

Ex. 8.4 7-4-2-1 code

Codeconverter 7-4-2-1-code to BCD-code.

When encoding the digits 0 ... 9 sometimes in the past a code having weights 7-4-2-1 instead of the binary code weights 8-4-2-1 was used.

In the cases where a digit's code word can be expressed in various ways the code word that contains the least number of ones is selected

(A variation of the 7-4-2-1 code is used today to store the bar code)



	7	4	2	1		8	4	2	1
	x_7	x_4	x_2	x_1		y_8	y_4	y_2	y_1
(0)	0	0	0	0	0	0	0	0	0
(1)	0	0	0	1	1	0	0	0	1
(2)	0	0	1	0	2	0	0	1	0
(3)	0	0	1	1	3	0	0	1	1
(4)	0	1	0	0	4	0	1	0	0
(5)	0	1	0	1	5	0	1	0	1
(6)	0	1	1	0	6	0	1	1	0
					7	0	1	1	1
					8	1	0	0	0
					9	1	0	0	1

Ex. 8.4 7-4-2-1 code

Codeconverter 7-4-2-1-code to BCD-code.

When encoding the digits 0 ... 9 sometimes in the past a code having weights 7-4-2-1 instead of the binary code weights 8-4-2-1 was used.

In the cases where a digit's code word can be expressed in various ways the code word that contains the least number of ones is selected

(A variation of the 7-4-2-1 code is used today to store the bar code)



	7	4	2	1		8	4	2	1
	x_7	x_4	x_2	x_1		y_8	y_4	y_2	y_1
(0)	0	0	0	0	0	0	0	0	0
(1)	0	0	0	1	1	0	0	0	1
(2)	0	0	1	0	2	0	0	1	0
(3)	0	0	1	1	3	0	0	1	1
(4)	0	1	0	0	4	0	1	0	0
(5)	0	1	0	1	5	0	1	0	1
(6)	0	1	1	0	6	0	1	1	0
(8)	1	0	0	0	7	0	1	1	1
(9)	1	0	0	1	8	1	0	0	0
					9	1	0	0	1

Ex. 8.4 7-4-2-1 code

Codeconverter 7-4-2-1-code to BCD-code.

When encoding the digits 0 ... 9 sometimes in the past a code having weights 7-4-2-1 instead of the binary code weights 8-4-2-1 was used.

In the cases where a digit's code word can be expressed in various ways the code word that contains the least number of ones is selected

(A variation of the 7-4-2-1 code is used today to store the bar code)

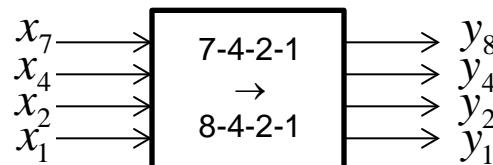


	7 4 2 1					8 4 2 1			
	x_7	x_4	x_2	x_1		y_8	y_4	y_2	y_1
(0)	0	0	0	0	0	0	0	0	0
(1)	0	0	0	1	1	0	0	0	1
(2)	0	0	1	0	2	0	0	1	0
(3)	0	0	1	1	3	0	0	1	1
(4)	0	1	0	0	4	0	1	0	0
(5)	0	1	0	1	5	0	1	0	1
(6)	0	1	1	0	6	0	1	1	0
(8)	1	0	0	0	7	0	1	1	1
(9)	1	0	0	1	8	1	0	0	0
(10)	1	0	1	0	9	1	0	0	1

	7	4	2	1		8	4	2	1
	x_7	x_4	x_2	x_1		y_8	y_4	y_2	y_1
(0)	0	0	0	0	0	0	0	0	0
(1)	0	0	0	1	1	0	0	0	1
(2)	0	0	1	0	2	0	0	1	0
(3)	0	0	1	1	3	0	0	1	1
(4)	0	1	0	0	4	0	1	0	0
(5)	0	1	0	1	5	0	1	0	1
(6)	0	1	1	0	6	0	1	1	0
(8)	1	0	0	0	7	0	1	1	1
(9)	1	0	0	1	8	1	0	0	0
(10)	1	0	1	0	9	1	0	0	1

8.4

Code converter



		y_8					
		x_2	x_1	00	01	11	10
x_7	x_4	0	0	1	3	2	
0	0	4	5	7	6		
0	1	12	13	15	14		
1	1	12	13	15	14		
1	0	8	9	11	10		

		y_4					
		x_2	x_1	00	01	11	10
x_7	x_4	0	0	0	1	3	2
0	0	0	4	5	7	6	
0	1	0	4	5	7	6	
1	1	12	13	15	14		
1	1	12	13	15	14		
1	0	8	9	11	10		

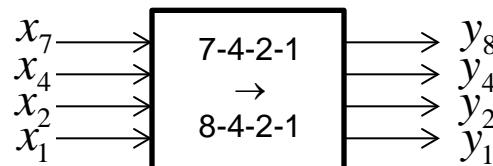
		y_2					
		x_2	x_1	00	01	11	10
x_7	x_4	0	0	0	1	3	2
0	0	0	4	5	7	6	
0	1	0	4	5	7	6	
1	1	12	13	15	14		
1	1	12	13	15	14		
1	0	8	9	11	10		

		y_1					
		x_2	x_1	00	01	11	10
x_7	x_4	0	0	0	1	3	2
0	0	0	4	5	7	6	
0	1	0	4	5	7	6	
1	1	12	13	15	14		
1	1	12	13	15	14		
1	0	8	9	11	10		

	7	4	2	1		8	4	2	1
	x_7	x_4	x_2	x_1		y_8	y_4	y_2	y_1
(0)	0	0	0	0	0	0	0	0	0
(1)	0	0	0	1	1	0	0	0	1
(2)	0	0	1	0	2	0	0	1	0
(3)	0	0	1	1	3	0	0	1	1
(4)	0	1	0	0	4	0	1	0	0
(5)	0	1	0	1	5	0	1	0	1
(6)	0	1	1	0	6	0	1	1	0
(8)	1	0	0	0	7	0	1	1	1
(9)	1	0	0	1	8	1	0	0	0
(10)	1	0	1	0	9	1	0	0	1

8.4

Code converter



		y ₈				
		x ₂ x ₁	00	01	11	10
		x ₇	0	0	3	2
	0	0	0	1	0	2
	1	0	4	5	7	6
	1	1	12	13	15	14
	0	1	8	9	11	10

		y ₄				
		x ₂ x ₁	00	01	11	10
		x ₇	0	0	3	2
	0	0	0	1	0	2
	1	0	4	5	7	6
	1	1	12	13	15	14
	0	1	8	9	11	10

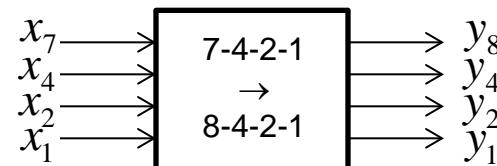
		y ₂				
		x ₂ x ₁	00	01	11	10
		x ₇	0	0	3	2
	0	0	0	1	0	2
	1	0	4	5	7	6
	1	1	12	13	15	14
	0	1	8	9	11	10

		y ₁				
		x ₂ x ₁	00	01	11	10
		x ₇	0	0	3	2
	0	0	0	1	0	2
	1	0	4	5	7	6
	1	1	12	13	15	14
	0	1	8	9	11	10

	7	4	2	1		8	4	2	1
	x_7	x_4	x_2	x_1		y_8	y_4	y_2	y_1
(0)	0	0	0	0	0	0	0	0	0
(1)	0	0	0	1	1	0	0	0	1
(2)	0	0	1	0	2	0	0	1	0
(3)	0	0	1	1	3	0	0	1	1
(4)	0	1	0	0	4	0	1	0	0
(5)	0	1	0	1	5	0	1	0	1
(6)	0	1	1	0	6	0	1	1	0
(8)	1	0	0	0	7	0	1	1	1
(9)	1	0	0	1	8	1	0	0	0
(10)	1	0	1	0	9	1	0	0	1

8.4

Code converter



		y ₈					
		x ₂	x ₁	00	01	11	10
		x ₇	x ₄	0	1	3	2
0	0	0	0	0	1	3	2
0	1	4	0	5	0	7	6
1	0	12	-	13	15	14	-
1	1	-	-	-	-	-	-
0	0	8	0	1	11	10	1

		y ₄					
		x ₂	x ₁	00	01	11	10
		x ₇	x ₄	0	1	3	2
0	0	0	0	0	1	3	2
0	1	0	4	1	5	1	7
1	0	1	12	13	15	14	-
1	1	-	-	-	-	-	-
0	0	1	8	1	9	0	11

		y ₂					
		x ₂	x ₁	00	01	11	10
		x ₇	x ₄	0	1	3	2
0	0	0	0	0	1	3	2
0	1	0	4	0	5	7	6
1	0	1	12	13	15	14	-
1	1	-	-	-	-	-	-
0	0	1	8	1	9	0	11

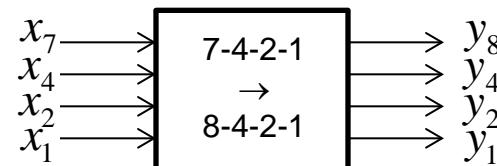
		y ₁					
		x ₂	x ₁	00	01	11	10
		x ₇	x ₄	0	1	3	2
0	0	0	0	0	1	3	2
0	1	0	4	0	5	7	6
1	0	1	12	13	15	14	-
1	1	-	-	-	-	-	-
0	0	1	8	1	9	0	11

$$y_8 = x_7 x_2 + x_7 x_1$$

	7	4	2	1		8	4	2	1
	x_7	x_4	x_2	x_1		y_8	y_4	y_2	y_1
(0)	0	0	0	0	0	0	0	0	0
(1)	0	0	0	1	1	0	0	0	1
(2)	0	0	1	0	2	0	0	1	0
(3)	0	0	1	1	3	0	0	1	1
(4)	0	1	0	0	4	0	1	0	0
(5)	0	1	0	1	5	0	1	0	1
(6)	0	1	1	0	6	0	1	1	0
(8)	1	0	0	0	7	0	1	1	1
(9)	1	0	0	1	8	1	0	0	0
(10)	1	0	1	0	9	1	0	0	1

8.4

Code converter



				y_8
				$x_2 x_1$
				00 01 11 10
x_7	0	0	1	0 1 3 2 0
x_4	0	0	1	0 1 3 2 0
0	0	4	5	0 5 7 - 6 0
1	1	0	1	1 1 7 - 5 1
1	1	12	13	12 13 15 14
1	1	-	-	- - - -
0	0	8	9	1 0 11 10 1

				y_4
				$x_2 x_1$
				00 01 11 10
x_7	0	0	1	0 1 3 2 0
x_4	0	0	1	0 1 3 2 0
0	0	4	5	0 5 7 - 6 0
1	1	0	1	1 1 7 - 5 1
1	1	12	13	12 13 15 14
1	1	-	-	- - - -
0	0	8	9	1 0 11 10 1

				y_2
				$x_2 x_1$
				00 01 11 10
x_7	0	0	1	0 1 3 2 1
x_4	0	0	1	0 1 3 2 1
0	0	4	5	0 5 7 - 6 1
1	1	0	1	1 1 7 - 5 1
1	1	12	13	12 13 15 14
1	1	-	-	- - - -
0	0	8	9	1 0 11 10 0

				y_1
				$x_2 x_1$
				00 01 11 10
x_7	0	0	1	0 1 3 2 0
x_4	0	0	1	0 1 3 2 0
0	0	4	5	0 5 7 - 6 0
1	1	0	1	0 5 1 7 - 6 0
1	1	12	13	12 13 15 14
1	1	-	-	- - - -
0	0	8	9	1 0 11 10 1

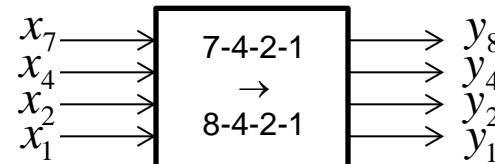
$$y_8 = x_7 x_2 + x_7 x_1$$

$$y_4 = x_4 + x_7 \overline{x_2} \overline{x_1}$$

	7	4	2	1		8	4	2	1
	x_7	x_4	x_2	x_1		y_8	y_4	y_2	y_1
(0)	0	0	0	0	0	0	0	0	0
(1)	0	0	0	1	1	0	0	0	1
(2)	0	0	1	0	2	0	0	1	0
(3)	0	0	1	1	3	0	0	1	1
(4)	0	1	0	0	4	0	1	0	0
(5)	0	1	0	1	5	0	1	0	1
(6)	0	1	1	0	6	0	1	1	0
(8)	1	0	0	0	7	0	1	1	1
(9)	1	0	0	1	8	1	0	0	0
(10)	1	0	1	0	9	1	0	0	1

