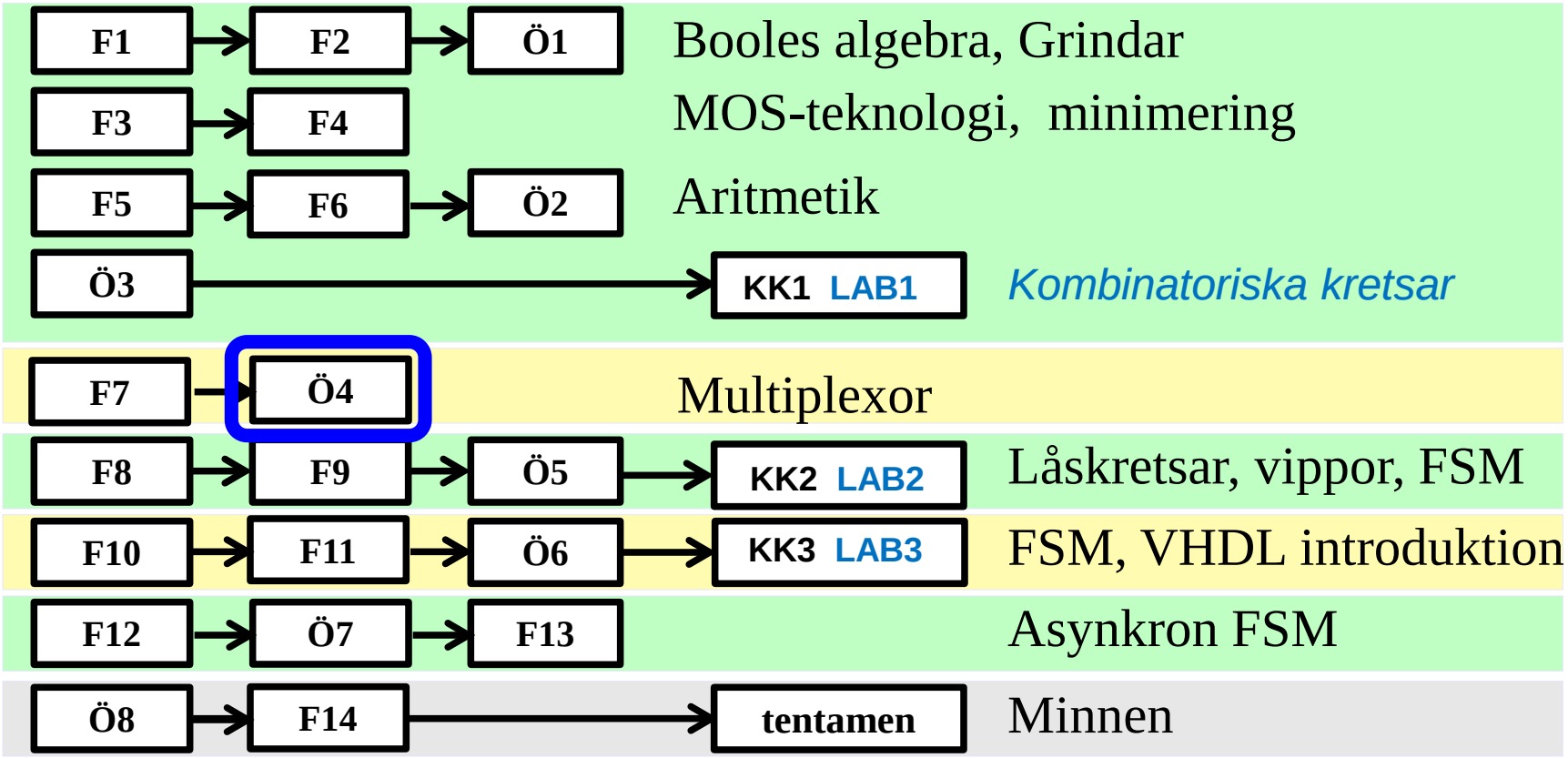
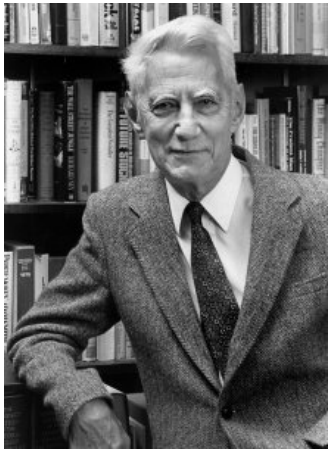
IE1204 Digital Design



Föreläsningar och övningar bygger på varandra! Ta alltid igen det Du missat!
Läs på i förväg – delta i undervisningen – arbeta igenom materialet efteråt!

William Sandqvist william@kth.se

Shannon dekomposition

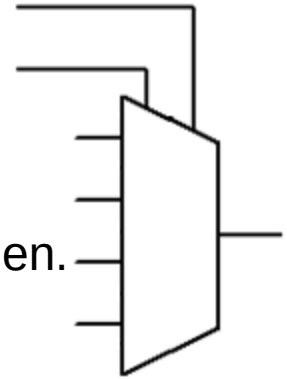


Claude Shannon
matematiker/elektrotekniker
(1916 –2001)

William Sandqvist william@kth.se

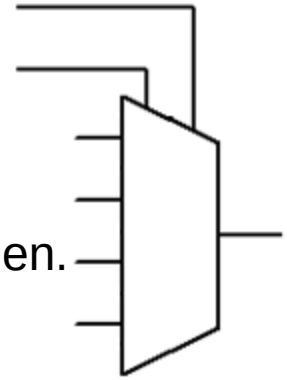
(ÖH 8.6)

Visa hur en 4-to-1 **multiplexor** kan användas som ”funktionsgenerator” för att tex. Generera OR-funktionen.

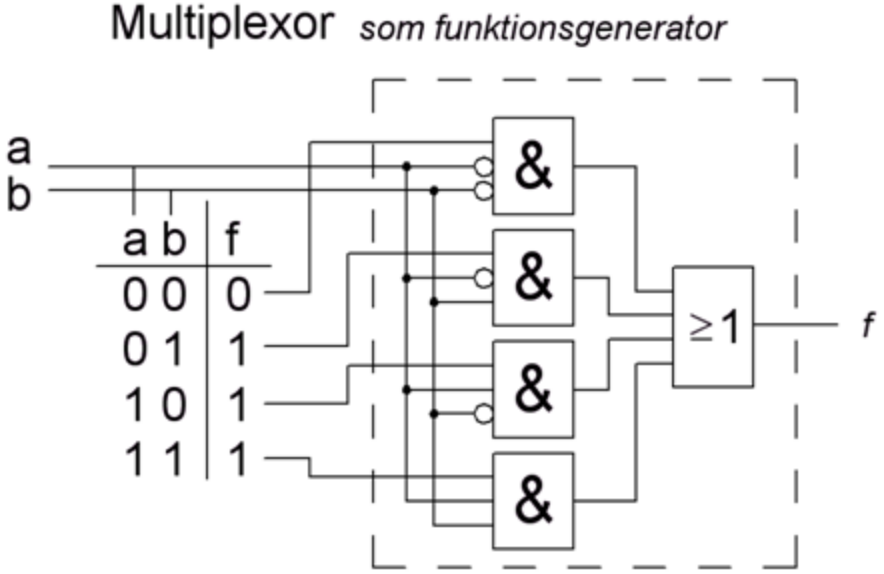


William Sandqvist william@kth.se

(ÖH 8.6)

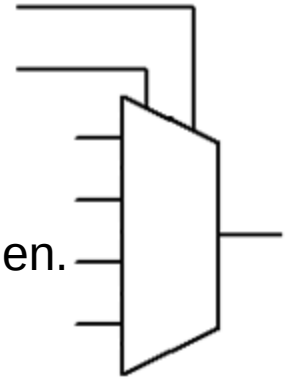


Visa hur en 4-to-1 **multiplexor** kan användas som ”funktionsgenerator” för att tex. Generera OR-funktionen.

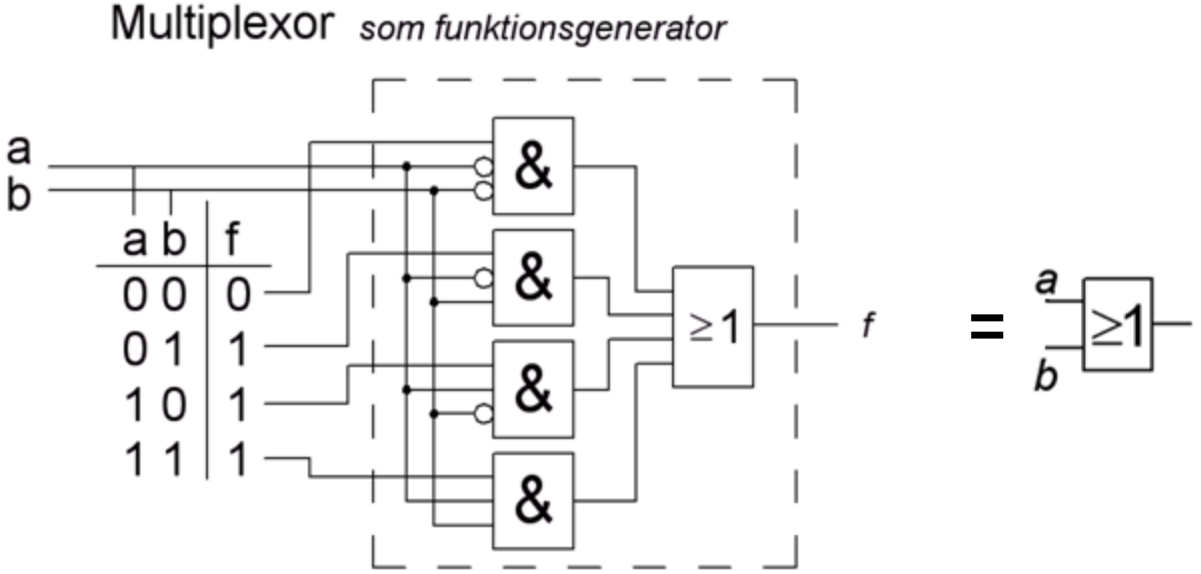


William Sandqvist william@kth.se

(ÖH 8.6)



Visa hur en 4-to-1 **multiplexor** kan användas som ”funktionsgenerator” för att tex. Generera OR-funktionen.



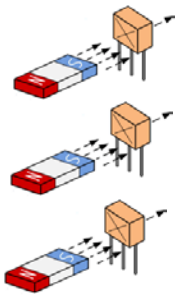
William Sandqvist william@kth.se

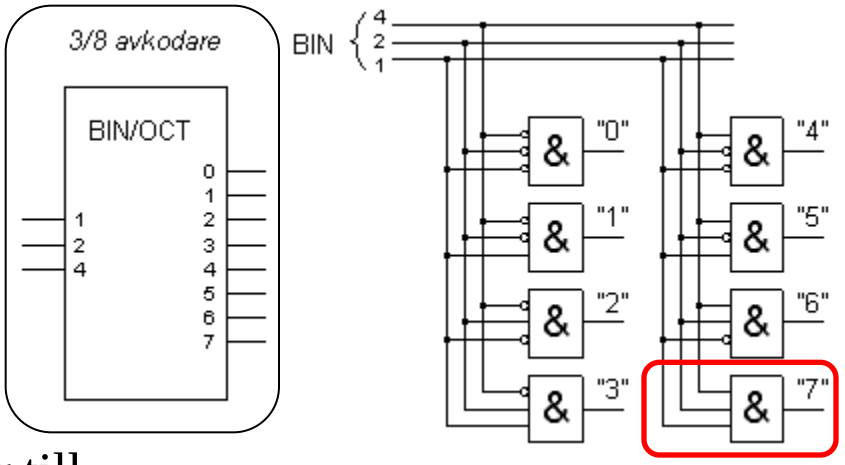
William Sandqvist william@kth.se

x/y avkodare

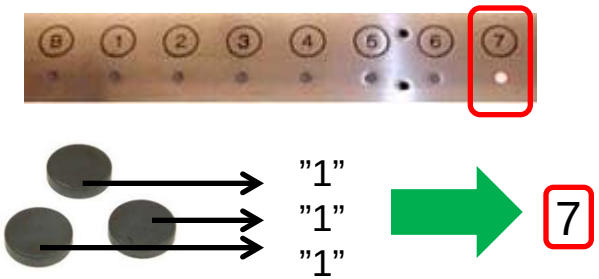
Ofta samlar man många avkodare till ett funktionsblock. En x/y-avkodare har x ingångssignaler som avkodas till y utsignaler.
I figuren visas en 3/8-avkodare, även kallad BIN/OCT-avkodare.

Ex.
Magneter och Hallsensorer





Kanske en våningsindikator till en hiss? (Med en grupp av tre magneter kan man överföra till hisskorgen vilken våning man befinner sig på).



William Sandqvist william@kth.se

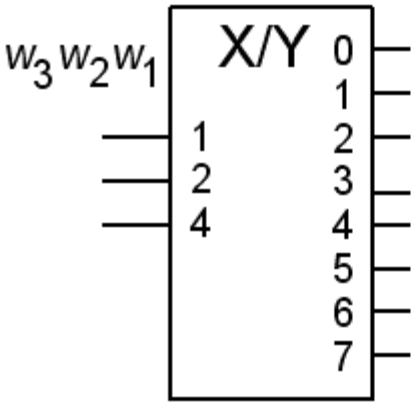
BV 6.1

Show how the function

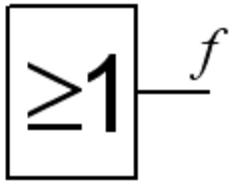
$$f(w_1, w_2, w_3) = \sum m(0, 2, 3, 4, 5, 7)$$

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

3-to-8 decoder



OR gate



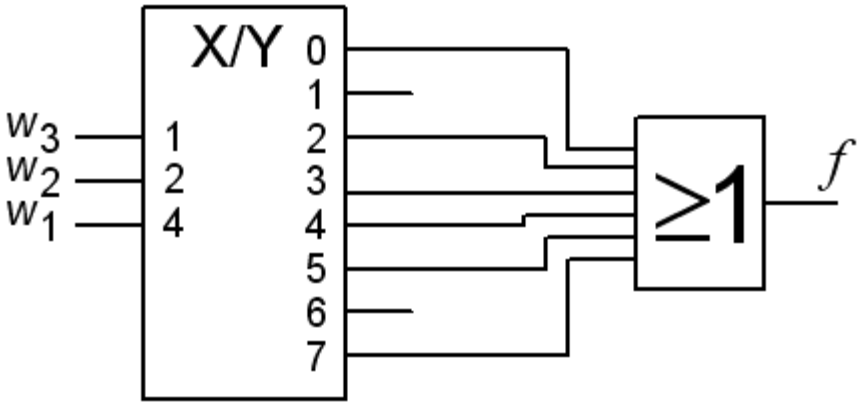
William Sandqvist william@kth.se

BV 6.1

Show how the function

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

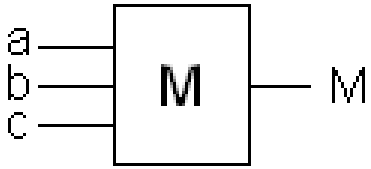
can be implemented using a 3-to-8 **decoder** and an OR gate.



William Sandqvist william@kth.se

William Sandqvist william@kth.se

ÖH 8.7



En majoritetsgrind antar på utgången samma värde som en *majoritet* av ingångarna. Grinden kan tex. användas i feltolerant logik, eller till bildbehandlingskretsar.

a) (Ställ upp grindens sanningstabell och minimera funktionen med Karnaughdiagram. Realisera funktionen med AND-OR grindar.)

b) Realisera majoritetsgrinden med en 8:1 MUX.

c) Använd Shannon dekomposition och realisera majoritetsgrinden med en 2:1 MUX och grindar.

d) Realisera majorotetsgrinden med bara 2:1 MUXar.

William Sandqvist william@kth.se

(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

Med AND OR grindar

William Sandqvist william@kth.se

(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

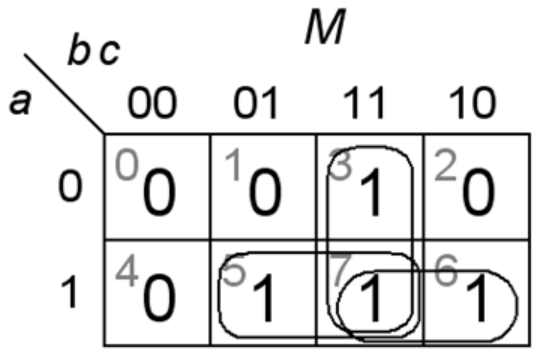
$\bar{a}bc$

$a\bar{b}c$

$ab\bar{c}$

abc

Med AND OR grindar

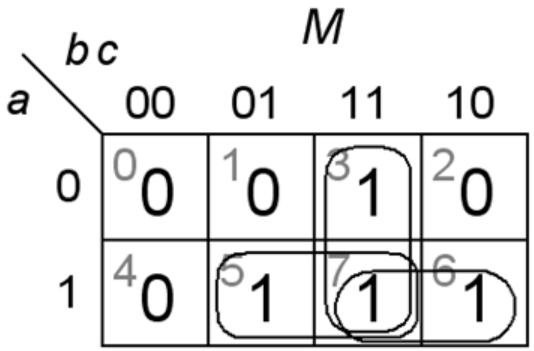


William Sandqvist william@kth.se

(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

Med AND OR grindar



$\bar{a}bc$

$a\bar{b}c$

$ab\bar{c}$

abc

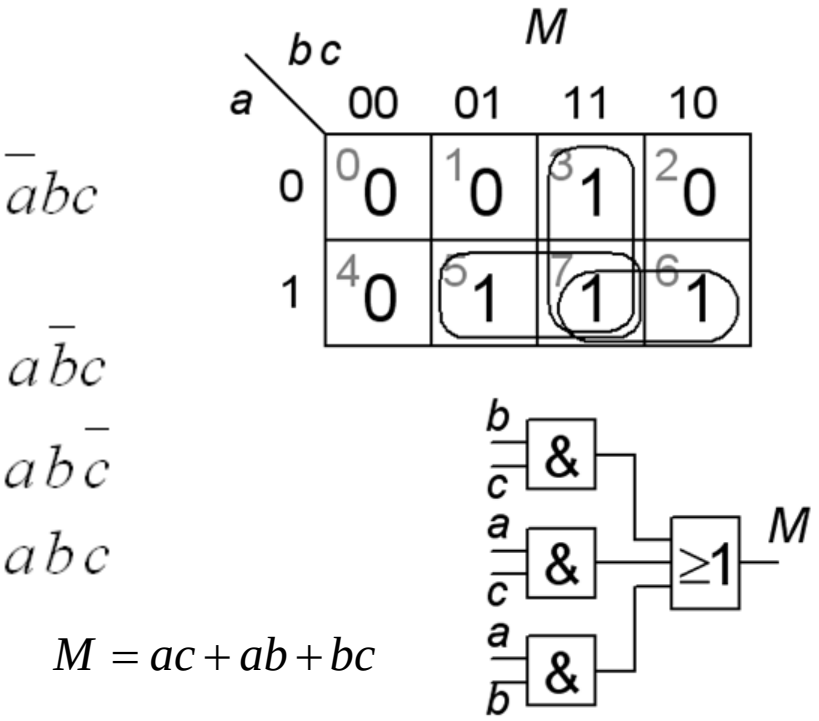
$$M = ac + ab + bc$$

William Sandqvist william@kth.se

(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

Med AND OR grindar



$$M = ac + ab + bc$$

William Sandqvist william@kth.se

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

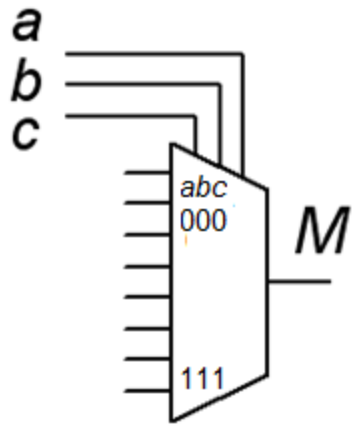
Med 8-to-1 mux …

William Sandqvist william@kth.se

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

Med 8-to-1 mux …

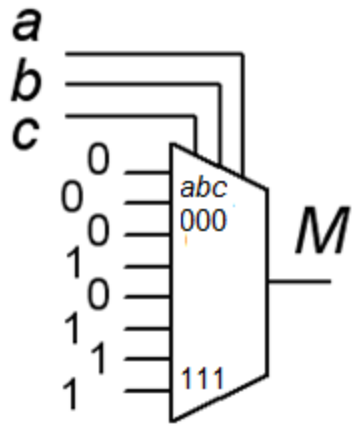


William Sandqvist william@kth.se

8.7b

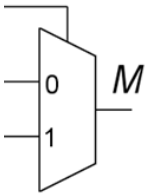
	a	b	c	M
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

Med 8-to-1 mux …



William Sandqvist william@kth.se

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 dekomposition. 2-to-1 mux och grindar.

$\bar{a}bc$

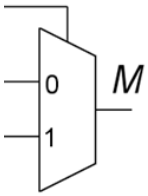
$a\bar{b}c$

$ab\bar{c}$

abc

William Sandqvist william@kth.se

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 dekomposition. 2-to-1 mux och grindar.

$\bar{a}bc$

$a\bar{b}c$

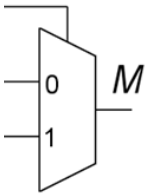
$ab\bar{c}$

abc

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

William Sandqvist william@kth.se

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 dekomposition. 2-to-1 mux och grindar.

$\bar{a}bc$

$a\bar{b}c$

$ab\bar{c}$

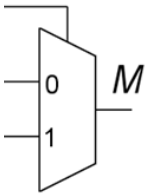
abc

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

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

William Sandqvist william@kth.se

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 dekomposition. 2-to-1 mux och grindar.

$\bar{a}bc$

$a\bar{b}c$

$ab\bar{c}$

abc

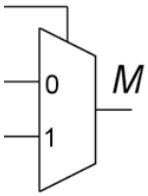
$$M = \bar{a}bc + a\bar{b}c + ab\bar{c} + abc =$$
$$= \bar{a}(bc) + a(\bar{b}c + b\bar{c} + bc) =$$

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

OR

William Sandqvist william@kth.se

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 dekomposition. 2-to-1 mux och grindar.

$\bar{a}bc$

$a\bar{b}c$

$ab\bar{c}$

abc

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

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

OR

William Sandqvist william@kth.se

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

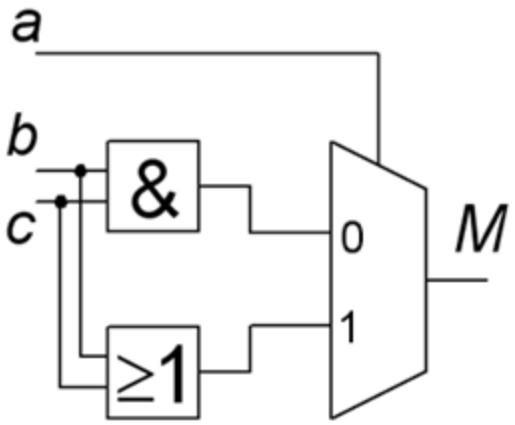
Shannon dekomposition. 2-to-1 mux och grindar.

$\bar{a}bc$

$a\bar{b}c$

$ab\bar{c}$

abc



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

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

OR

William Sandqvist william@kth.se

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

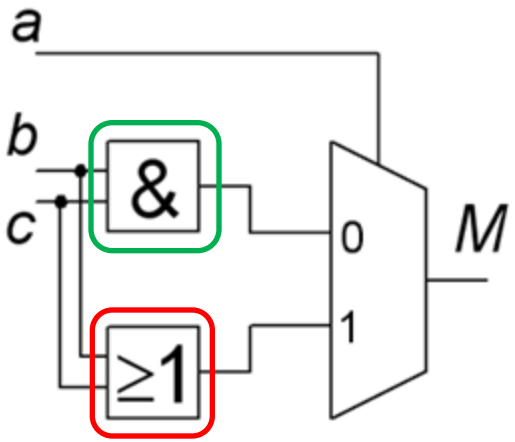
Shannon dekomposition. 2-to-1 mux och grindar.

$\bar{a}bc$

$a\bar{b}c$

$ab\bar{c}$

abc



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

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

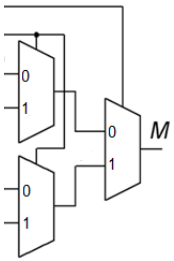
OR

William Sandqvist william@kth.se

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 dekomposition. Enbart 2-to-1 muxar.