8.4

Code converter



y_8			
x_7	x_2	x_1	
0	0	1	11
0	0	0	20
0	4	5	7
1	0	0	60
1	12	13	15
1	-	-	14
0	8	9	11
0	1	11	10

y_4			
x_7	x_2	x_1	
0	0	1	11
0	0	0	20
0	4	5	7
1	1	1	61
1	12	13	15
1	-	-	14
0	1	9	10

y_2			
x_7	x_2	x_1	
0	0	1	11
0	0	0	21
0	4	5	7
1	0	0	61
1	12	13	15
1	-	-	14
0	1	9	10

y_1			
x_7	x_2	x_1	
0	0	1	11
0	1	1	20
0	4	5	7
1	0	0	60
1	12	13	15
1	-	-	14
0	1	9	10

$$y_8 = x_7 x_2 + x_7 x_1$$

$$y_4 = x_4 + x_7 \overline{x_2} \overline{x_1}$$

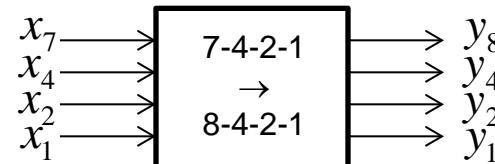
$$y_2 = \overline{x_7} x_2 + x_7 \overline{x_2} \overline{x_1}$$

$$y_1 = \overline{x_7} \overline{x_1} + x_7 x_2 + x_7 \overline{x_2} \overline{x_1}$$

	7	4	2	1		8	4	2	1
	x_7	x_4	x_2	x_1		y_8	y_4	y_2	y_1
(0)	0	0	0	0	0	0	0	0	0
(1)	0	0	0	1	1	0	0	0	1
(2)	0	0	1	0	2	0	0	1	0
(3)	0	0	1	1	3	0	0	1	1
(4)	0	1	0	0	4	0	1	0	0
(5)	0	1	0	1	5	0	1	0	1
(6)	0	1	1	0	6	0	1	1	0
(8)	1	0	0	0	7	0	1	1	1
(9)	1	0	0	1	8	1	0	0	0
(10)	1	0	1	0	9	1	0	0	1

8.4

Code converter



Common groupings can provide for shared gates!

				y_8
x_7	x_2	x_1		
0	00	01	11	10
0	0	0	3	2
0	4	5	7	6
1	0	0	-	0
1	12	13	15	14
1	-	-	-	-
0	8	9	11	10
0	1	-	1	1

				y_4
x_7	x_2	x_1		
0	00	01	11	10
0	0	0	3	2
0	4	5	7	6
1	1	1	-	1
1	12	13	15	14
1	-	-	-	-
0	1	0	11	10
0	1	-	1	1

				y_2
x_7	x_2	x_1		
0	00	01	11	10
0	0	0	1	1
0	4	0	-	1
1	12	13	15	14
1	-	-	-	-
0	1	0	11	10
0	1	-	1	1

				y_1
x_7	x_2	x_1		
0	00	01	11	10
0	0	1	1	0
0	4	5	7	6
1	1	1	-	0
1	12	13	15	14
1	-	-	-	-
0	1	0	11	10
0	1	-	1	1

$$y_8 = x_7 x_2 + x_7 x_1$$

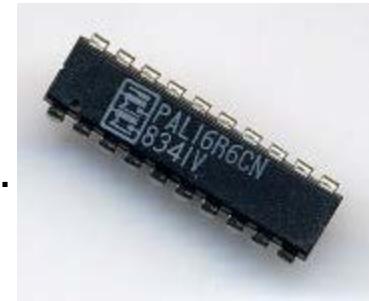
$$y_4 = x_4 + x_7 \overline{x_2} \overline{x_1}$$

$$y_2 = \overline{x_7} x_2 + x_7 \overline{x_2} \overline{x_1}$$

$$y_1 = \overline{x_7} \overline{x_1} + x_7 x_2 + x_7 \overline{x_2} \overline{x_1}$$

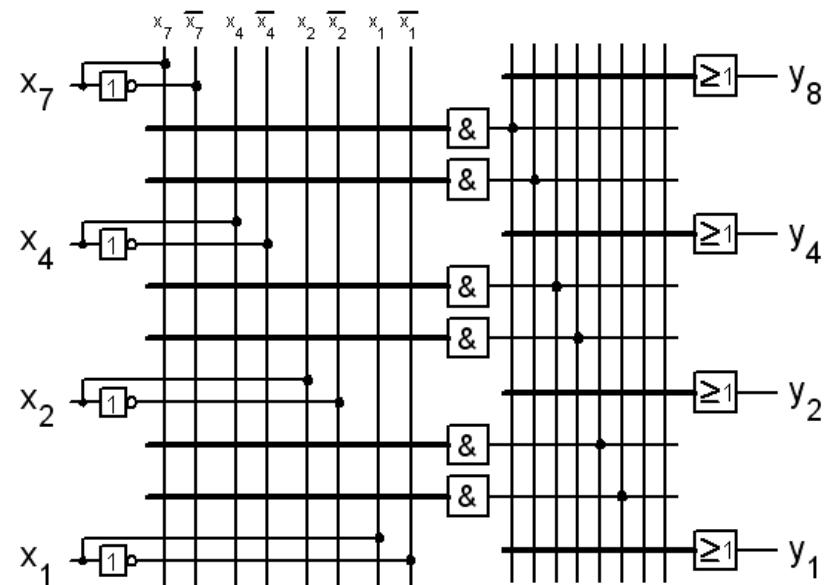
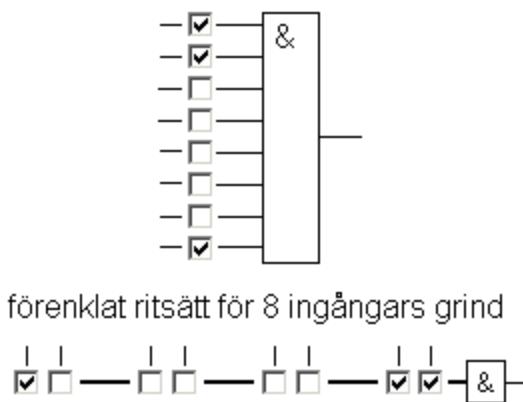
8.4

PLA circuits containing programmable AND and OR gates.
(This turned out to be unnecessarily complex, so the common chips became PAL circuits with only the AND network programmable).



The gates have many programmable input connections. The many inputs are usually drawn in a "simplified" way.

Programmerbar logik



8.4

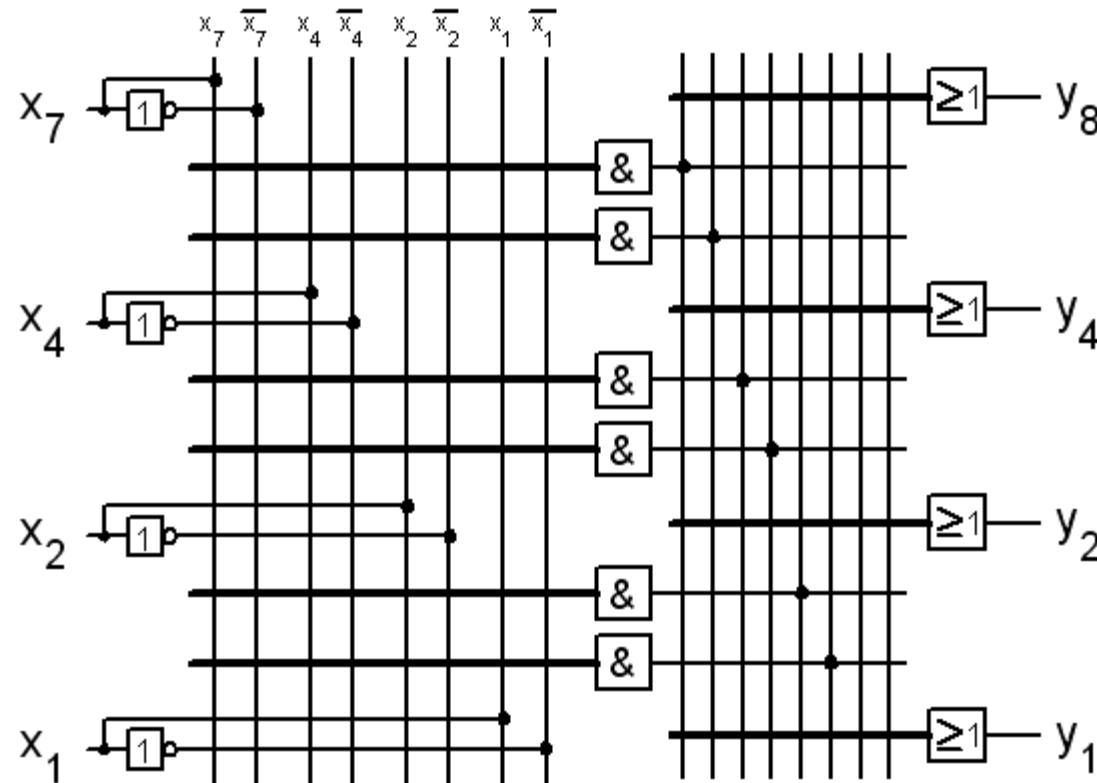
$$y_8 = \boxed{x_7 x_2} + x_7 x_1$$

$$y_4 = x_4 + \boxed{x_7 \overline{x_2} \overline{x_1}}$$

$$y_2 = \overline{x_7} x_2 + \boxed{x_7 \overline{x_2} \overline{x_1}}$$

$$y_1 = \overline{x_7} x_1 + \boxed{x_7 x_2} + \boxed{x_7 \overline{x_2} \overline{x_1}}$$

Shared-gates!



8.4

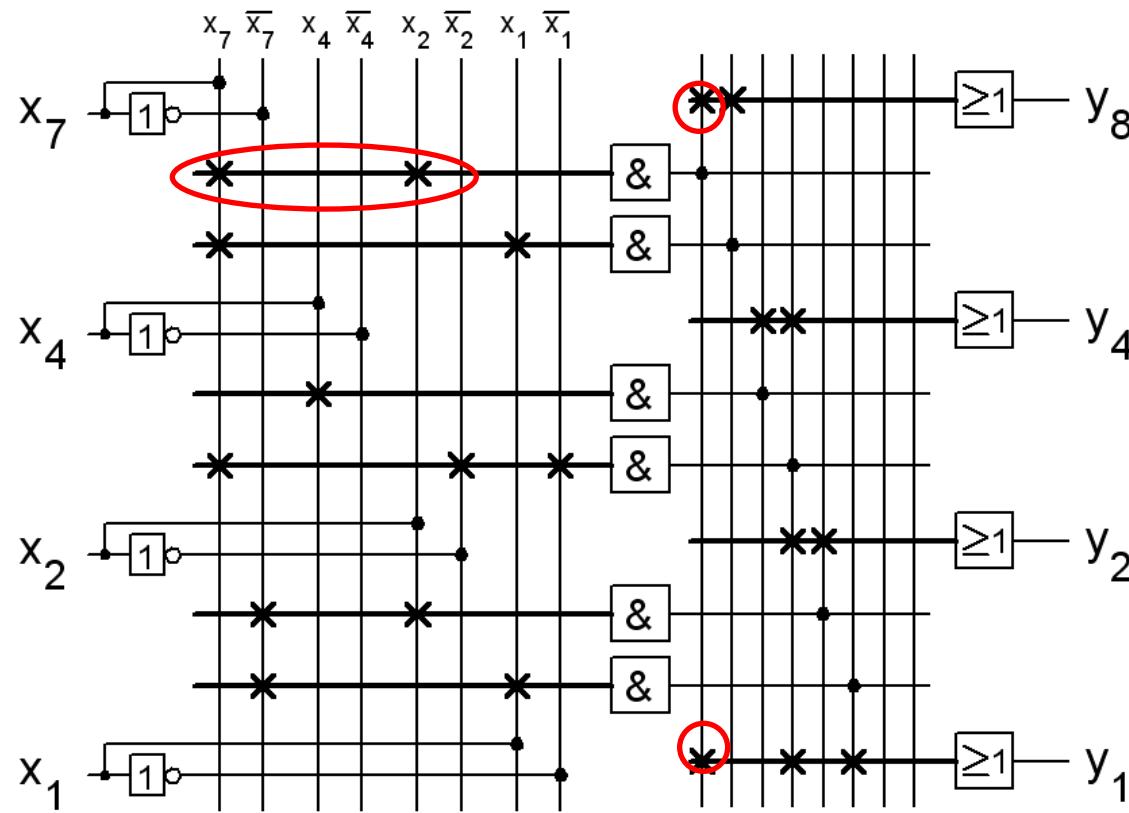
$$y_8 = \boxed{x_7 x_2} + x_7 x_1$$

$$y_4 = x_4 + \boxed{x_7 \overline{x_2} \overline{x_1}}$$

$$y_2 = \overline{x_7} x_2 + \boxed{x_7 \overline{x_2} \overline{x_1}}$$

$$y_1 = \overline{x_7} x_1 + \boxed{x_7 x_2} + \boxed{x_7 \overline{x_2} \overline{x_1}}$$

Shared-gates!