$\bar{a}bc$

$a\bar{b}c$

$ab\bar{c}$

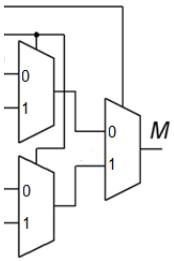
abc

William Sandqvist william@kth.se

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 dekomposition. Enbart 2-to-1 muxar.



$\overline{a}bc$

$a\overline{b}c$

$ab\overline{c}$

abc

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

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

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

William Sandqvist william@kth.se

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

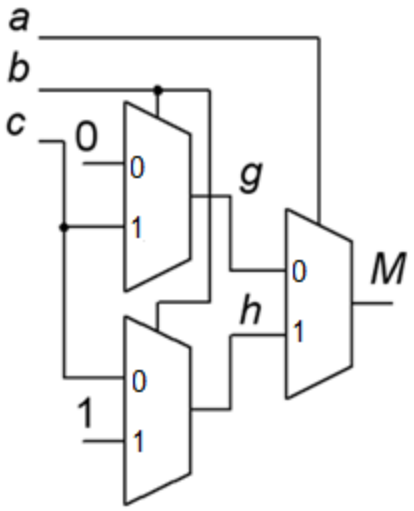
Shannon dekomposition. Enbart 2-to-1 muxar.

$\bar{a}bc$

$a\bar{b}c$

$ab\bar{c}$

abc



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

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

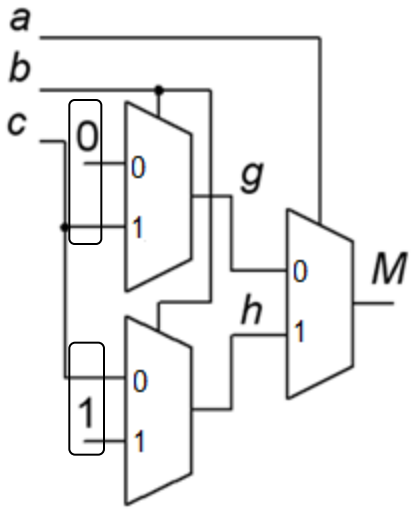
$$h = b + c \quad = b + (b + \bar{b})c = \bar{b}\,c + b + bc = \bar{b}c + b(1+c) = \bar{b} \cdot c + b \cdot 1$$

William Sandqvist william@kth.se

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 dekomposition. Enbart 2-to-1 muxar.



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

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

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

William Sandqvist william@kth.se

William Sandqvist william@kth.se

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.

$$f(w_1, w_2, w_3) = \sum m(000, 010, 011, 110) =$$
$$= \overline{w}_1 \overline{w}_2 \overline{w}_3 + \overline{w}_1 w_2 \overline{w}_3 + \overline{w}_1 w_2 w_3 + w_1 w_2 \overline{w}_3 =$$
$$= \overline{w}_1 (\overline{w}_2 \overline{w}_3 + w_2 \overline{w}_3 + w_2 w_3) + w_1 (w_2 \overline{w}_3) =$$
$$= \overline{w}_1 (w_2 + \overline{w}_3) + w_1 (w_2 \overline{w}_3)$$

William Sandqvist william@kth.se

BV 6.5

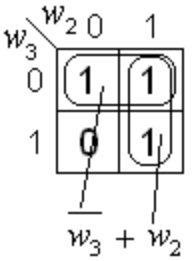
For the function

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

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

$$f(w_1, w_2, w_3) = \sum m(000, 010, 011, 110) =$$
$$= \overline{w}_1 \overline{w}_2 \overline{w}_3 + \overline{w}_1 w_2 \overline{w}_3 + \overline{w}_1 w_2 w_3 + w_1 w_2 \overline{w}_3 =$$
$$= \overline{w}_1 (\overline{w}_2 \overline{w}_3 + w_2 \overline{w}_3 + w_2 w_3) + w_1 (w_2 \overline{w}_3) =$$
$$= \overline{w}_1 (w_2 + \overline{w}_3) + w_1 (w_2 \overline{w}_3)$$

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



William Sandqvist william@kth.se

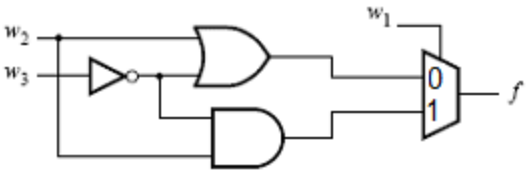
BV 6.5

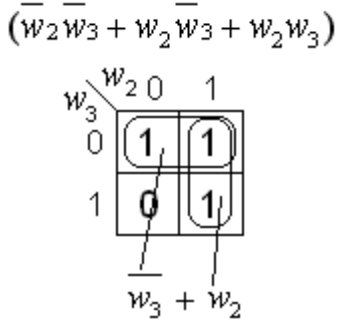
For the function

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

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

$$f(w_1, w_2, w_3) = \sum m(000, 010, 011, 110) =$$
$$= \overline{w}_1 \overline{w}_2 \overline{w}_3 + \overline{w}_1 w_2 \overline{w}_3 + \overline{w}_1 w_2 w_3 + w_1 w_2 \overline{w}_3 =$$
$$= \overline{w}_1 (\overline{w}_2 \overline{w}_3 + w_2 \overline{w}_3 + w_2 w_3) + w_1 (w_2 \overline{w}_3) =$$
$$= \overline{w}_1 (w_2 + \overline{w}_3) + w_1 (w_2 \overline{w}_3)$$

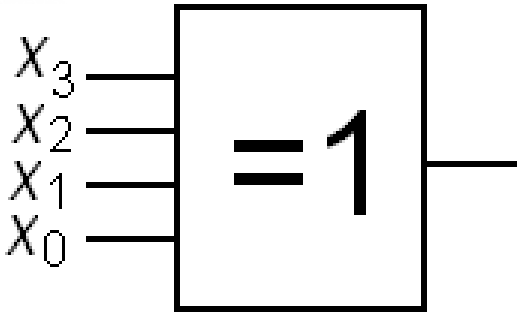




William Sandqvist william@kth.se

William Sandqvist william@kth.se

(ÖH 8.9)



Visa hur en 4 ingångars exorgrind (XOR, udda paritetsfunktion) realiseras i en FPGA-krets. Visa innehållet i SRAM-cellerna (LUT, LookUp Table).

XOR

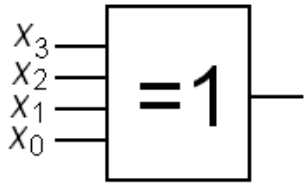
x_3x_2 \ x_1x_0	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0

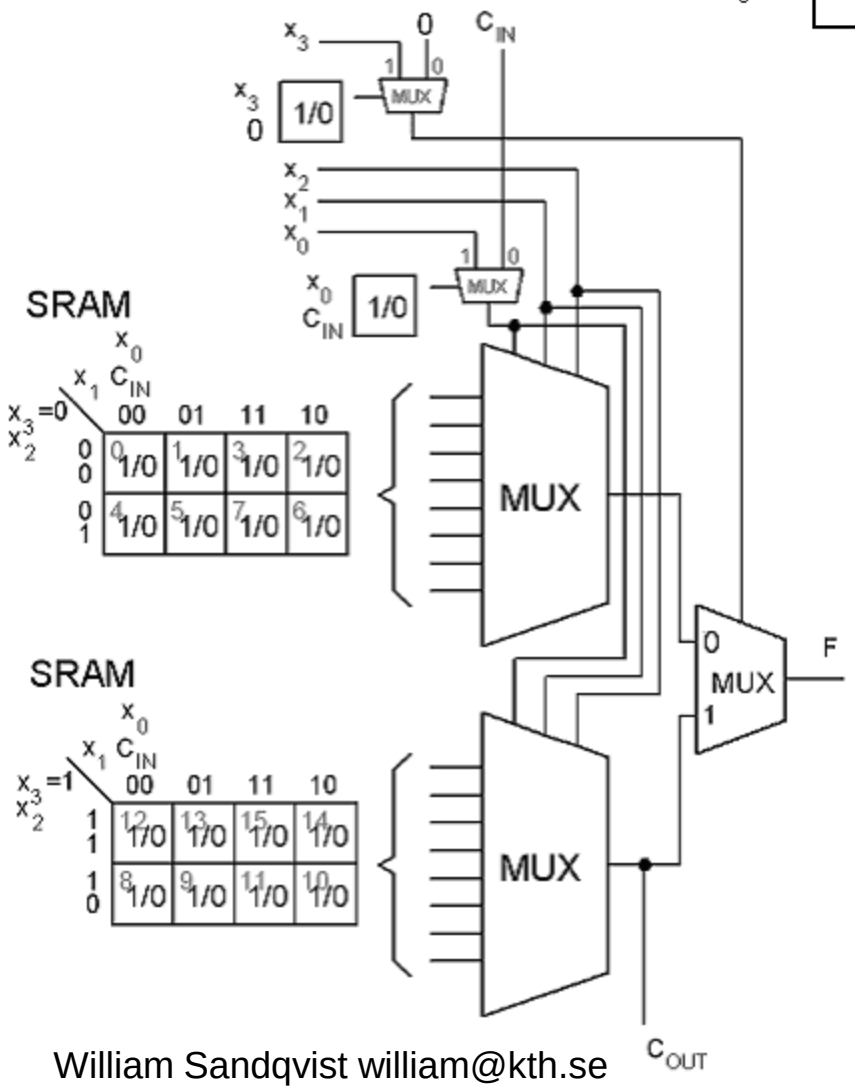
William Sandqvist william@kth.se

(8.9)

XOR

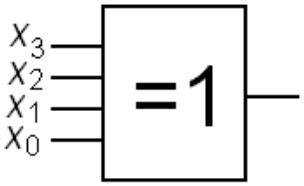
x_3x_2 \ x_1x_0	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0





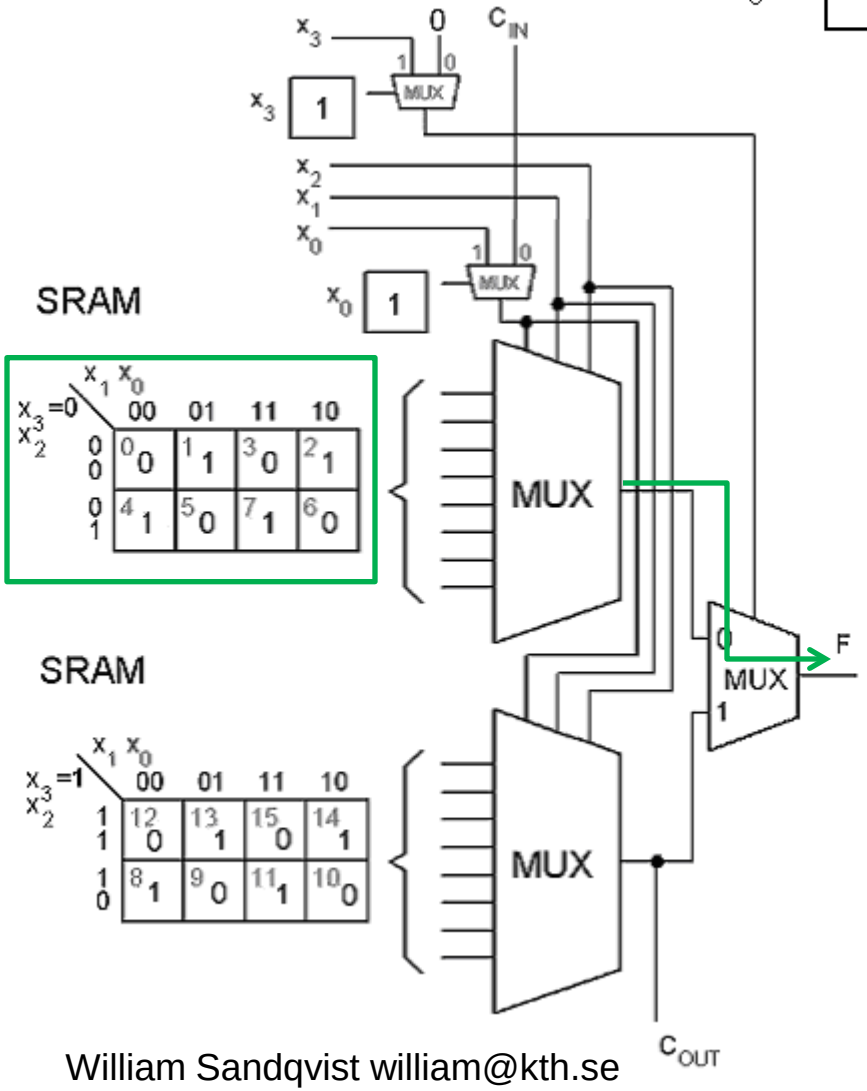
William Sandqvist william@kth.se

(8.9)