8.4

$$y_8 = \boxed{x_7 x_2} + x_7 x_1$$

$$y_4 = x_4 + \boxed{x_7 \overline{x_2} \overline{x_1}}$$

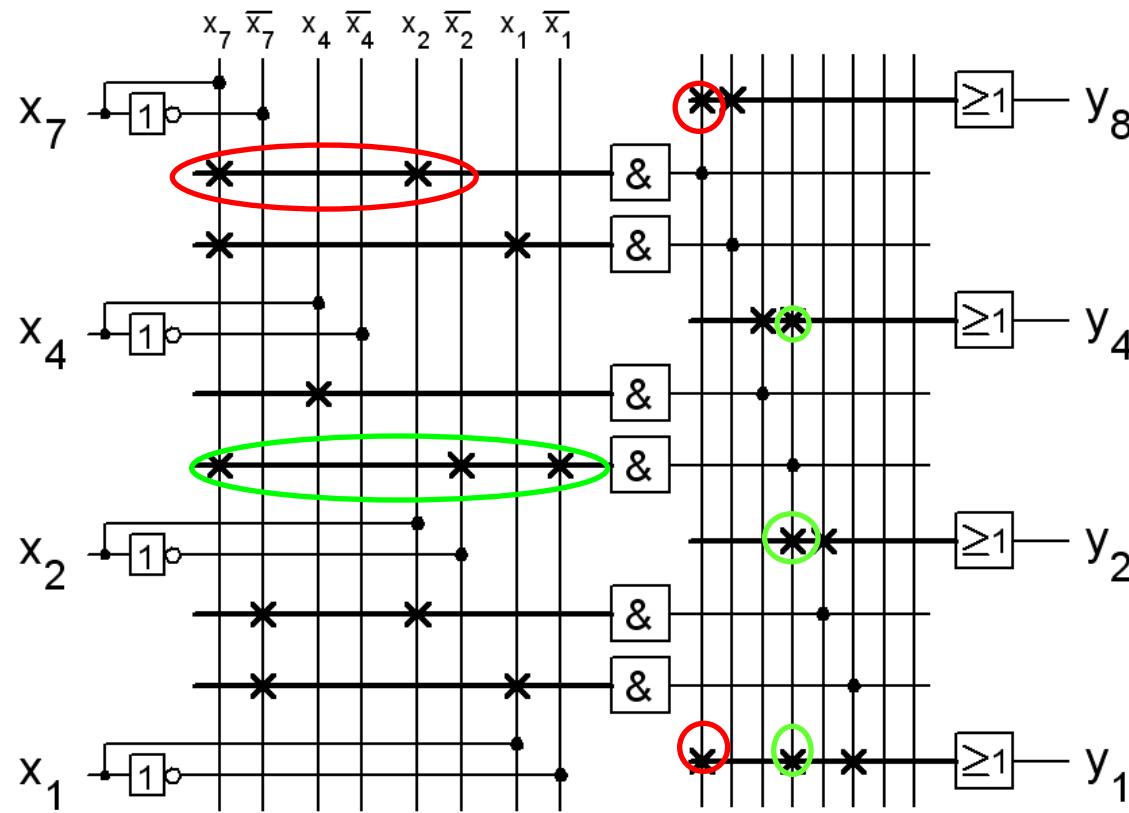
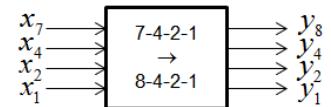
$$y_2 = \overline{x_7} x_2 + \boxed{x_7 \overline{x_2} \overline{x_1}}$$

$$y_1 = \overline{x_7} x_1 + \boxed{x_7 x_2} + \boxed{x_7 \overline{x_2} \overline{x_1}}$$

Shared-gates!

One chip

Code converter



Real numbers

Decimal point „,“ and Binary point „.“

$$10,3125_{10} = 1010.0101_2$$

Bin → Dec

$$\begin{array}{cccccccc} 1 & 0 & 1 & 0 & . & 0 & 1 & 0 & 1 \\ 2^3 & 2^2 & 2^1 & 2^0 & . & 2^{-1} & 2^{-2} & 2^{-3} & 2^{-4} \\ 8 & 4 & 2 & 1 & & 0,5 & 0,25 & 0,125 & 0,0625 \end{array}$$

$$8 + 0 + 2 + 0 + 0 + 0,25 + 0 + 0,0625 = 10,3125$$

Ex. 1.2b

$110100.010_2 =$

Ex. 1.2b

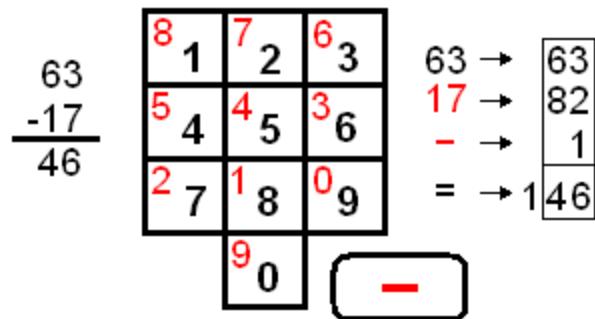
$$110100.010_2 =$$

$$= (2^5 + 2^4 + 2^2 + 2^{-2} = 32 + 16 + 4 + 0.25) =$$

$$= 52,25_{10}$$

Calculation with complement

Subtraction with an adding machine = counting with the complement



$$63 - 17 = 46$$

The number -17 is entered with red digits 17 and gets 82. When the - key is pressed 1 is added. The result is: $63+82+1 = 146$. If only two digits are shown: 46



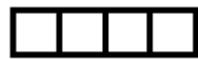
2-complement

$$\begin{array}{r} 7 \\ -3 \\ \hline 4 \end{array} \quad \begin{array}{c} 1 \boxed{0} \boxed{0} 1 \\ - \end{array} \quad \begin{array}{l} (7) \ 0111 \rightarrow 0111 \\ (3) \ 0011 \rightarrow 1100 \\ - \rightarrow 0001 \\ = \rightarrow 10100 \end{array} \quad (4)$$

The binary number 3, 0011, gets negative -3 if one inverts the digits and adds one, 1101.

Register arithmetic

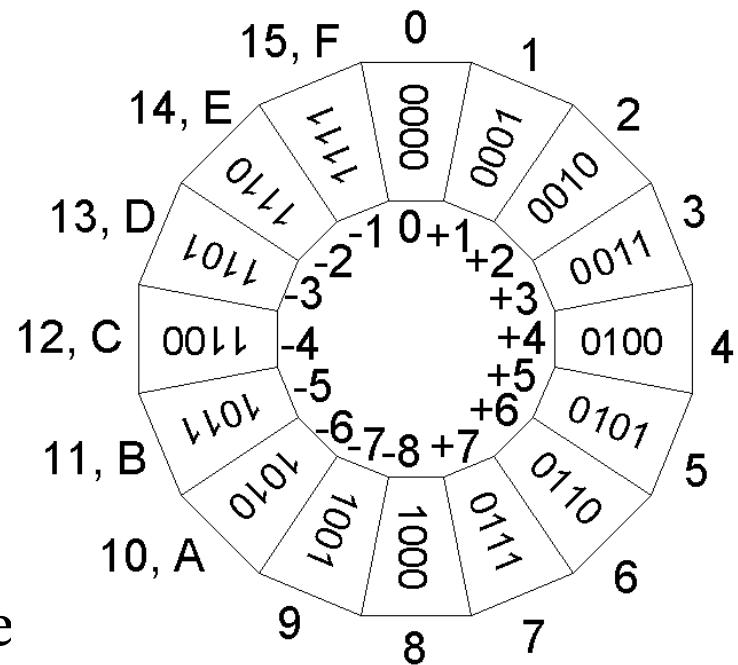
- Computer registers are "rings"



A four bit register could contain $2^4 = 16$ numbers.

Either 8 positive (+0...+7) and 8 negative (-1...-8) "**signed integers**", or 16 (0...F) "**unsigned integers**".

If the register is full +1 makes the register to the "turn around".



Register width

- 4 bit is called a **Nibble**. The register contains $2^4 = 16$ numbers. 0...15, -8...+7
- 8 bit is called a **Byte**. The register contains $2^8 = 256$ numbers 0...255, -128...+127
- 16 bit is a **Word**. $2^{16} = 65536$ numbers.
0...65535, -32768...+32767

Today, general sizes are now 32 bits (Double Word) and 64 bits (Quad Word)..

Ex. 1.8

Write the following signed numbers with two's complement notation,
 $x = (x_6, x_5, x_4, x_3, x_2, x_1, x_0)$.

a) -23

b) -1 =

c) +38 =

d) -64 =

Ex. 1.8

Write the following signed numbers with two's complement notation,
 $x = (x_6, x_5, x_4, x_3, x_2, x_1, x_0)$.

a) $-23 = (+23_{10} = 0010111_2 \rightarrow -23_{10} = 1101000_2 + 1_2) = 1101001_2$
 $= 105_{10}$

b) $-1 =$

c) $+38 =$

d) $-64 =$

Ex. 1.8

Write the following signed numbers with two's complement notation,
 $x = (x_6, x_5, x_4, x_3, x_2, x_1, x_0)$.

a) $-23 = (+23_{10} = 0010111_2 \rightarrow -23_{10} = 1101000_2 + 1_2) = 1101001_2 = 105_{10}$

b) $-1 = (+1_{10} = 0000001_2 \rightarrow -1_{10} = 1111110_2 + 1_2) = 1111111_2 = 127_{10}$

c) $+38 =$

d) $-64 =$

Ex. 1.8

Write the following signed numbers with two's complement notation,
 $x = (x_6, x_5, x_4, x_3, x_2, x_1, x_0)$.

a) $-23 = (+23_{10} = 0010111_2 \rightarrow -23_{10} = 1101000_2 + 1_2) = 1101001_2 = 105_{10}$

b) $-1 = (+1_{10} = 0000001_2 \rightarrow -1_{10} = 1111110_2 + 1_2) = 1111111_2 = 127_{10}$

c) $+38 = (32_{10} + 4_{10} + 2_{10}) = 0100110_2 = 38_{10}$

d) $-64 =$

Ex. 1.8

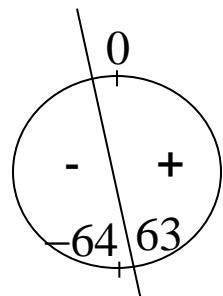
Write the following signed numbers with two's complement notation,
 $x = (x_6, x_5, x_4, x_3, x_2, x_1, x_0)$.

a) $-23 = (+23_{10} = 0010111_2 \rightarrow -23_{10} = 1101000_2 + 1_2) = 1101001_2 = 105_{10}$

b) $-1 = (+1_{10} = 0000001_2 \rightarrow -1_{10} = 1111110_2 + 1_2) = 1111111_2 = 127_{10}$

c) $+38 = (32_{10} + 4_{10} + 2_{10}) = 0100110_2 = 38_{10}$

d) $-64 = (+64_{10} = 1000000_2)$ is a to big positive number (for 7 bits)!
But will still function for $-64_{10} \rightarrow 0111111_2 + 1_2) = 1000000_2 = 64_{10}$



Ex. 2.1

a) $110 + 010$ b) $1110 + 1001$

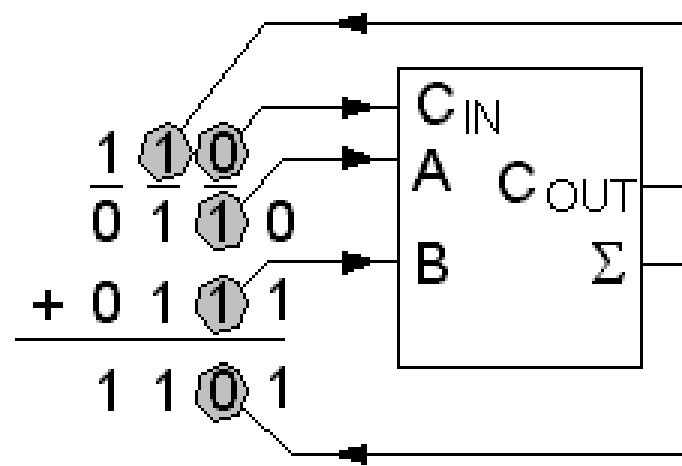
c) 11 0011.01 + 111.1 d) 0.1101 + 0.1110

$$\begin{array}{r} \text{a) } \begin{array}{r} \frac{1}{1} \\ \frac{1}{1} \end{array} \end{array}$$

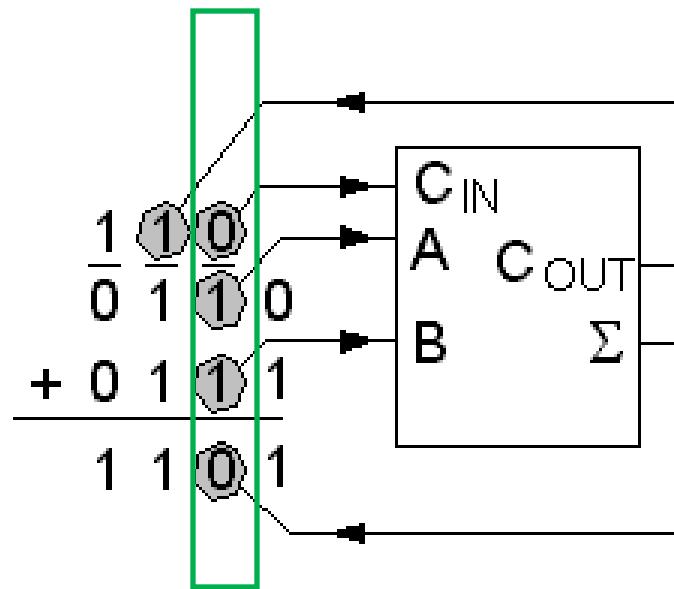
$$\begin{array}{r} \text{b) } \begin{array}{r} 1 \\ 1 \ 1 \ 1 \ 0 \\ + \ 1 \ 0 \ 0 \ 1 \\ \hline 1 \ 0 \ 1 \ 1 \ 1 \end{array} \end{array}$$

d)
$$\begin{array}{r}
 \frac{1}{0.1} \\
 + 0.1 \\
 \hline
 1.1
 \end{array}
 \quad
 \begin{array}{r}
 1 \\
 0 \\
 1 \\
 1
 \end{array}$$

Full adder



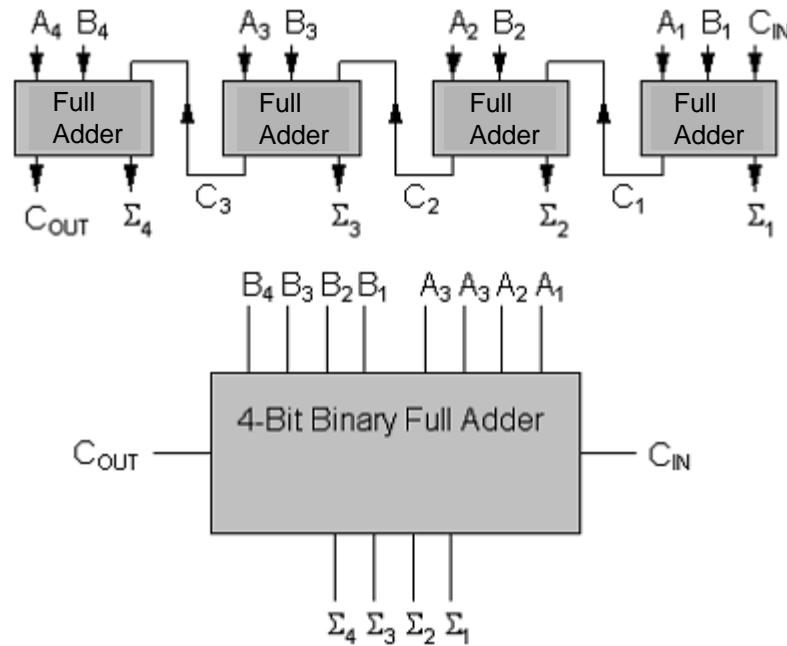
Full adder



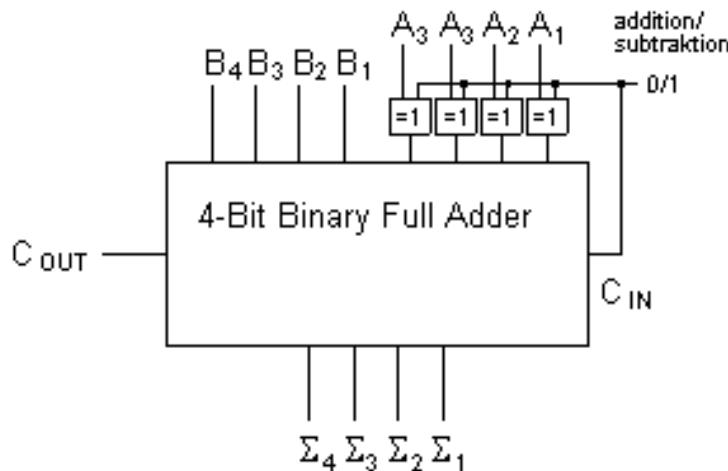
A logic circuit that makes a binary addition on any bit position with two binary numbers is called a full adder.

4-bit adder

An addition circuit for binary four bit numbers thus consists of four fulladder circuits.

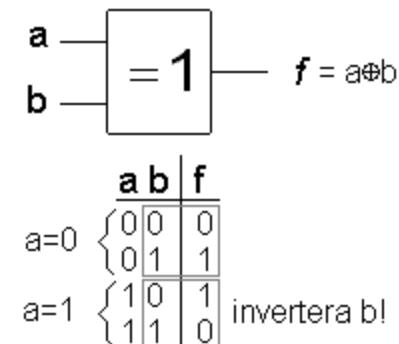


Subtraction?



Subtracting the binary numbers can be done with the two-complement. Negative numbers are represented as the true complement, which means that all bits are inverted and a one is added. The adder is then used also for subtraction.

The inversion of the bits could be done with XOR-gates, and a one could then be added to the number by letting $C_{IN} = 1$.



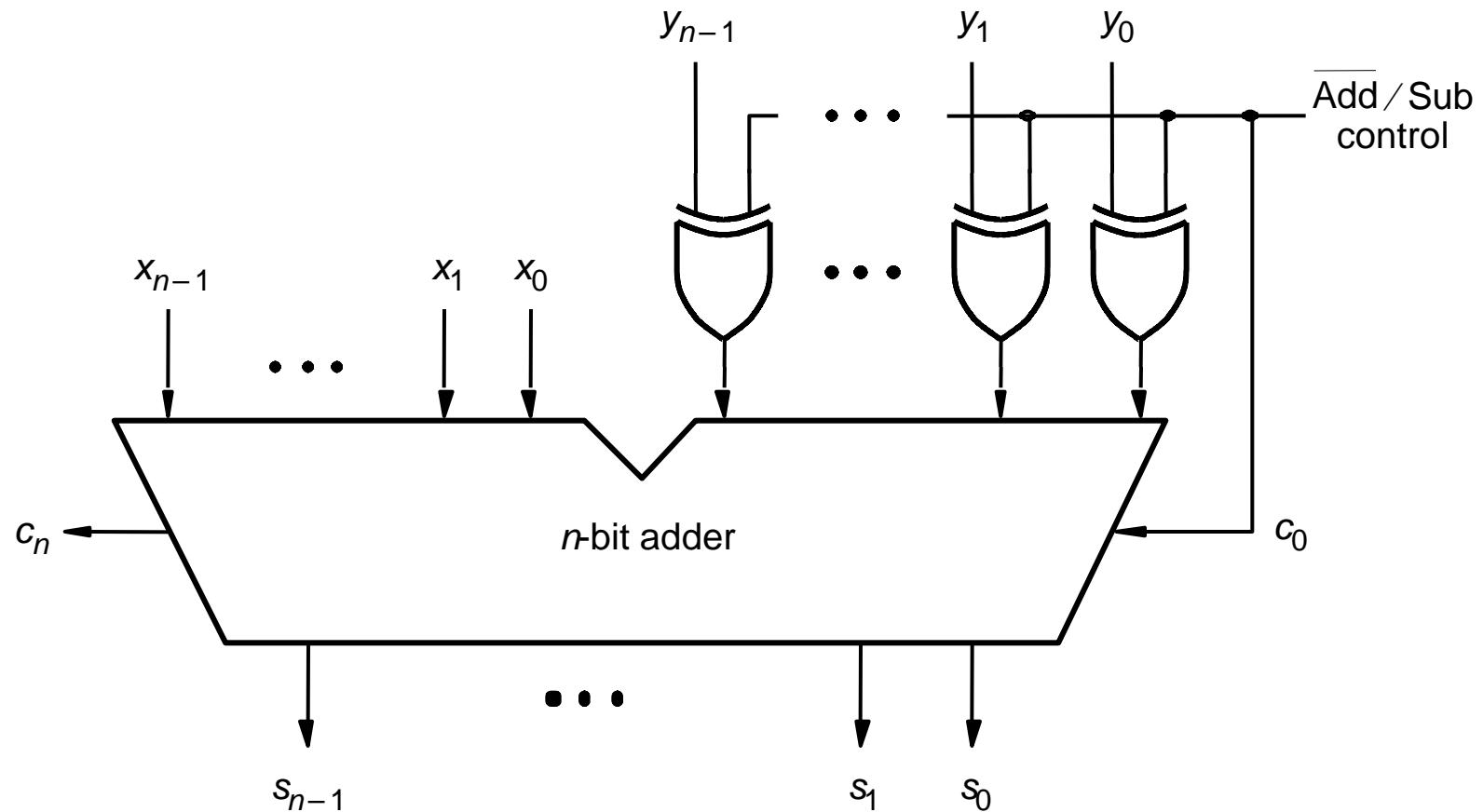
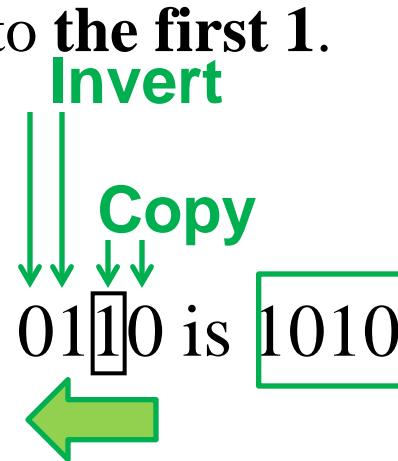


Figure 5.13. Adder/subtractor unit.

2-complement "fast"

- In order to easily produce 2's complement of a binary number, you can use the following procedure:
 - Start from right
 - Copy all bits from all zeroes to **the first 1.**
 - Invert all the rest of the bits

Example: 2-complement of



Ex. 2.2

Add or subtract (add with the corresponding negative number) the numbers below. The numbers are represented as binary 2-complement 4-bit numbers (nibble).