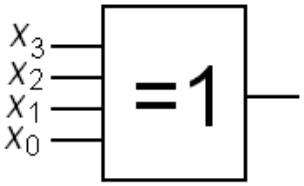
XOR

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



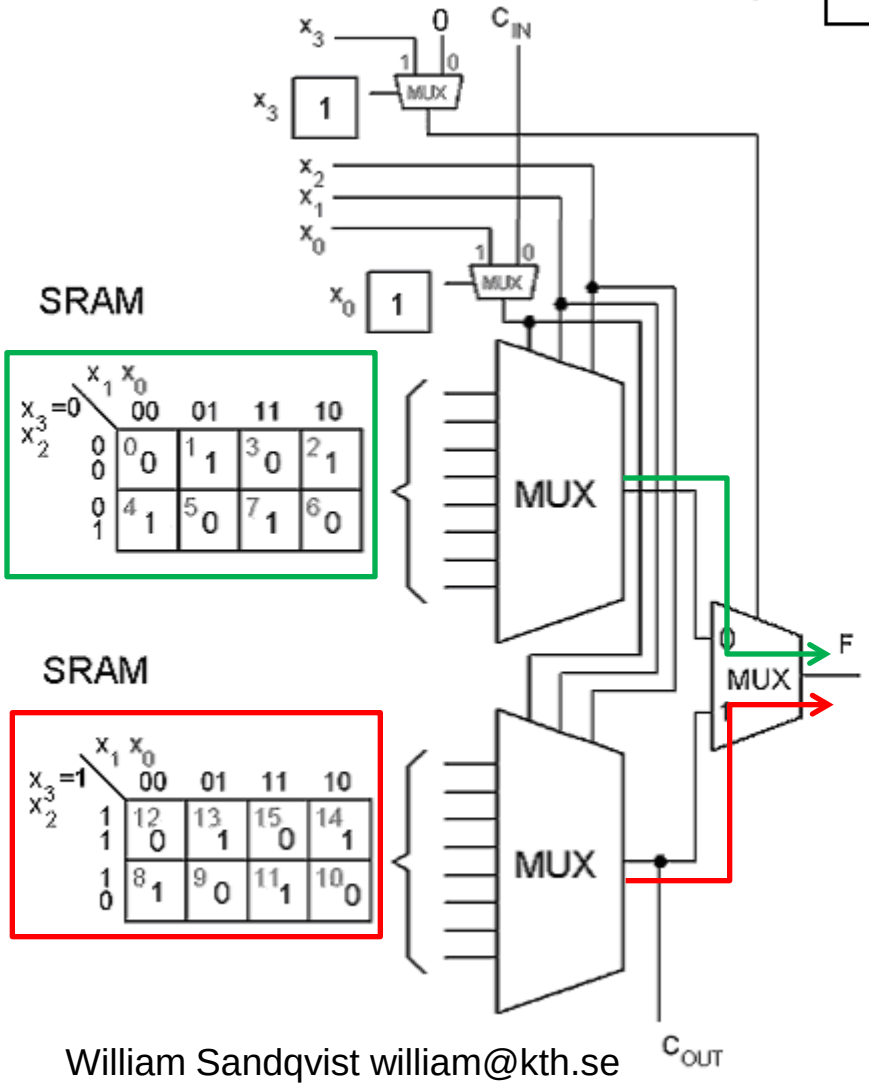
William Sandqvist william@kth.se

(8.9)



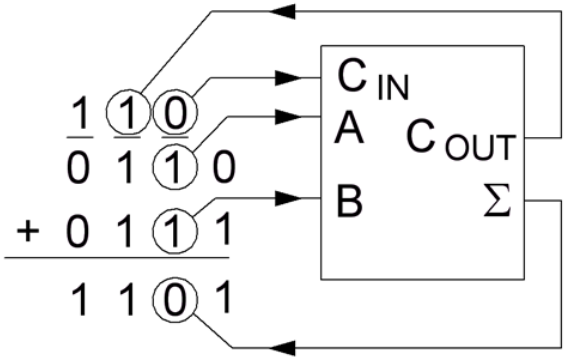
XOR

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



William Sandqvist william@kth.se

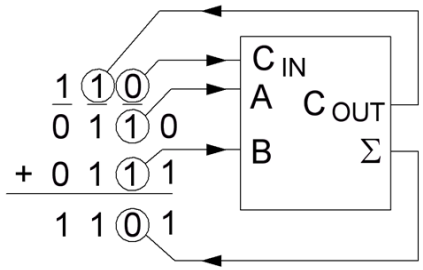
(ÖH 8.8)

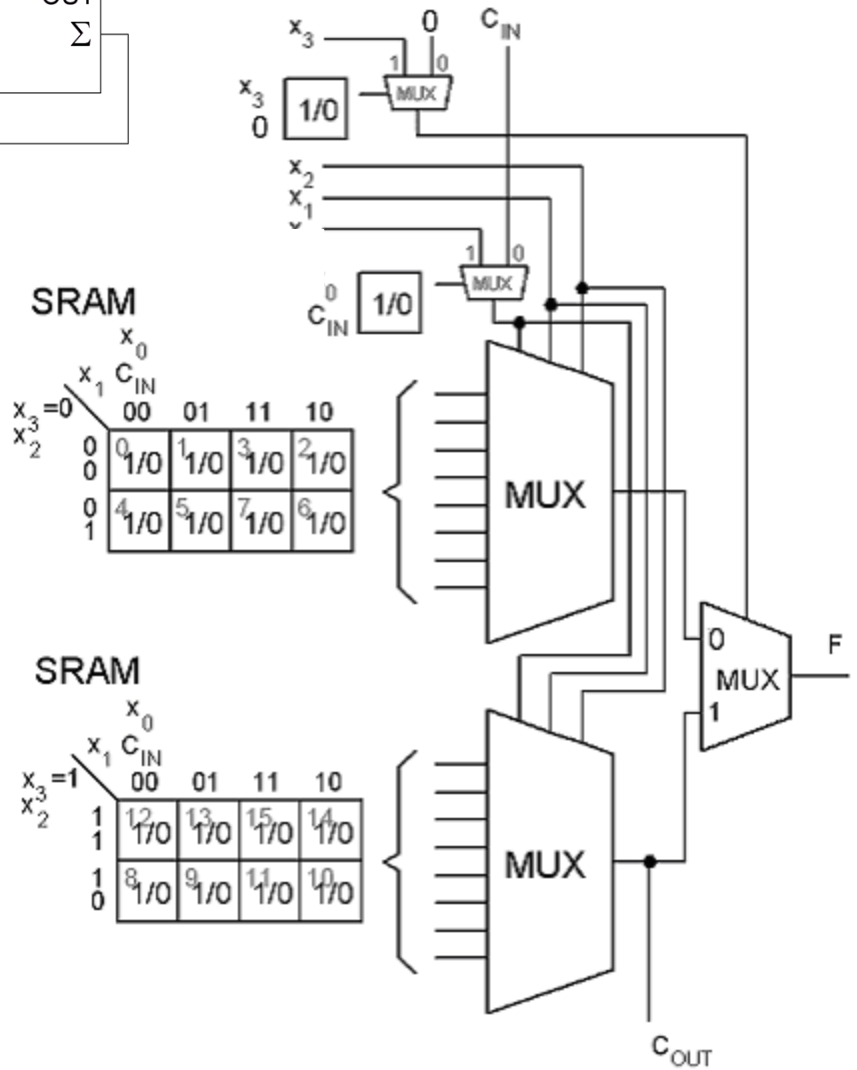


Ställ upp Ställ upp heladderarens sanningstabell. Visa hur en heladderare realiseras i en FPGA-krets. Logikelementen i en FPGA har möjlighet att kaskadkoppla C_{OUT} och C_{IN} mellan ”grannarna”. Visa innehållet i SRAM-cellerna (LUT, LookUp Table).

William Sandqvist william@kth.se

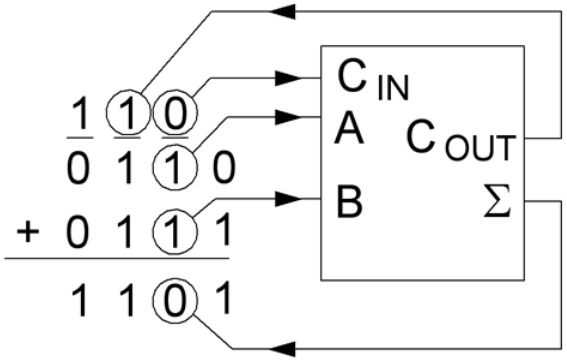
(8.8)



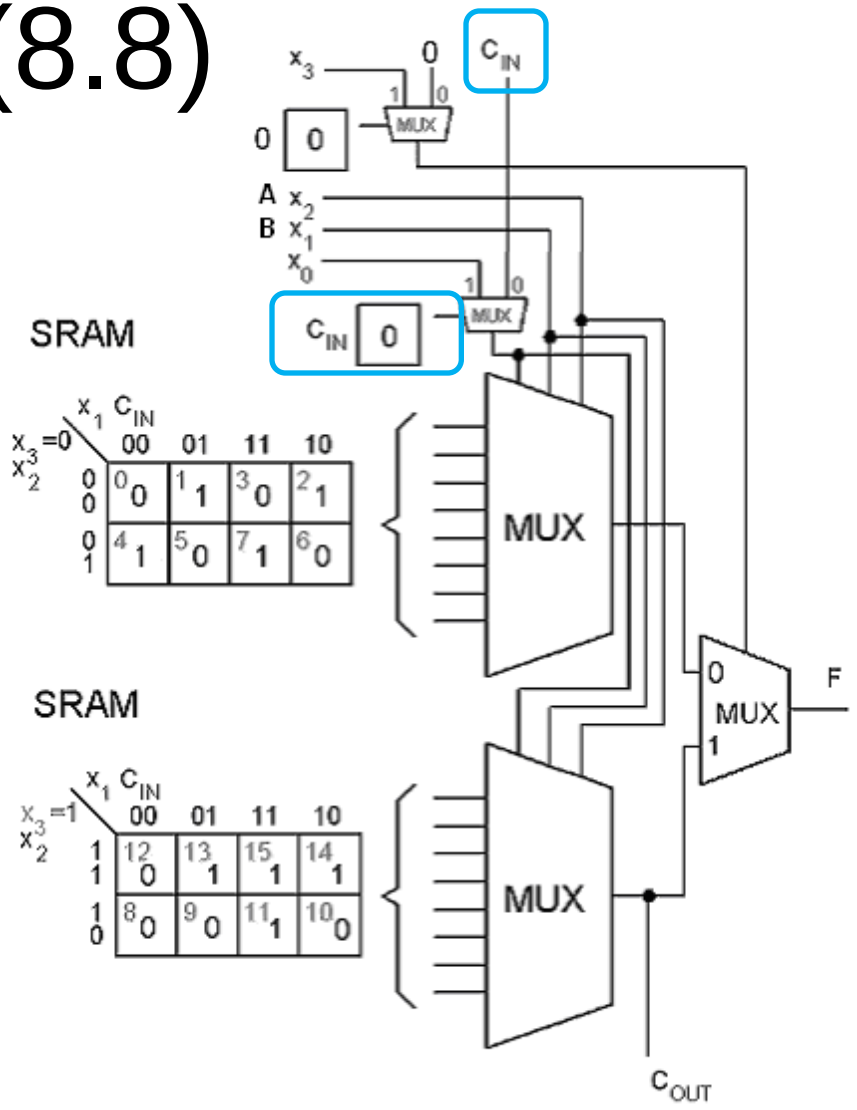


William Sandqvist william@kth.se

(8.8)

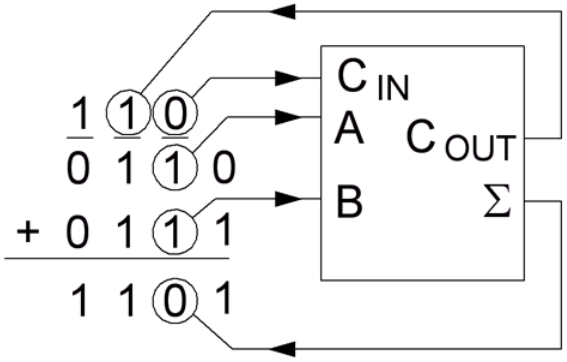


x_2	x_1	x_0		
A	B	C_{IN}	Σ	C_{OUT}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