- a) $1 + 2$
- b) $4 - 1$
- c) $7 - 8$
- d) $-3 - 5$

The negative number that are used in the examples:

$$-1_{10} = (+1_{10} = 0001_2 \rightarrow -1_{10} = 1110_2 + 1_2) = 1111_2$$

$$-8_{10} = (+8_{10} = 1000_2 \rightarrow -8_{10} = 0111_2 + 1_2) = 1000_2$$

$$-3_{10} = (+3_{10} = 0011_2 \rightarrow -3_{10} = 1100_2 + 1_2) = 1101_2$$

$$-5_{10} = (+5_{10} = 0101_2 \rightarrow -5_{10} = 1010_2 + 1_2) = 1011_2$$

2.2

$$-1_{10} = 1111_2$$

$$-8_{10} = 1000_2$$

$$-3_{10} = 1101_2$$

$$-5_{10} = 1011_2$$

$$1+2=3$$

$$\begin{array}{r} a) \quad 1+2=3 \\ \begin{array}{r} \begin{array}{r} 0 & 0 & 0 & 1 \\ + & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 1 & 1 \end{array} \\ =1 \\ =2 \\ =3 \end{array} \end{array}$$

$$4-1=3$$

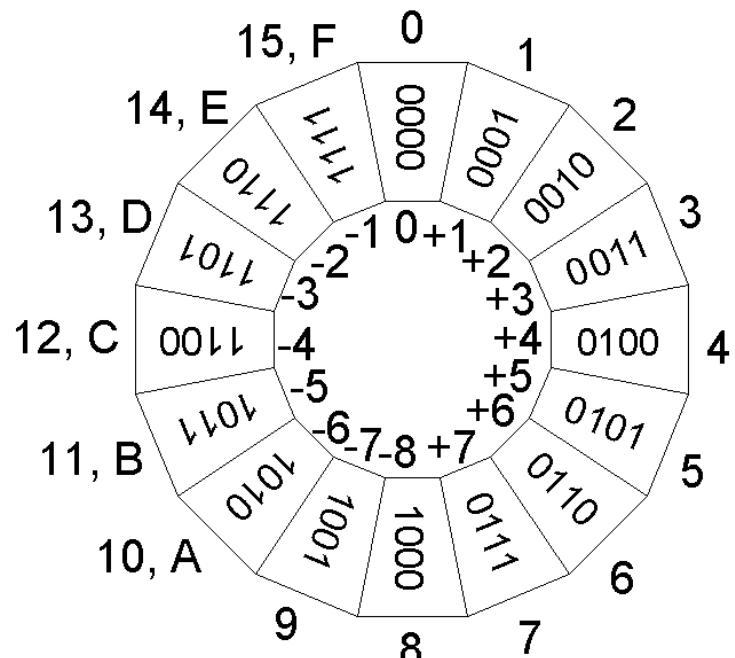
$$\begin{array}{r} b) \quad 4-1=3 \\ \begin{array}{r} \begin{array}{r} 1 & 1 \\ 0 & 1 & 0 & 0 \\ + & 1 & 1 & 1 & 1 \\ \hline 0 & 0 & 1 & 1 \end{array} \\ =4 \\ =-1 \\ =3 \end{array} \end{array}$$

$$7-8=-1$$

$$\begin{array}{r} c) \quad 7-8=-1 \\ \begin{array}{r} \begin{array}{r} 0 & 1 & 1 & 1 \\ + & 1 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 1 \end{array} \\ =7 \\ =-8 \\ =-1 \end{array} \end{array}$$

$$-3-5=-8$$

$$\begin{array}{r} d) \quad -3-5=-8 \\ \begin{array}{r} \begin{array}{r} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ + & 1 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 \end{array} \\ =-3 \\ =-5 \\ =-8 \end{array} \end{array}$$



Ex. 2.3 a,b

Multiplicate by hand the following pairs of unsigned binary numbers.

a) 110·010 b) 1110·1001

$$110 \cdot 010 = (6 \cdot 2 = 12) = 1100$$

$$1110 \cdot 1001 = 1111110$$

a)
$$\begin{array}{r} 1 & 1 & 0 \\ \times & 0 & 1 & 0 \\ \hline 0 & 0 & 0 \end{array} = 6$$

$$\begin{array}{r} 1 & 1 & 0 \\ \times & 1 & 0 & 0 & 1 \\ \hline 1 & 1 & 1 & 0 \end{array} = 14$$

$$\begin{array}{r} 1 & 1 & 0 \\ + & 0 & 0 & 0 \\ \hline 0 & 1 & 1 & 0 & 0 \end{array} = 12$$

b)
$$\begin{array}{r} 1 & 1 & 1 & 0 \\ \times & 1 & 0 & 0 & 1 \\ \hline 1 & 1 & 1 & 0 \end{array} = 14$$

$$\begin{array}{r} 0 & 0 & 0 & 0 \\ + & 1 & 1 & 1 & 0 \\ \hline 1 & 1 & 1 & 1 & 0 \end{array} = 9$$

$$\begin{array}{r} 0 & 0 & 0 & 0 \\ + & 1 & 1 & 1 & 0 \\ \hline 1 & 1 & 1 & 1 & 1 & 0 \end{array} = 126$$

Ex. 2.3 c,d

Multiplicate by hand the following pairs of unsigned binary numbers.

$$\begin{array}{r} 110011.01 \cdot 111.1 = \\ = 110000000.011 \end{array}$$

c)

$$\begin{array}{r} 11001101 \\ \times \quad 1111 \\ \hline 11001101 \\ 11001101 \\ 11001101 \\ + 11001101 \\ \hline 110000000011 \end{array}$$

$$= 110000000.011$$

$$(51,25 \cdot 7,5 = 384,376)$$

$$\begin{array}{r} 0.1101 \cdot 0.1110 = \\ = 0.10110110 \end{array}$$

d)

$$\begin{array}{r} 1101 \\ \times \quad 1110 \\ \hline 0000 \\ 1101 \\ 1101 \\ + \quad 1101 \\ \hline 10110110 \end{array}$$

$$= 0.10110110$$

$$(0,8125 \cdot 0,875 = 0,7109375)$$

Fixpoint multiplication is an "integer multiplication", the binarypoint is inserted in the result.

Ex. 2.4

Divide by hand the following pairs of unsigned binary numbers.

Method the Stairs:

$$110/010 = (6/2 = 3) = 011$$

Ex. 2.4

Divide by hand the following pairs of unsigned binary numbers.

Method the Stairs:

$$110/010 = (6/2=3) = 011$$

$$1110/1001 = (14/9) = 1.10 \dots$$

a)

$$\begin{array}{r} 1\ 1 \\ \hline 1\ 0 \left[\begin{array}{r} 1\ 1\ 0 \\ - 1\ 0 \\ \hline 1\ 0 \\ - 1\ 0 \\ \hline 0 \end{array} \right] \end{array}$$

b)

$$\begin{array}{r} 1\ 1\ 0\ 0\ 0\ 1\ \dots \\ \hline 1\ 0\ 0\ 1 \left[\begin{array}{r} 1\ 1\ 1\ 0 \\ - 1\ 0\ 0\ 1 \\ \hline 1\ 0\ 1\ 0 \\ - 1\ 0\ 0\ 1 \\ \hline 1\ 0\ 0\ 0\ 0 \\ - 1\ 0\ 0\ 1 \\ \hline 1\ 1\ 1\ \dots \end{array} \right] \end{array}$$

If integer division the answer will be 1.

Ex 2.4

Divide by hand the following pairs of unsigned binary numbers.

Method Short division:

a) $110/010 = (6/2=3) = 011$

$$\begin{array}{r} 1 \\ \boxed{1} \cancel{1} 0 = \\ \hline 10 \end{array} \quad \begin{array}{r} 1 \\ 1 \cancel{1} 0 = 1 \\ \hline 10 \end{array} \quad \begin{array}{r} 1 \\ 1 \cancel{1} \boxed{0} = 11 \\ \hline 10 \end{array}$$

Ex 2.4

Divide by hand the following pairs of unsigned binary numbers.

Method Short division:

b) $1110/1001 = (14/9 = 1,55\dots) = 1.10\dots$

$$\frac{1110}{1001} = \begin{array}{r} 101 \\ \hline 1001 \end{array} = 1 \quad \frac{1110.}{1001} = \begin{array}{r} 1010 \\ \hline 1001 \end{array} = 1. \quad \frac{1110.}{1001} = \begin{array}{r} 1 \\ \hline 1001 \end{array} = 1.1 \quad \dots$$

If integer division the answer will be 1.

IEEE – 32 bit float

$+5,5_{10} = +101.1_2$
Normalisering: $+1.011 \cdot 2^{+2}$

tecken	exponent	signifikand
31	30	23
0	10000001	01100000000000000000000000000000
		0

1. $+2 + 127 = 129$
 10000001
excess - 127 !

The exponent is written excess-127. It is then possible to sort float by size with ordinary integer arithmetic!

Dec \rightarrow IEEE-754

2.5 Float format

IEEE 32 bit float

s	eeeeeeee	ffffffffffffffffffffffffffff		
31	30	23	22	0

2.5 Float format

IEEE 32 bit float

s	eeeeeeee	ffffffffffffffffffffffffffff		
31	30	23	22	0

What is:

4	0	C	8	0	0	0	0
01000000110010000000000000000000							

2.5 Float format

IEEE 32 bit float

s	eeeeeeee	ffffffffffffffffffffffffffff		
31	30	23	22	0

What is:

4	0	C	8	0	0	0	0
01000000110010000000000000000000							
0	10000001	10010000000000000000000000000000					
+	129-127	1	+ 0.5	+ 0.0625			

2.5 Float format

IEEE 32 bit float

s	eeeeeeee	ffffffffffffffffffffffffffffffff		
31	30	23	22	0

What is:

4	0	C	8	0	0	0	0
01000000110010000000000000000000							

0 10000001 100100000000000000000000

+ 129-127 1 + 0.5+0.0625

$+1,5625 \cdot 2^2 = +6,25$

IEEE-754 Floating-Point Conversion from 32-bit Hexadecimal to Floating-Point - Mozilla Firefox

Arkiv Redigera Visa Historik Bokmärken Verktyg Hjälp

http://babbage.cs.qc.cuny.edu/IEEE-754/32bit.html

IEEE-754 Floating-Point Conversion
From 32-bit Hexadecimal Representation
To Decimal Floating-Point
Along with the Equivalent 64-bit Hexadecimal and Binary Patterns

Enter the 32-bit hexadecimal representation of a floating-point number here,
then click the **Compute** button.

Hexadecimal Representation:

Results:

Decimal Value Entered:

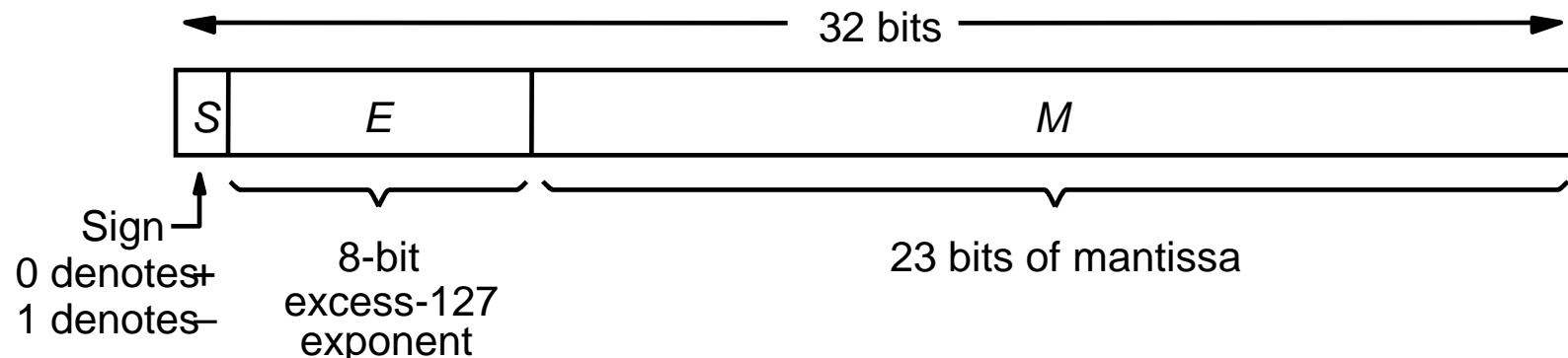
Single precision (32 bits):

Binary: Status:

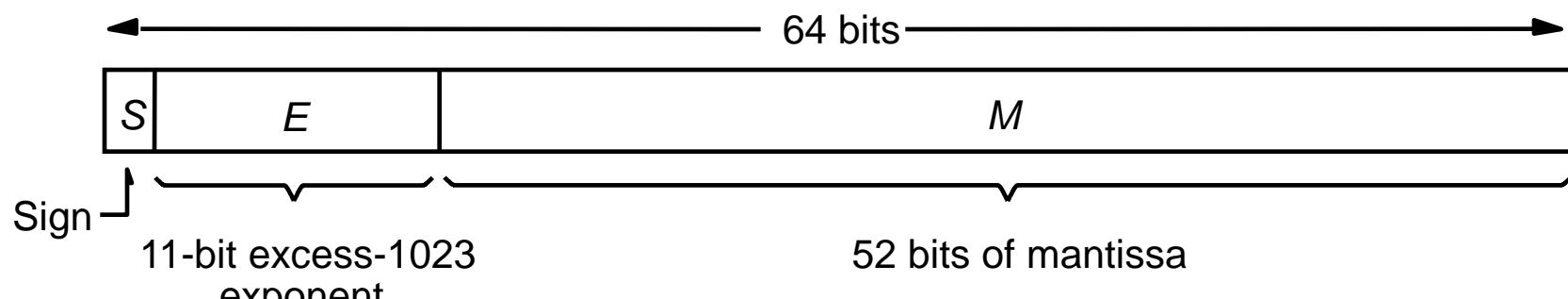
Bit 31 Sign Bit <input type="text" value="0"/> 0: + 1: -	Bits 30 - 23 Exponent Field <input type="text" value="10000001"/> Decimal value of exponent field and exponent <input type="text" value="129"/> - 127 = <input type="text" value="2"/>	Bits 22 - 0 Significand <input type="text" value="1.100100000000000000000000"/> Decimal value of the significand <input type="text" value="1.5625000"/>
--	--	---