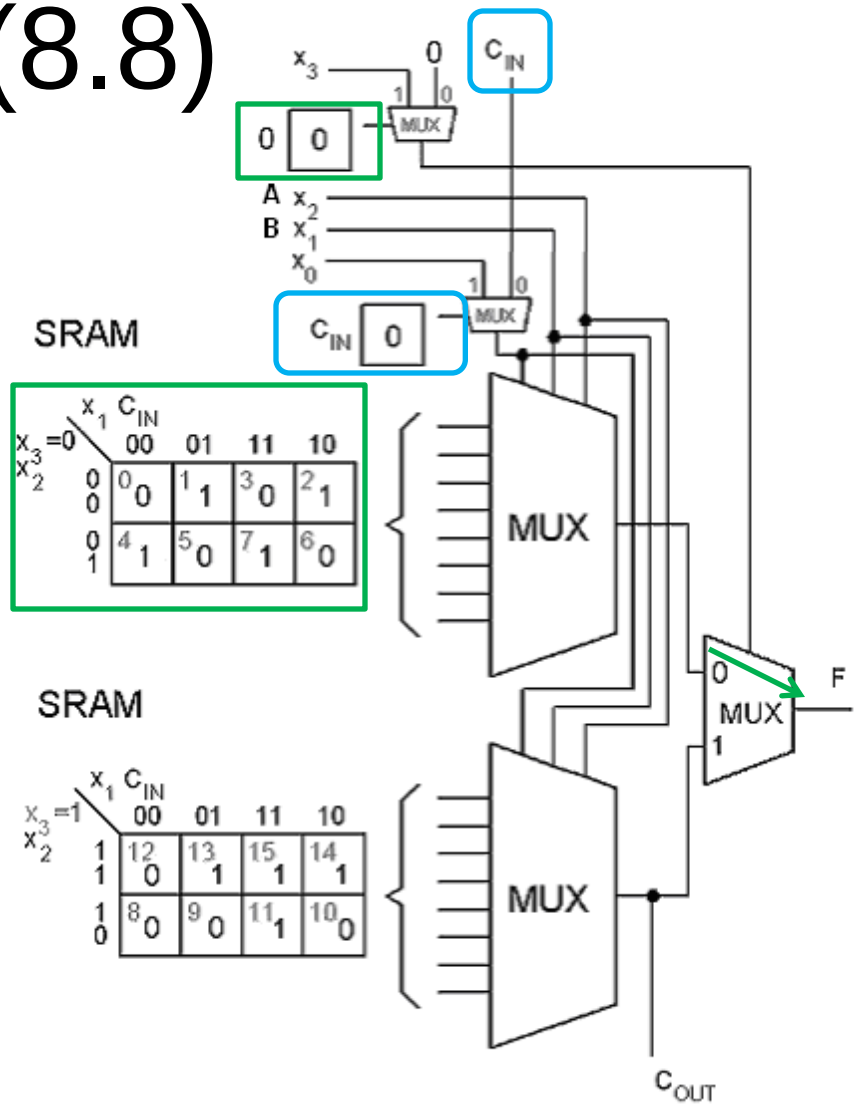


William Sandqvist william@kth.se

(8.8)

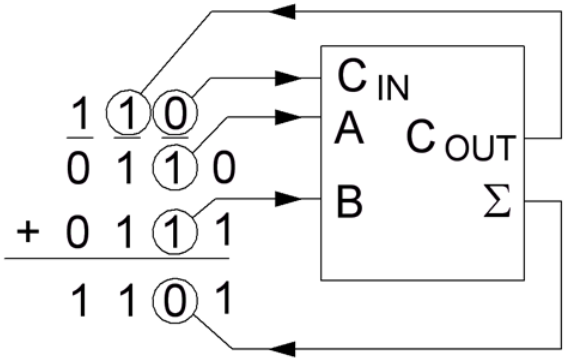


x_2	x_1	x_0		
A	B	C_{IN}	Σ	C_{OUT}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

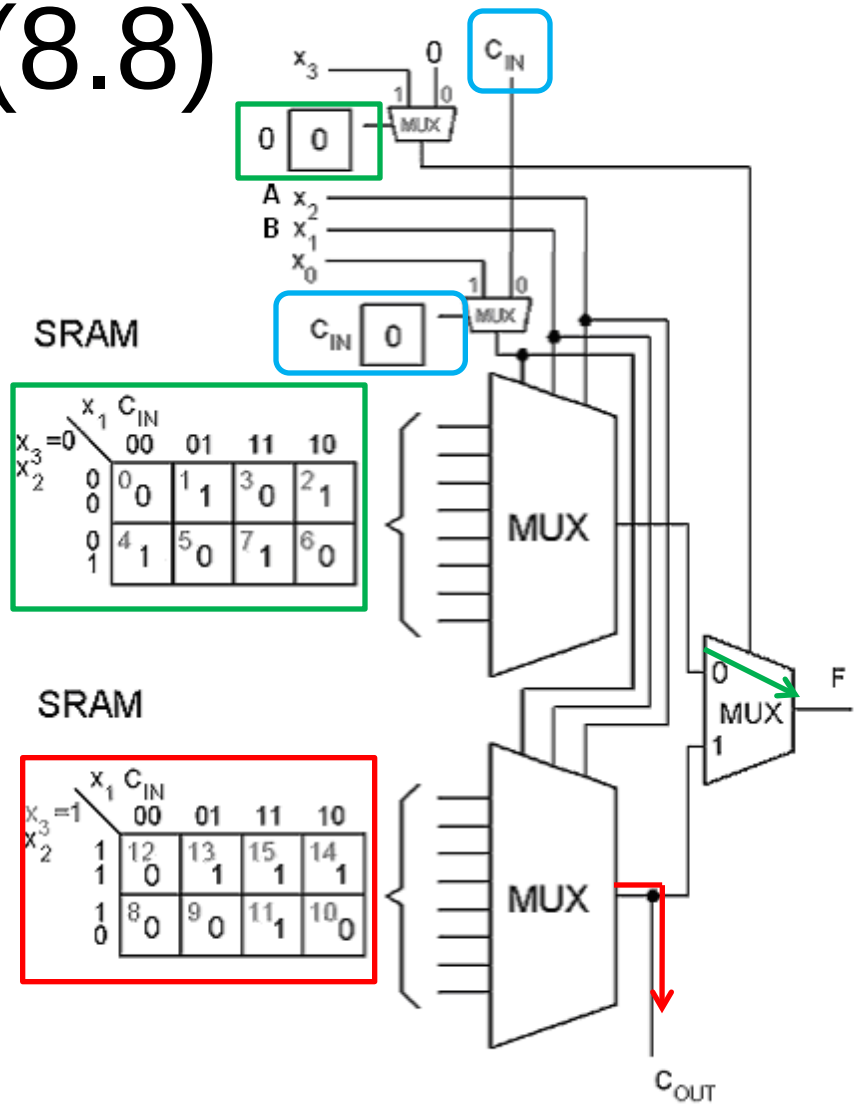


William Sandqvist william@kth.se

(8.8)



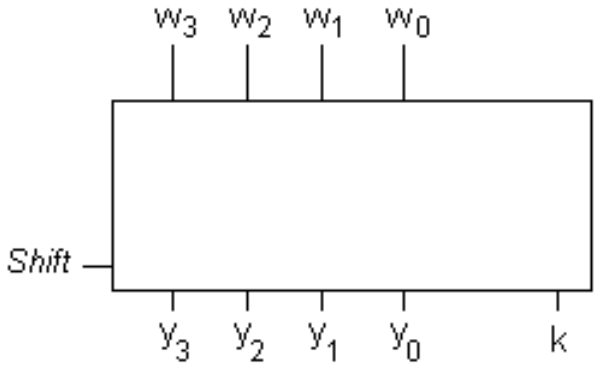
x_2	x_1	x_0		
A	B	C_{IN}	Σ	C_{OUT}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



William Sandqvist william@kth.se

(BV ex 6.31)

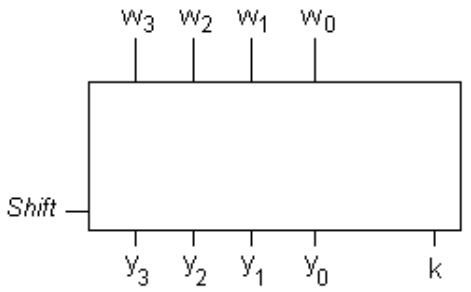
In digital systems it is often necessary to have circuits that can shift the bits of a vector one or more bit positions to the left or right.
Design a circuit that can shift a four-bit vector $W = w_3w_2w_1w_0$ one bit position to the right when a control signal *Shift* is equal to 1.
Let the outputs 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 $k = 0$.



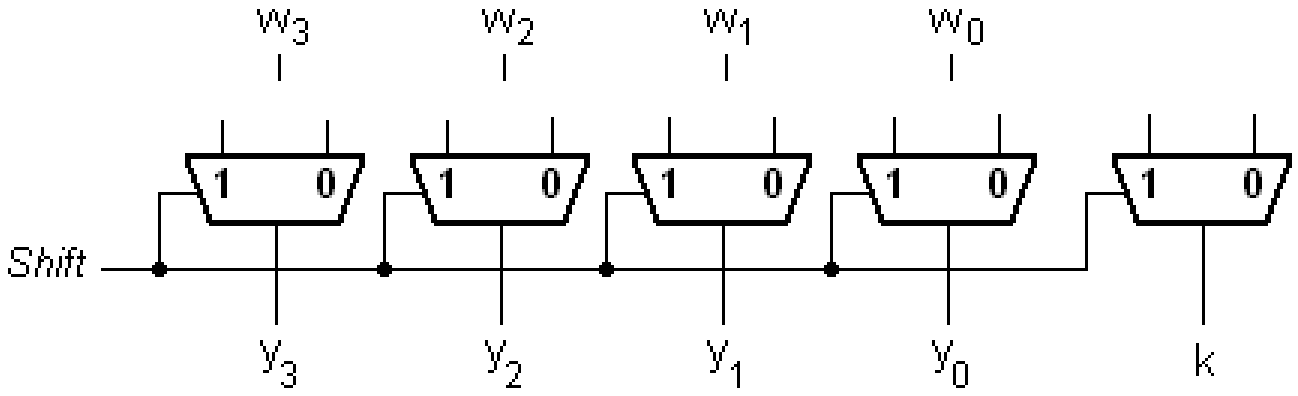
William Sandqvist william@kth.se

William Sandqvist william@kth.se

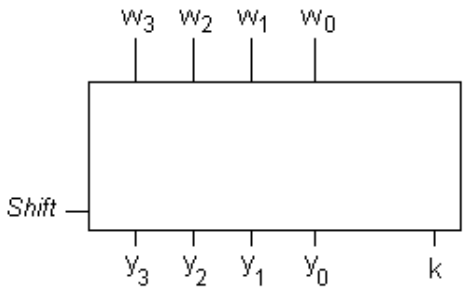
(BV ex 6.31)



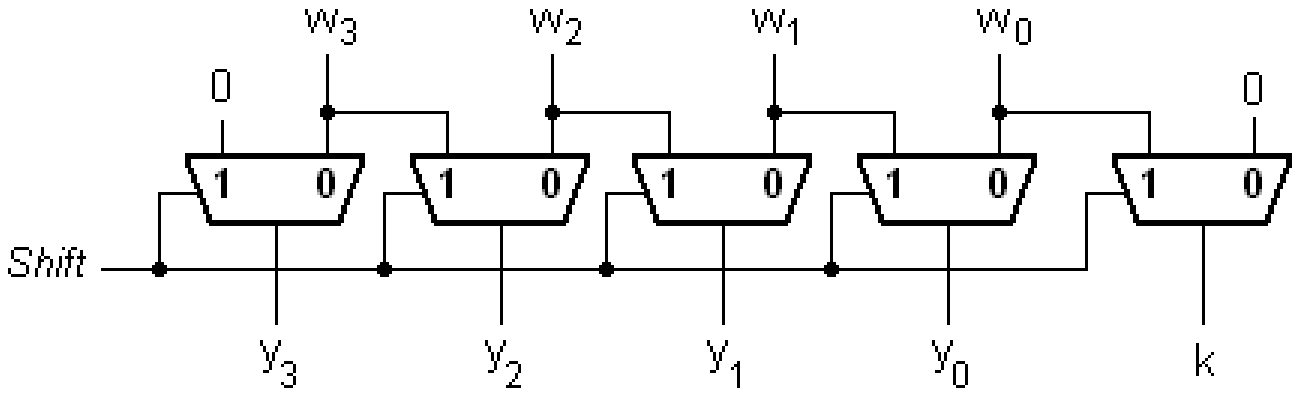
Vi använder multiplexorer:



(BV ex 6.31)



Vi använder multiplexorer:



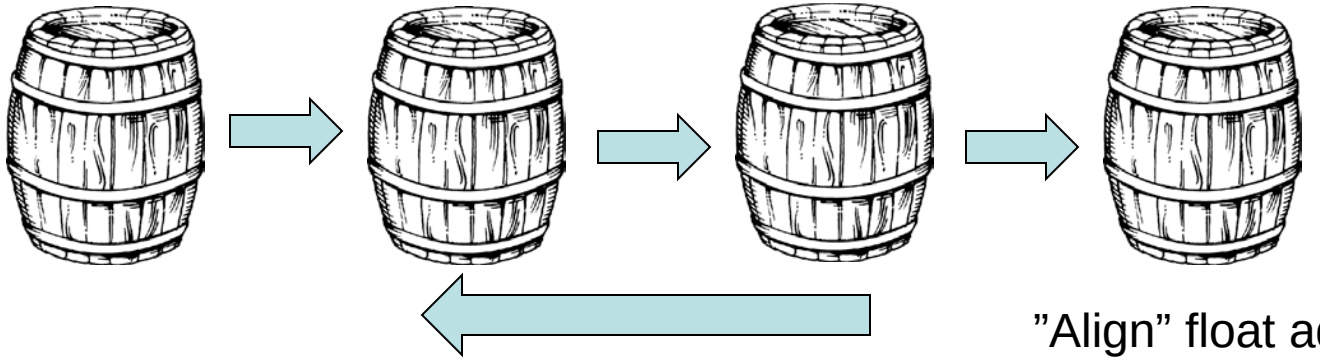
William Sandqvist william@kth.se

William Sandqvist william@kth.se

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 vacated bit on the left side with 0. If the bits that are shifted out are placed into the vacated 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.



”Align” float add/sub

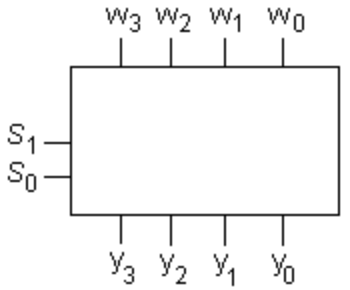
En barrelshifter används för att snabba upp flyttalsoperationer.

William Sandqvist william@kth.se

Barrel shifter

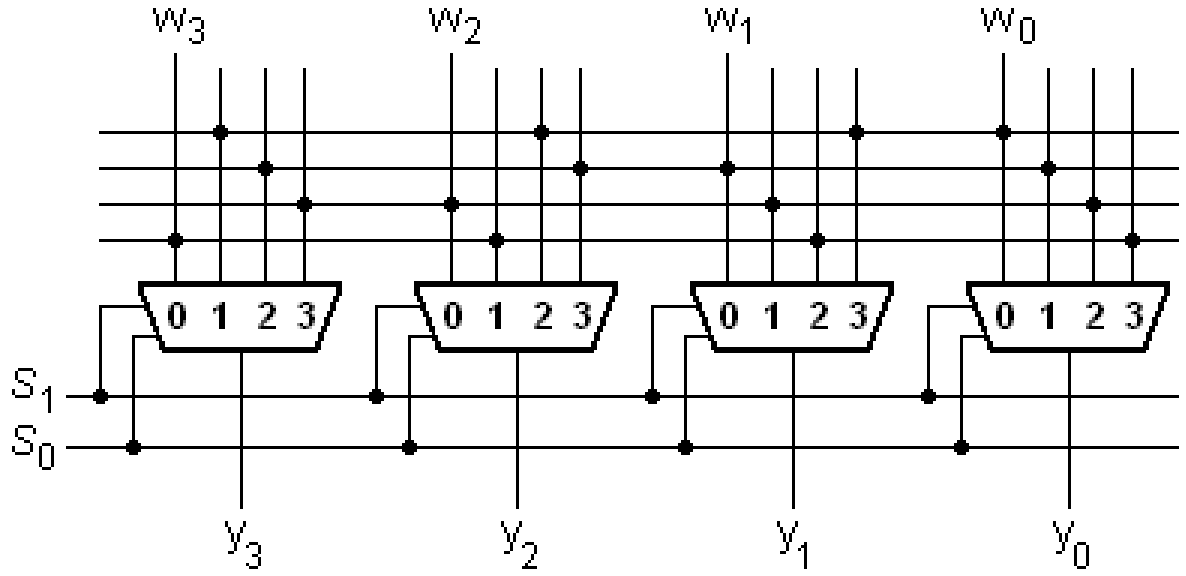
William Sandqvist william@kth.se

BV ex. 6.32



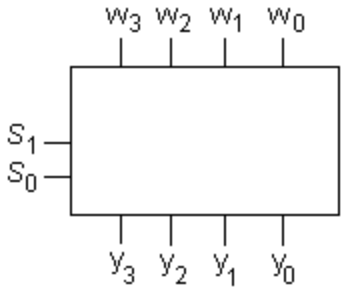
Sanningstabell:

S_1	S_0	y_3	y_2	y_1	y_0
0	0	w_3	w_2	w_1	w_0
0	1	w_0	w_3	w_2	w_1
1	0	w_1	w_0	w_3	w_2
1	1	w_2	w_1	w_0	w_3



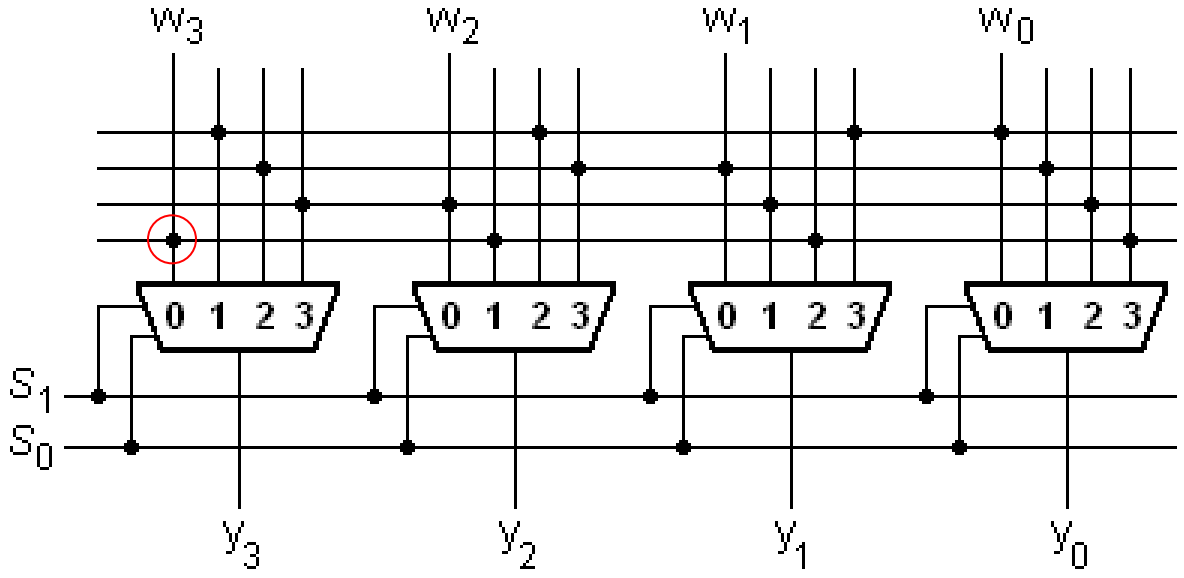
William Sandqvist william@kth.se

BV ex. 6.32



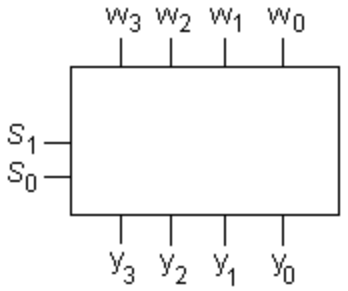
Sanningstabell:

S_1	S_0	y_3	y_2	y_1	y_0
0	0	w_3	w_2	w_1	w_0
0	1	w_0	w_3	w_2	w_1
1	0	w_1	w_0	w_3	w_2
1	1	w_2	w_1	w_0	w_3



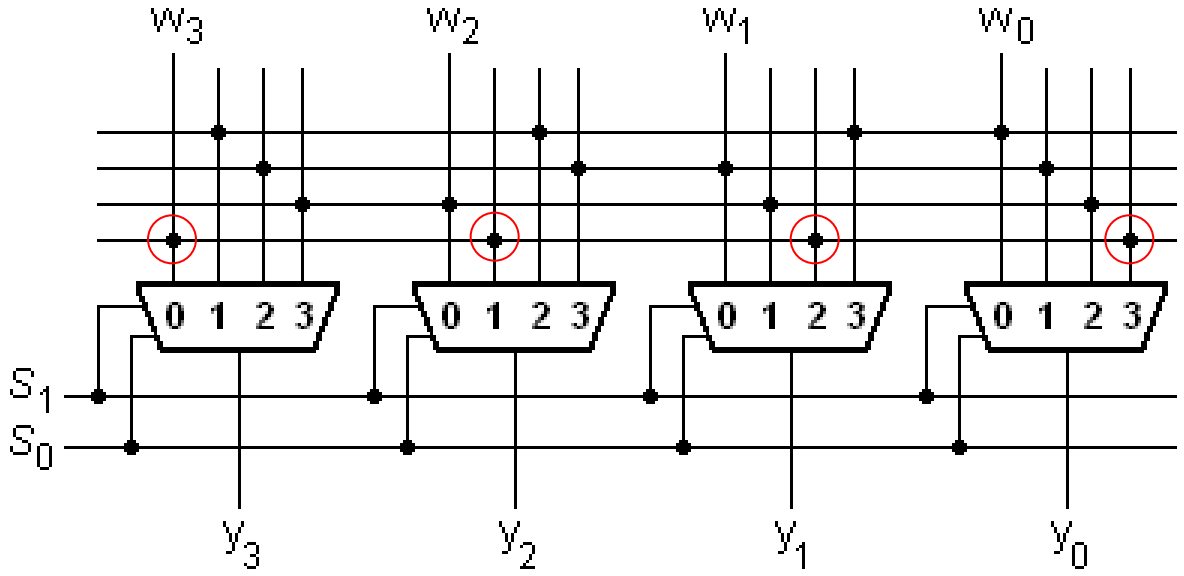
William Sandqvist william@kth.se

BV ex. 6.32



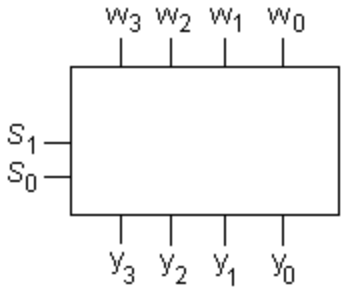
Sanningstabell:

S_1	S_0	y_3	y_2	y_1	y_0
0	0	w_3	w_2	w_1	w_0
0	1	w_0	w_3	w_2	w_1
1	0	w_1	w_0	w_3	w_2
1	1	w_2	w_1	w_0	w_3