Dec → IEEE-754

William Sandqvist william@kth.se



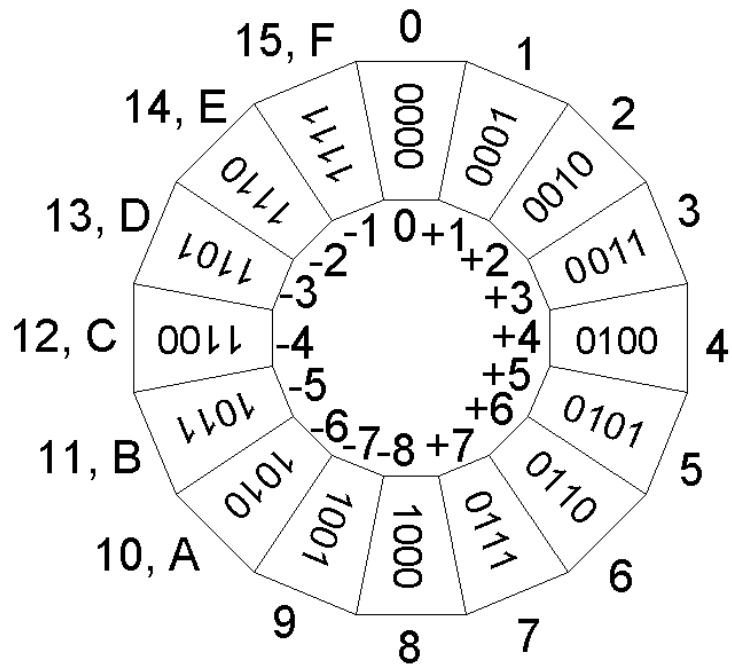
(a) Single precision



(b) Double precision

Figure 5.34. IEEE Standard floating-point formats.

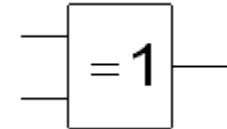
Overflow



When using signed numbers the sum of two positive numbers could be incorrectly negative (eg. "+4" + "+5" = "-7"), in the same way the sum of two negative numbers could incorrectly be positive (eg. "-6" + "-7" = "+3").

This is called **Overflow**.

Logic to detect overflow



For 4-bit-numbers

Overflow if c_3 and c_4 are *different*

Otherwise it's not overflow

XOR detects
"not equal"

$$\text{Overflow} = c_3 \bar{c}_4 + \bar{c}_3 c_4 = c_3 \oplus c_4$$

For n -bit-numbers

$$\text{Overflow} = c_{n-1} \oplus c_n$$

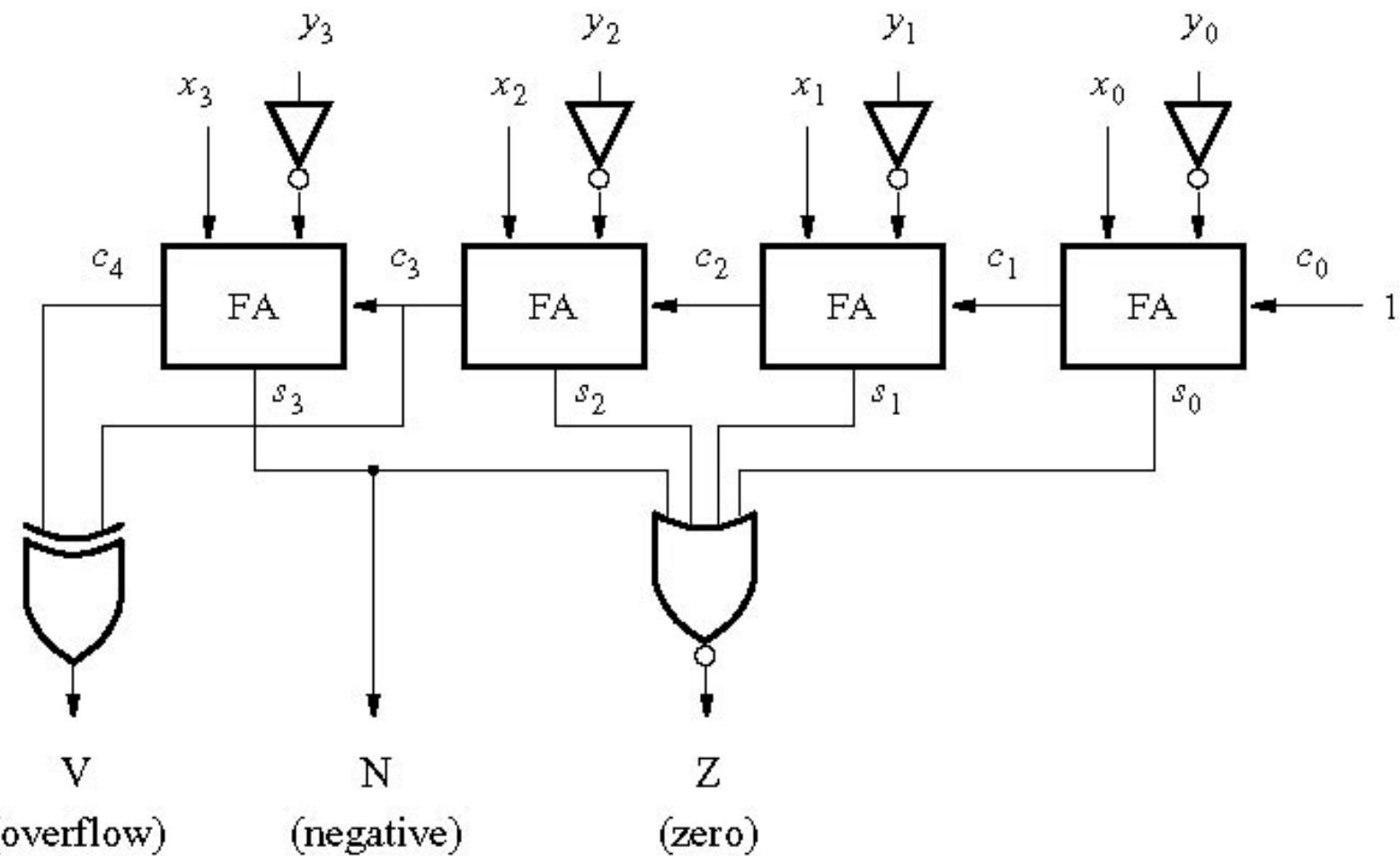


Figure 5.42. A **comparator** circuit.

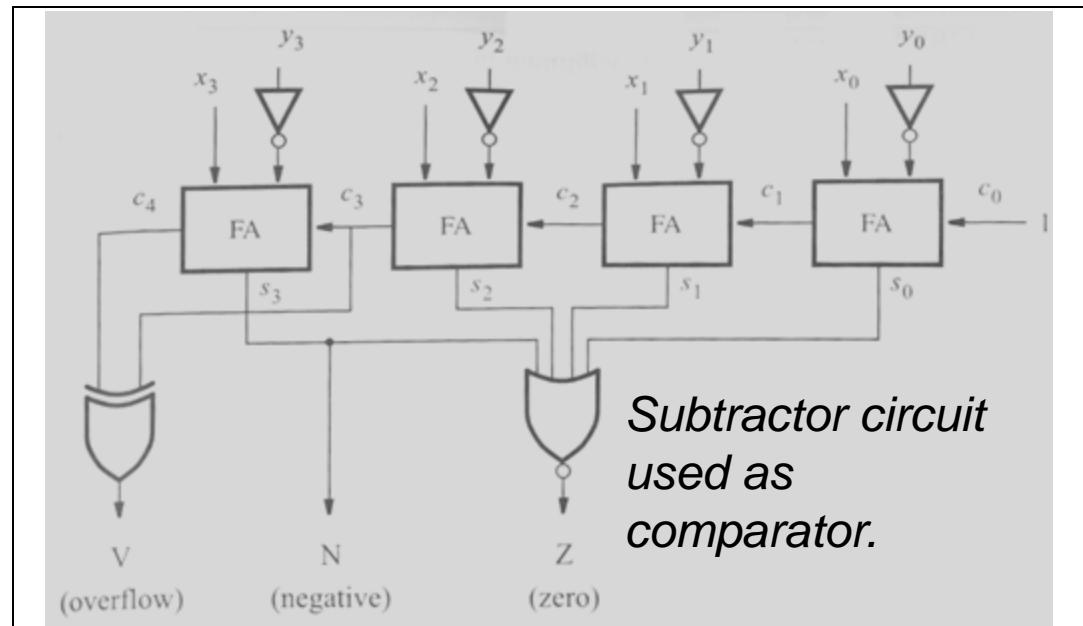
BV ex 5.10, < > =

Flags, Comparator. Two four-bit signed numbers, $X = x_3x_2x_1x_0$ and $Y = y_3y_2y_1y_0$, can be compared by using a subtractor circuit, which performs the operation $X - Y$. The three Flag-outputs denote the following:

- $Z = 1$ if the result is 0; otherwise $Z = 0$
- $N = 1$ if the result is negative; otherwise $N = 0$
- $V = 1$ if arithmetic overflow occurs; otherwise $V = 0$

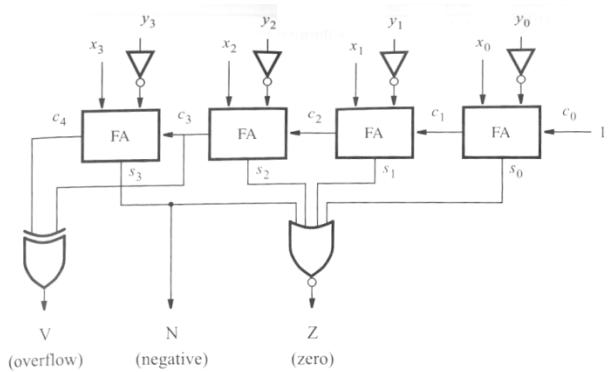
Show how Z , N , and V can be used to determine the cases

$X = Y$, $X < Y$, $X > Y$.



BV ex 5.10

X = Y ?

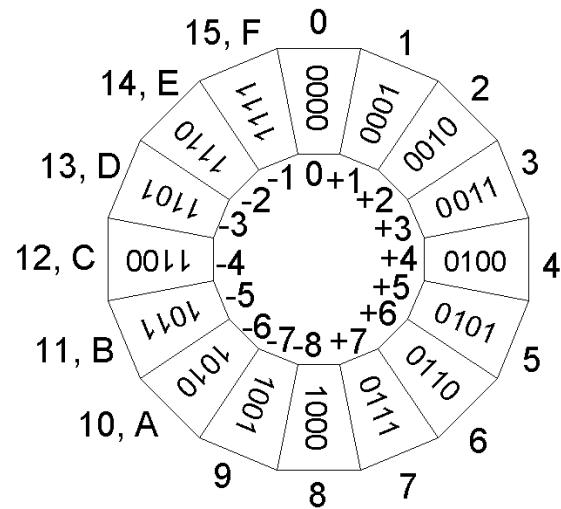


X = Y ?