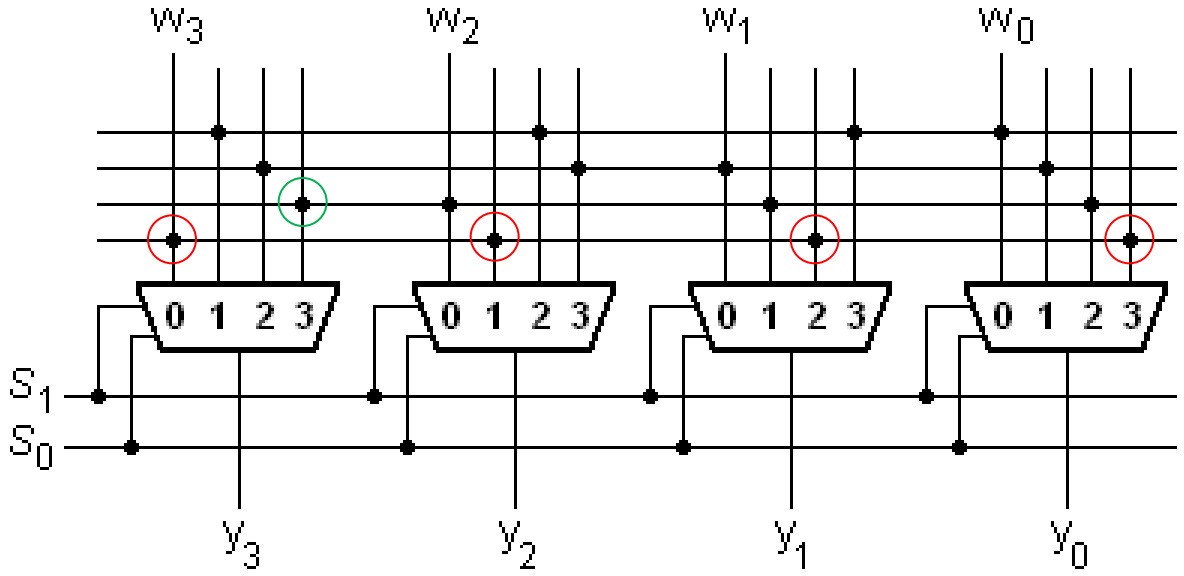
William Sandqvist william@kth.se

BV ex. 6.32



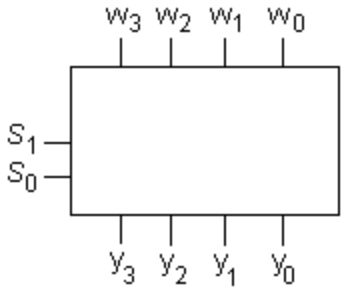
Sanningstabell:

S_1	S_0	y_3	y_2	y_1	y_0
0	0	w_3	w_2	w_1	w_0
0	1	w_0	w_3	w_2	w_1
1	0	w_1	w_0	w_3	w_2
1	1	w_2	w_1	w_0	w_3



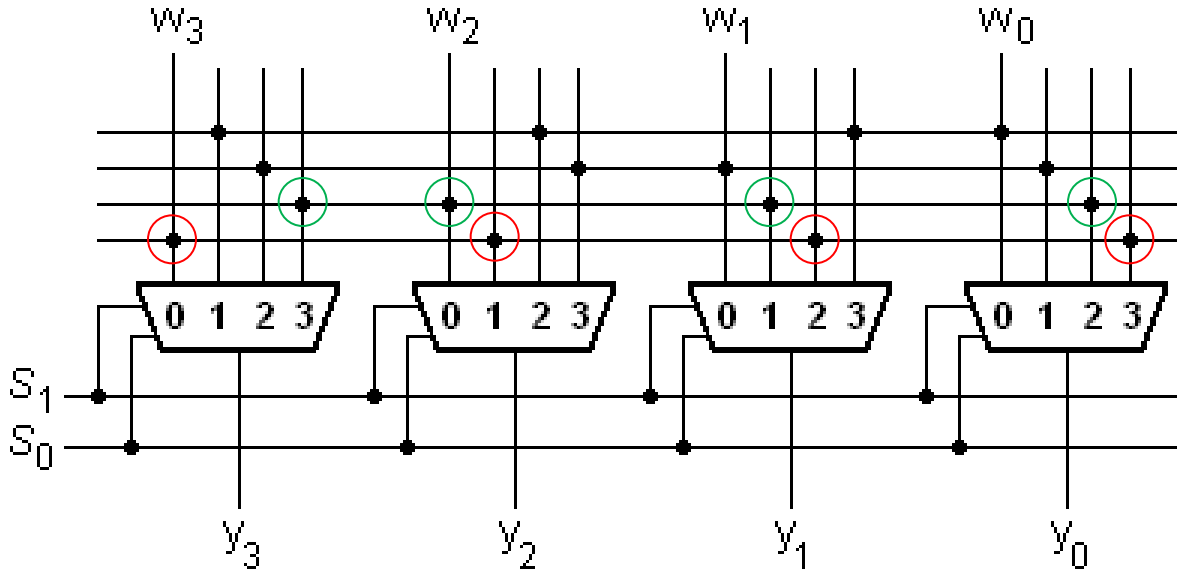
William Sandqvist william@kth.se

BV ex. 6.32



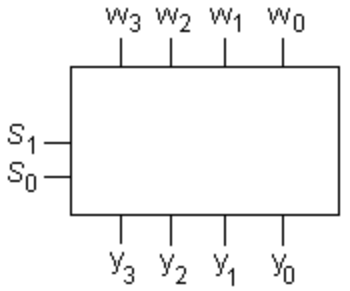
Sanningstabell:

S_1 S_0	y_3	y_2	y_1	y_0
0 0	w_3	w_2	w_1	w_0
0 1	w_0	w_3	w_2	w_1
1 0	w_1	w_0	w_3	w_2
1 1	w_2	w_1	w_0	w_3



William Sandqvist william@kth.se

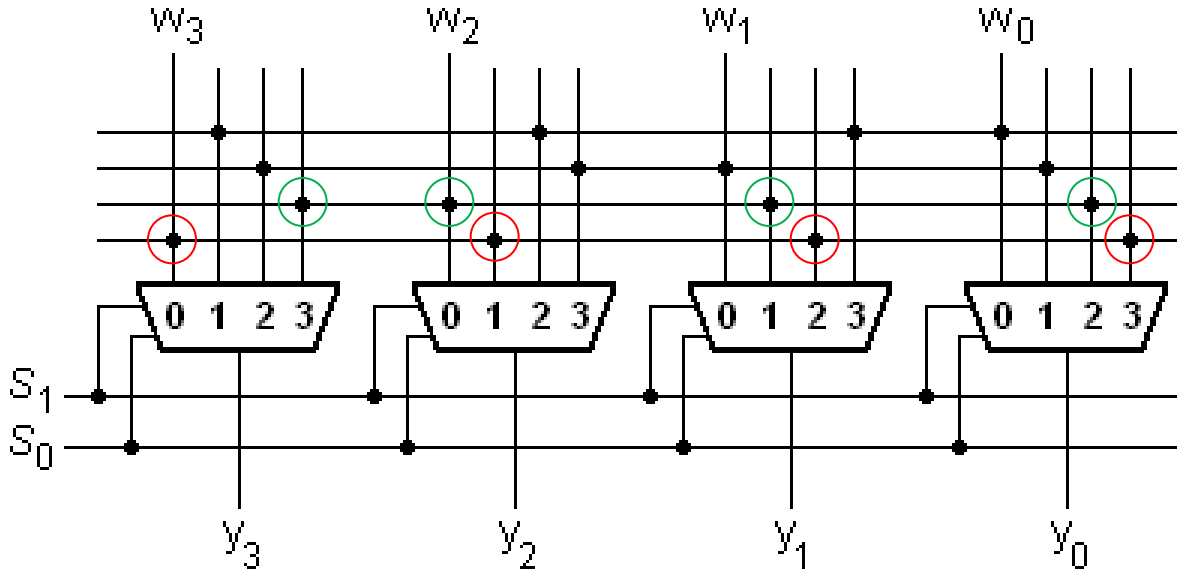
BV ex. 6.32



Sanningstabell:

S_1 S_0	y_3	y_2	y_1	y_0
0 0	w_3	w_2	w_1	w_0
0 1	w_0	w_3	w_2	w_1
1 0	w_1	w_0	w_3	w_2
1 1	w_2	w_1	w_0	w_3

Och så vidare …



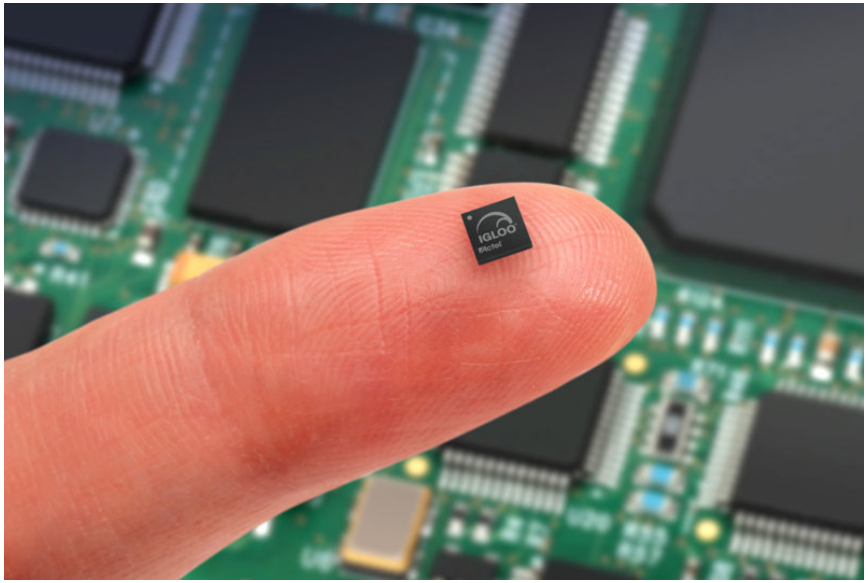
William Sandqvist william@kth.se

William Sandqvist william@kth.se

= Lågpris FPGA

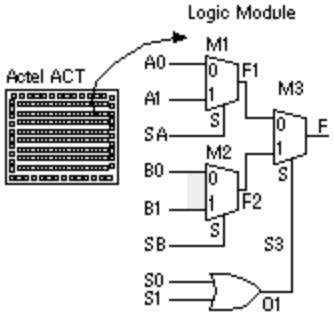
Key Benefits

- Lowest FPGA unit cost starting 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



William Sandqvist william@kth.se

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.

William Sandqvist william@kth.se

BV 6.16

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

William Sandqvist william@kth.se

BV 6.16

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

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

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

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

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

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

William Sandqvist william@kth.se

BV 6.16

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

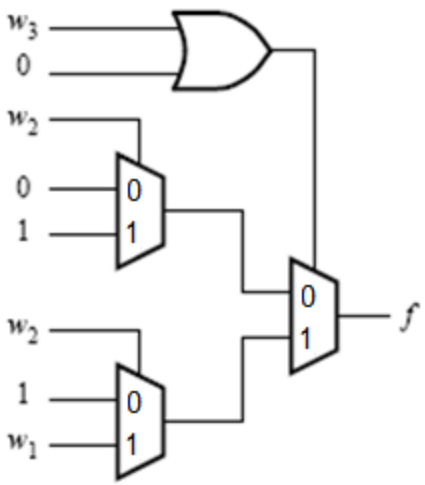
$$f = \overline{w_3}\,(w_2) + w_3\,(w_1 + \overline{w_2})$$

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

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

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

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



William Sandqvist william@kth.se

BV 6.16

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

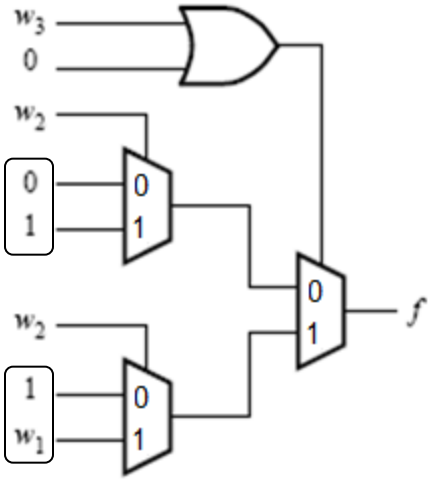
$$f = \overline{w_3}\,(w_2) + w_3\,(w_1 + \overline{w_2})$$

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

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

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

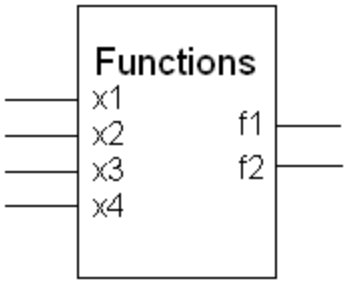
$$f = \overline{w_3}\,(\overline{w_2}\cdot \boxed{0} + w_2\cdot \boxed{1}) + w_3\left(\overline{w_2}\cdot \boxed{1} + w_2\cdot \boxed{w_1}\right)$$



William Sandqvist william@kth.se

William Sandqvist william@kth.se

VHDL BV 2.51a



Write VHDL code to describe the following functions

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

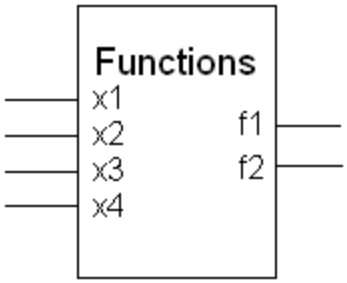
$$f_2 = (x_1 + \overline{x_3}) \cdot (x_1 + x_2 + \overline{x_4}) \cdot (x_2 + \overline{x_3} + \overline{x_4})$$

VHDL koden skrivs med en texteditor och sparas i en fil med ändelsen `.vhd` . Koden består alltid av två avsnitt ENTITY och ARCHITECTURE.