$X - Y$

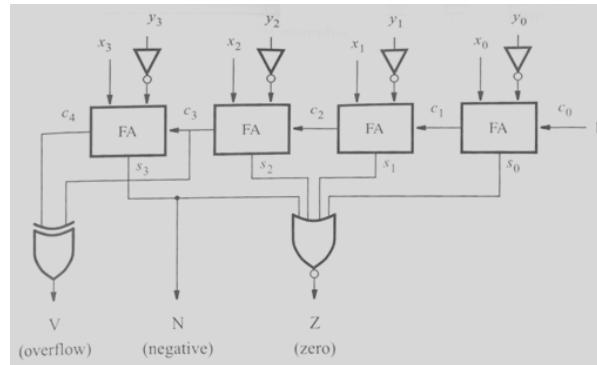
$$V = c_4 \oplus c_3 \quad N = s_3$$

$$Z = \overline{(s_3 + s_2 + s_1 + s_0)}$$



BV ex 5.10

$X = Y ?$



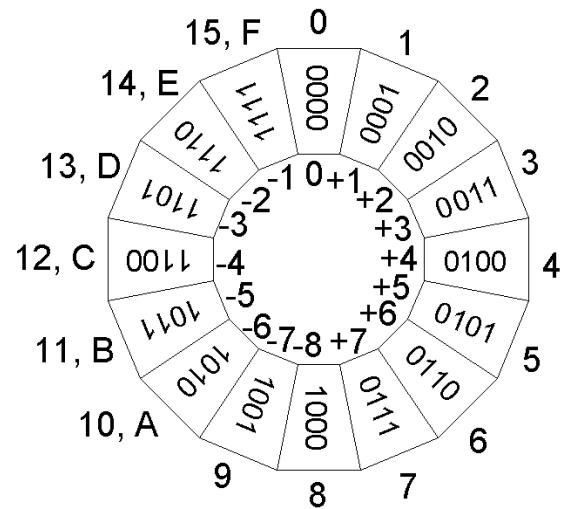
$X = Y ?$

$$X = Y \Rightarrow Z = 1$$

$X - Y$

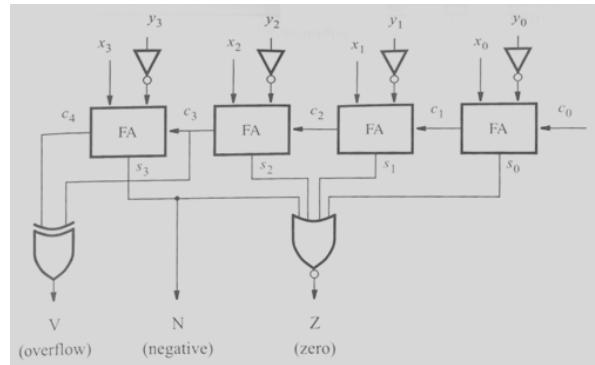
$$V = c_4 \oplus c_3 \quad N = s_3$$

$$Z = \overline{(s_3 + s_2 + s_1 + s_0)}$$



BV ex 5.10

$X < Y ?$



$X - Y$

$$V = c_4 \oplus c_3 \quad N = s_3$$

$$Z = \overline{(s_3 + s_2 + s_1 + s_0)}$$

Some test numbers:

$X < Y$

$$3 \quad 4$$

$$-4 \quad -3$$

$$-3 \quad 4$$

$$-5 \quad 4$$

$X - Y$

$$3 - 4 = -1$$

$$-4 - -3 = -1$$

$$-3 - 4 = -7$$

$$-5 - 4 = +7$$

$V \quad N$

$$0 \quad 1$$

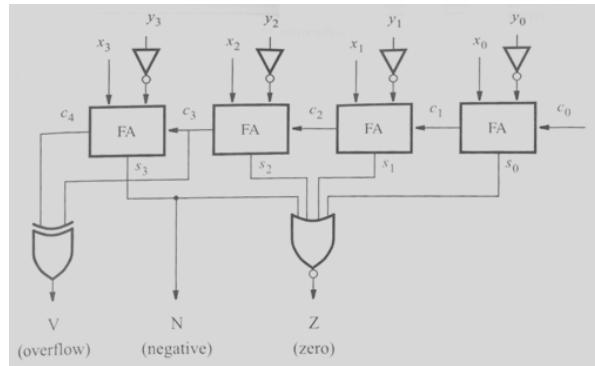
$$0 \quad 1$$

$$0 \quad 1$$

$$1 \quad 0$$

BV ex 5.10

X < Y ?



$X - Y$

$$V = c_4 \oplus c_3 \quad N = s_3$$

$$Z = \overline{(s_3 + s_2 + s_1 + s_0)}$$

If X and Y has the same sign $X - Y$ will always be correct and the flag $V = 0$. X, Y positive eg. $3 - 4$ $N = 1$. X, Y negative eg. $-4 - (-3)$ $N = 1$.

If X neg and Y pos and $X - Y$ has the correct sign, $V = 0$ and $N = 1$.

Tex. $-3 - 4$.

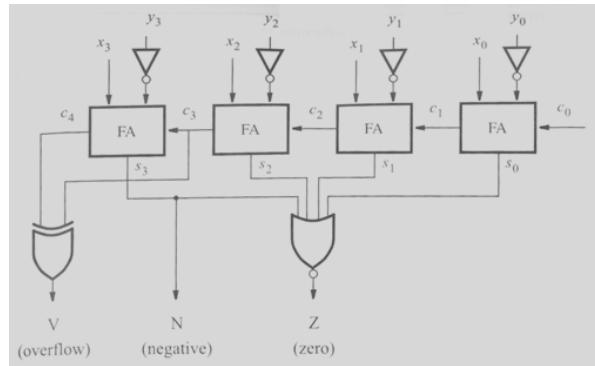
If X neg and Y but $X - Y$ gets the wrong sign, $V = 1$.

Then $N = 0$. Ex. $-5 - 4$.

- Summary: when $X < Y$ the flags V and N is always different. This could be indicated by a XOR gate.

BV ex 5.10

$X < Y ?$



$X - Y$

$$V = c_4 \oplus c_3 \quad N = s_3$$

$$Z = \overline{(s_3 + s_2 + s_1 + s_0)}$$

If X and Y has the same sign $X - Y$ will always be correct and the flag $V = 0$. X, Y positive eg. $3 - 4$ $N = 1$. X, Y negative eg. $-4 - (-3)$ $N = 1$.

If X neg and Y pos and $X - Y$ has the correct sign, $V = 0$ and $N = 1$.

Tex. $-3 - 4$.

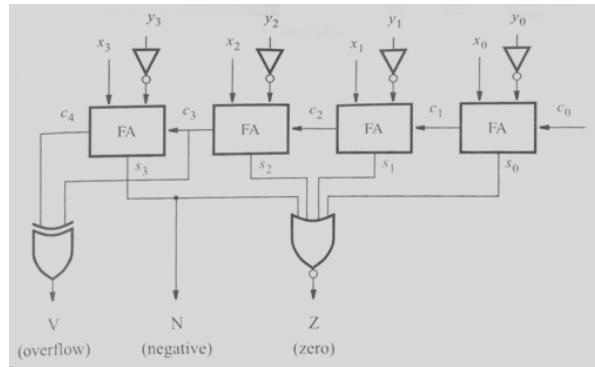
If X neg and Y but $X - Y$ gets the wrong sign, $V = 1$.

Then $N = 0$. Ex. $-5 - 4$.

- Summary: when $X < Y$ the flags V and N is always different. This could be indicated by a XOR gate.

$$X < Y \Rightarrow N \oplus V$$

BV ex 5.10



$$X - Y$$

$$V = c_4 \oplus c_3 \quad N = s_3$$

$$Z = \overline{(s_3 + s_2 + s_1 + s_0)}$$

$$X = Y \Rightarrow Z = 1$$

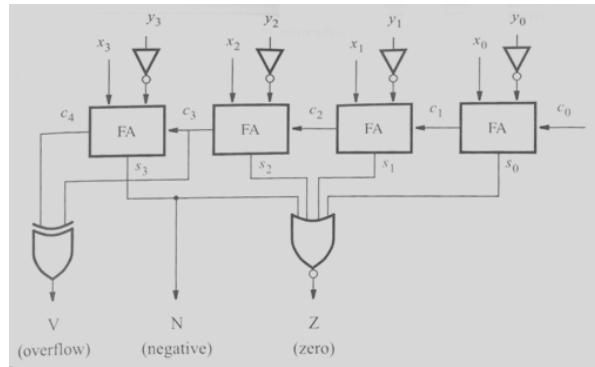
$$X < Y \Rightarrow N \oplus V$$

$$X \leq Y \Rightarrow$$

$$X > Y \Rightarrow$$

$$X \geq Y \Rightarrow$$

BV ex 5.10



$$X - Y$$

$$V = c_4 \oplus c_3 \quad N = s_3$$

$$Z = \overline{(s_3 + s_2 + s_1 + s_0)}$$

$$X = Y \Rightarrow Z = 1$$

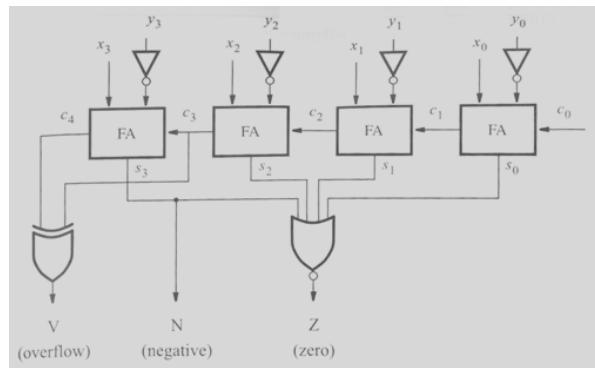
$$X < Y \Rightarrow N \oplus V$$

$$X \leq Y \Rightarrow Z + N \oplus V$$

$$X > Y \Rightarrow \overline{Z + N \oplus V} = \overline{Z} \cdot \overline{(N \oplus V)}$$

$$X \geq Y \Rightarrow \overline{N \oplus V}$$

BV ex 5.10



$$X - Y$$

$$V = c_4 \oplus c_3 \quad N = s_3$$

$$Z = \overline{(s_3 + s_2 + s_1 + s_0)}$$

$$X = Y \Rightarrow Z = 1$$

$$X < Y \Rightarrow N \oplus V$$

$$X \leq Y \Rightarrow Z + N \oplus V$$

$$X > Y \Rightarrow \overline{Z + N \oplus V} = \overline{Z} \cdot \overline{(N \oplus V)}$$

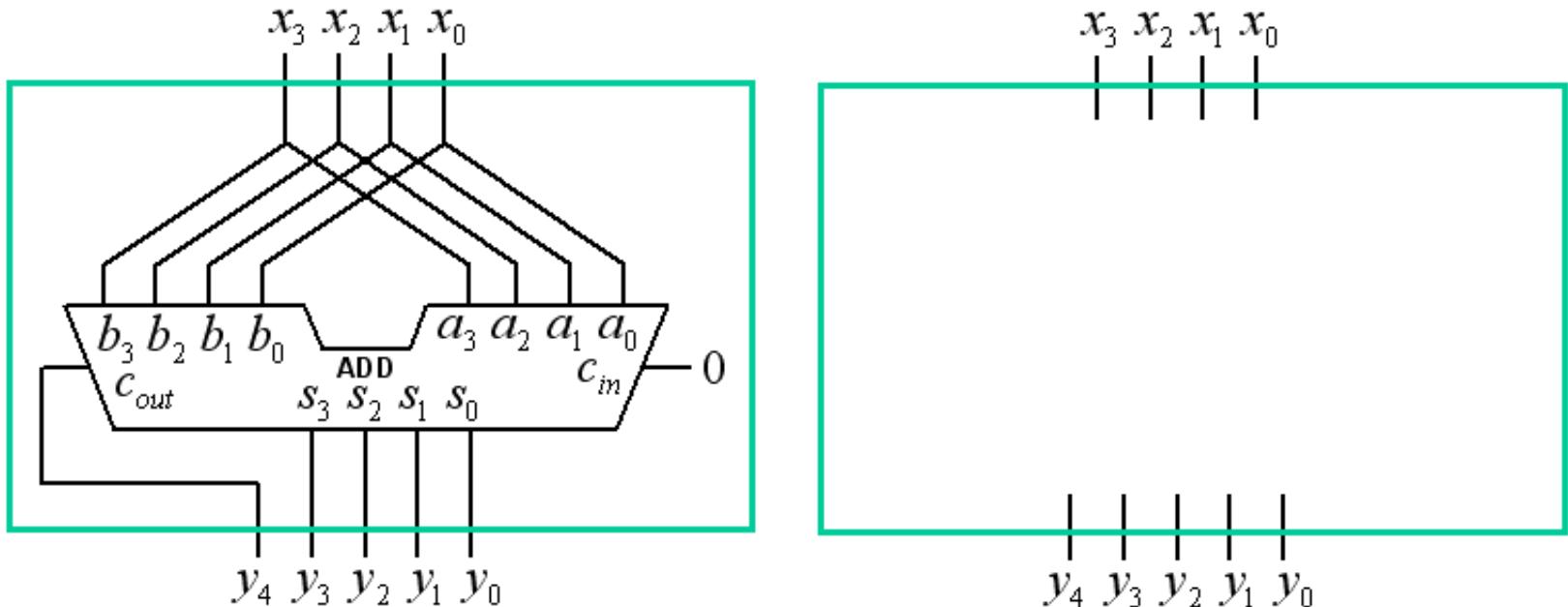
$$X \geq Y \Rightarrow \overline{N \oplus V}$$

This is how a computer can perform the most common comparisions

...