Entity är en beskrivning av hur kretsen ”ser ut utifrån” (gränssnittet), och Architecture hur den ”ser ut inuti.”

William Sandqvist william@kth.se

VHDL BV 2.51a



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

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

Programkod skrivs med texteditorer. Man kan därför bara göra text-kommentarer till koden. Ett typsnitt med fast bredd används (ex. `Courier New`).

Kommentarer börjar med `--`

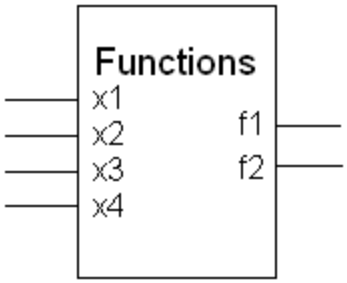
Om man vill, kan man ”rita” förtydligande ASCII-grafik inom kommentarraderna.

Man brukar också indentera textblock som hör ihop för ökad tydlighet.

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

William Sandqvist william@kth.se

VHDL BV 2.51a



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

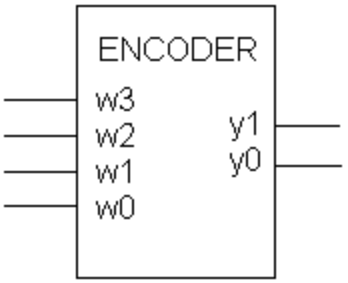
$$f_2 = (x_1 + \overline{x_3})\cdot(x_1 + x_2 + \overline{x_4})\cdot(x_2 + \overline{x_3} + \overline{x_4})$$

```
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 ;
```

William Sandqvist william@kth.se

VHDL BV 6.21

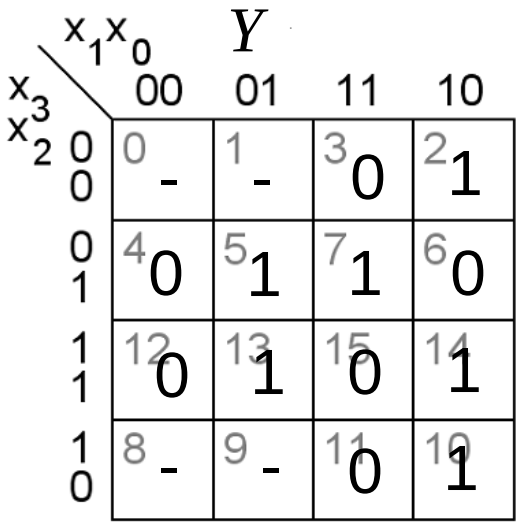


Using a **selected** signal assignement, write VHDL code for a 4-to-2 binary encoder. Only one of w0 …w3 is ”1” at a time.

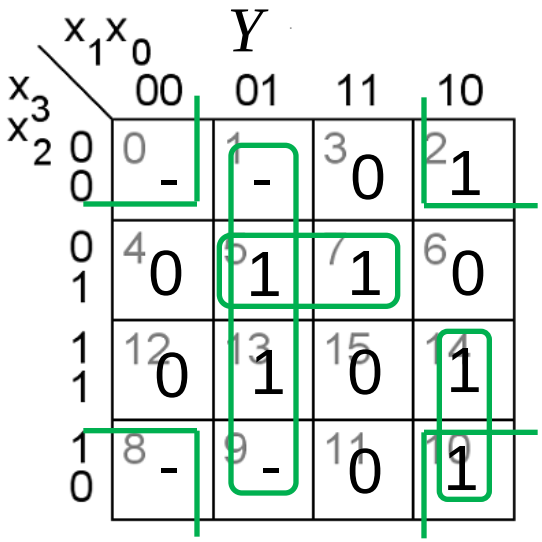
```
LIBRARY ieee;
USE IEEE.std_logic_1164.all;
ENTITY ENCODER IS
   PORT( w :IN  STD_LOGIC_VECTOR( 3 DOWNTO 0 )  ;
         y :OUT STD_LOGIC_VECTOR( 1 DOWNTO 0 ) );
END ENCODER
ARCHITECTURE Behavior OF ENCODER IS
BEGIN
   WITH w SELECT
      y <= ”00” WHEN ”0001”,
           ”01” WHEN ”0010”,
      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

William Sandqvist william@kth.se

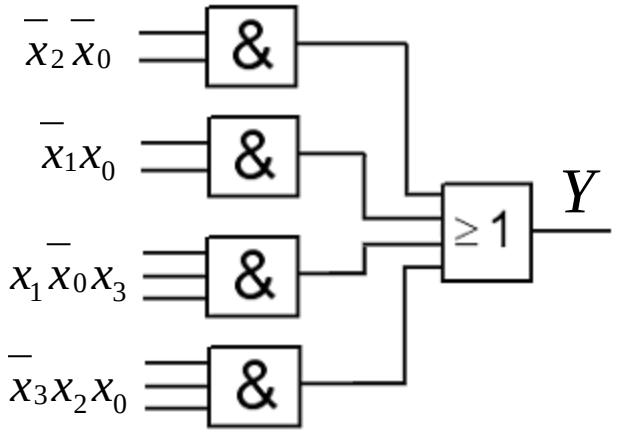
(8.10) Extrauppgift



William Sandqvist william@kth.se



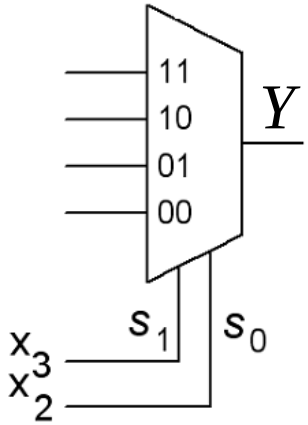
$$Y = \bar{x}_2 \bar{x}_0 + \bar{x}_1 x_0 + x_1 \bar{x}_0 x_3 + \bar{x}_3 x_2 x_0$$

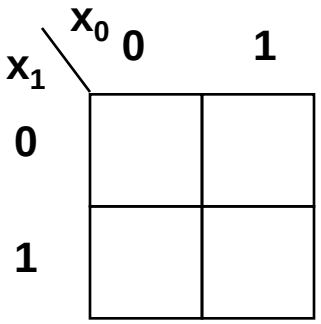


William Sandqvist william@kth.se

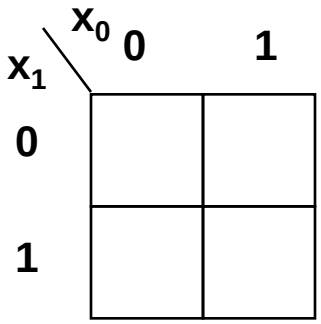
x_3x_2 \ x_1x_0	00	01	11	10
00	0 -	1 -	3 0	2 1
01	4 0	5 1	7 1	6 0
11	12 0	13 1	15 0	14 1
10	8 -	9 -	11 0	10 1

Y

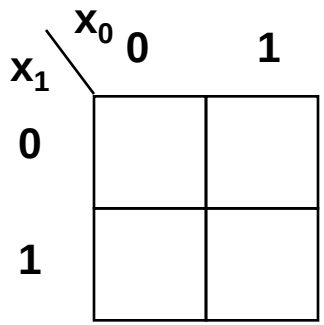




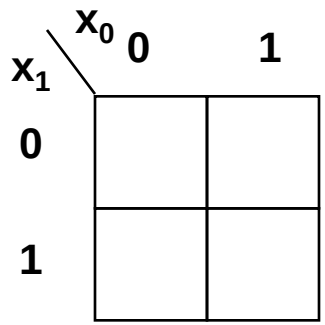
$x_3x_2(0,0) \Rightarrow$



$x_3x_2(0,1) \Rightarrow$



$x_3x_2(1,1) \Rightarrow$

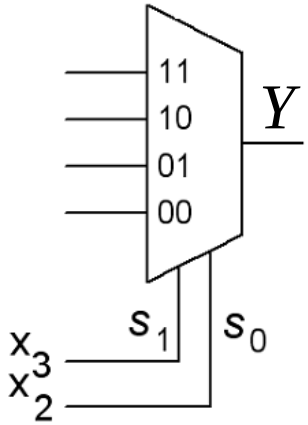


$x_3x_2(1,0) \Rightarrow$

William Sandqvist william@kth.se

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



x_1 \ x_0	0	1
0	-	-
1	1	0

$x_3x_2(0,0) \Rightarrow$

x_1 \ x_0	0	1
0	0	1
1	0	1

$x_3x_2(0,1) \Rightarrow$

x_1 \ x_0	0	1
0	0	1
1	1	0

$x_3x_2(1,1) \Rightarrow$

x_1 \ x_0	0	1
0	-	-
1	1	0

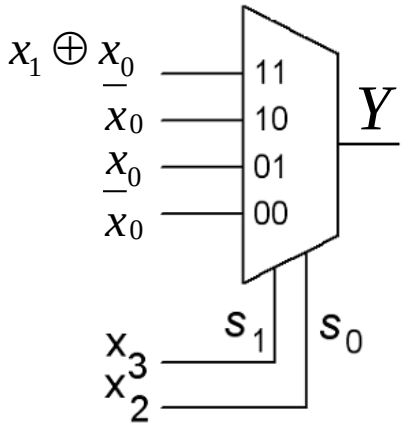
$x_3x_2(1,0) \Rightarrow$

William Sandqvist william@kth.se

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

$$Y = \bar{x}_2\bar{x}_0 + \bar{x}_1x_0 + x_1\bar{x}_0x_3 + \bar{x}_3x_2x_0$$



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

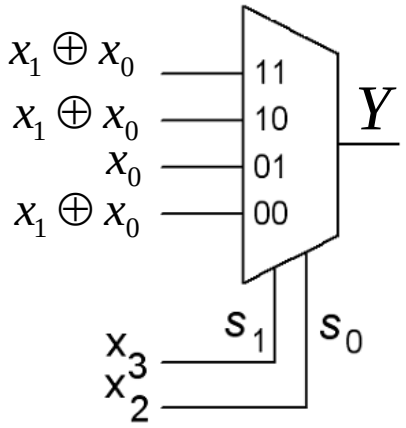
$x_3x_2(1,0) \Rightarrow$

$Y = \bar{x}_0$

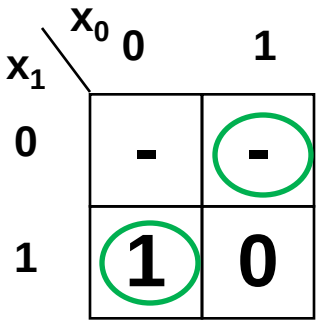
William Sandqvist william@kth.se

x_3x_2 \ 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

$$Y = \bar{x}_2\bar{x}_0 + \bar{x}_1x_0 + x_1\bar{x}_0x_3 + \bar{x}_3x_2x_0$$

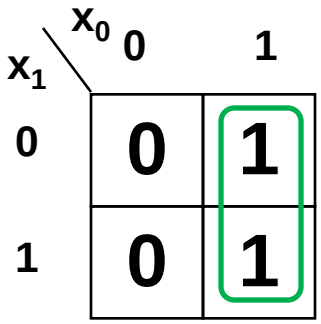


Eller …



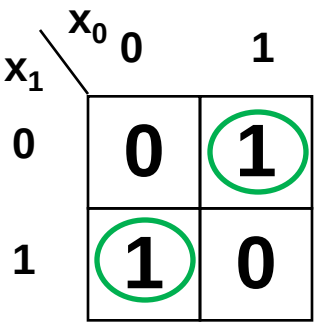
$x_3x_2(0,0) \Rightarrow$

$Y = x_1 \oplus x_0$



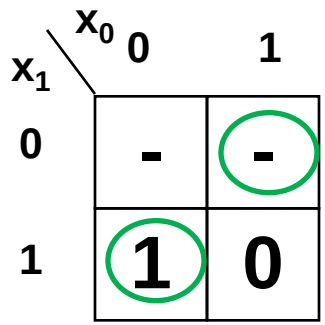
$x_3x_2(0,1) \Rightarrow$

$Y = x_0$



$x_3x_2(1,1) \Rightarrow$

$Y = x_1 \oplus x_0$



$x_3x_2(1,0) \Rightarrow$

$Y = x_1 \oplus x_0$

Eller om det är svårt att få tag i x_0 inverterad …

William Sandqvist william@kth.se

William Sandqvist william@kth.se