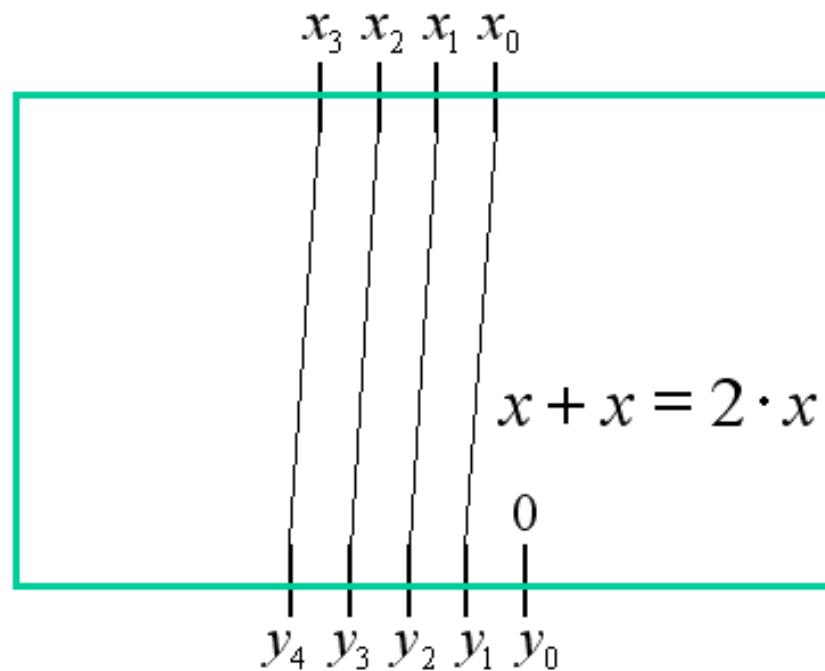
William Sandqvist [william@kth.se](mailto:wiliam@kth.se)

(Ex 8.12) Adder circuit

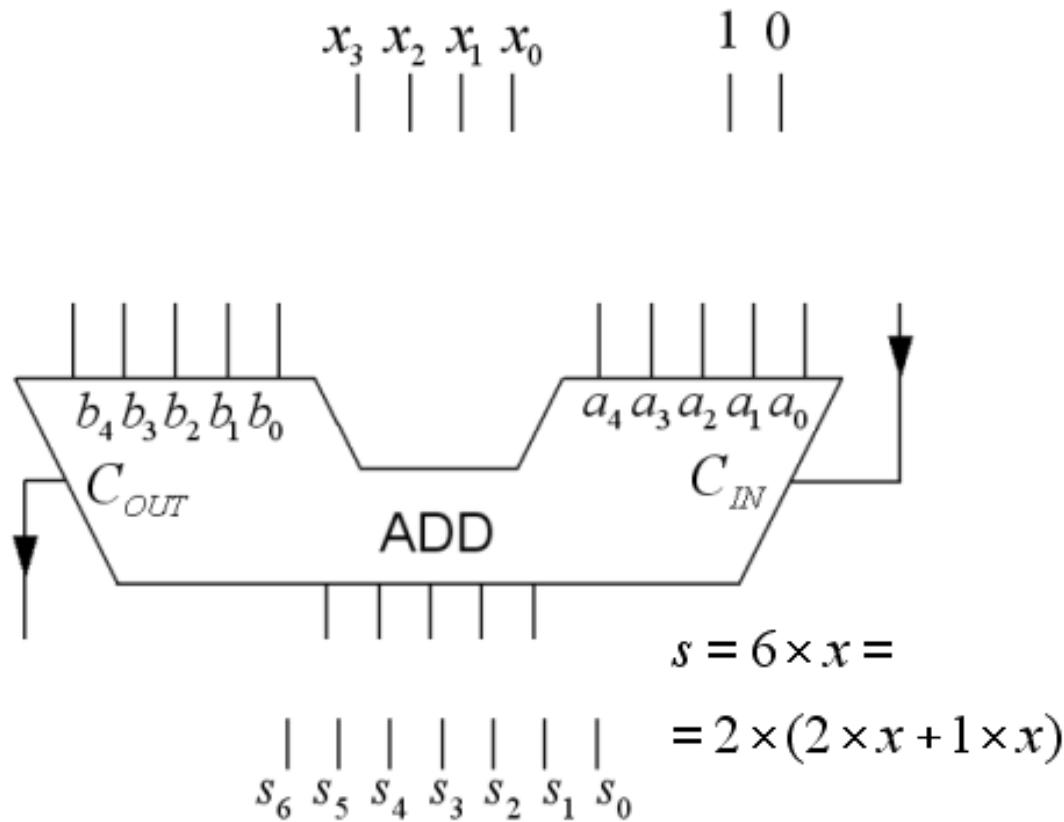


A four bit unsigned integer x ($x_3x_2x_1x_0$) is connected to an 4-bit adder as in the figure. The result is a 5-bit number y ($y_4y_3y_2y_1y_0$). Draw the figure to the right how the same results can be obtained *without using the adder*. There are also bits with the values 0 and 1 if needed.

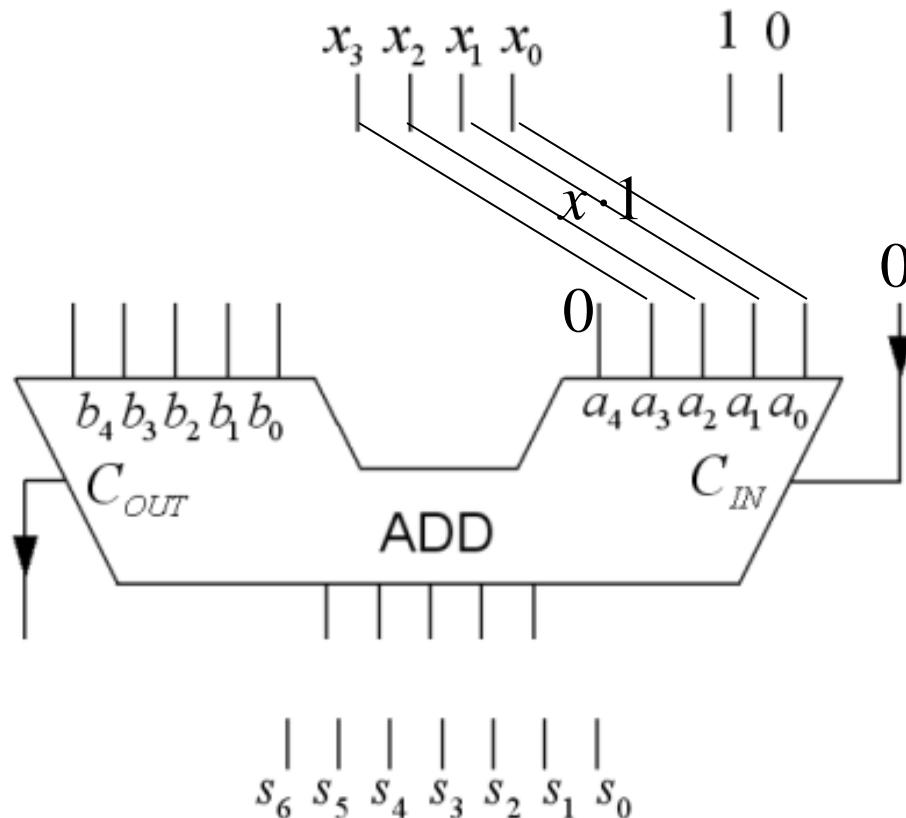
(Ex 8.12) Adder circuit



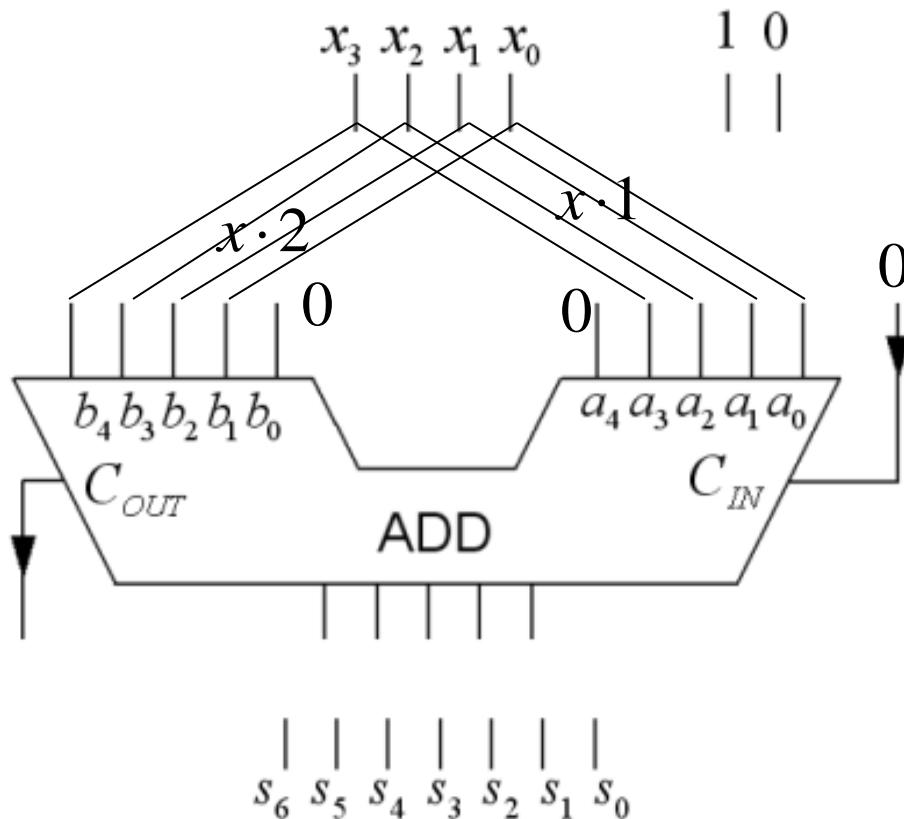
Ex 8.11 Multiply with 6 ?



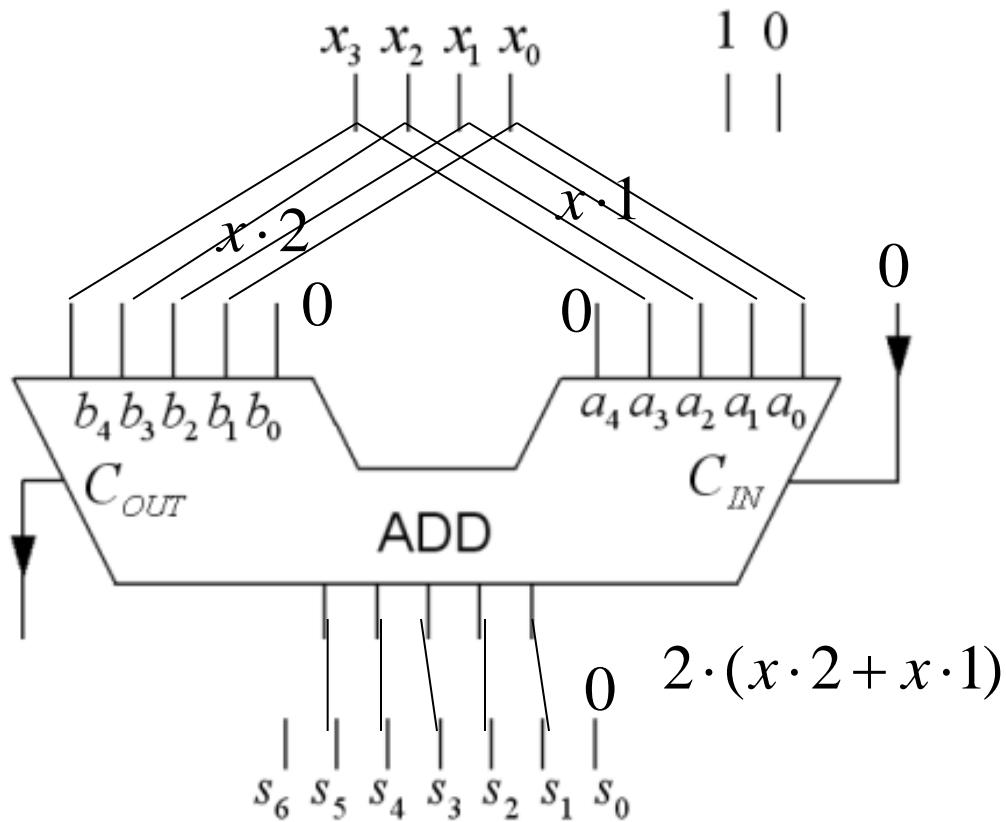
Ex 8.11 Multiply with 6 !



Ex 8.11 Multiply with 6 !



Ex 8.11 Multiply with 6 !

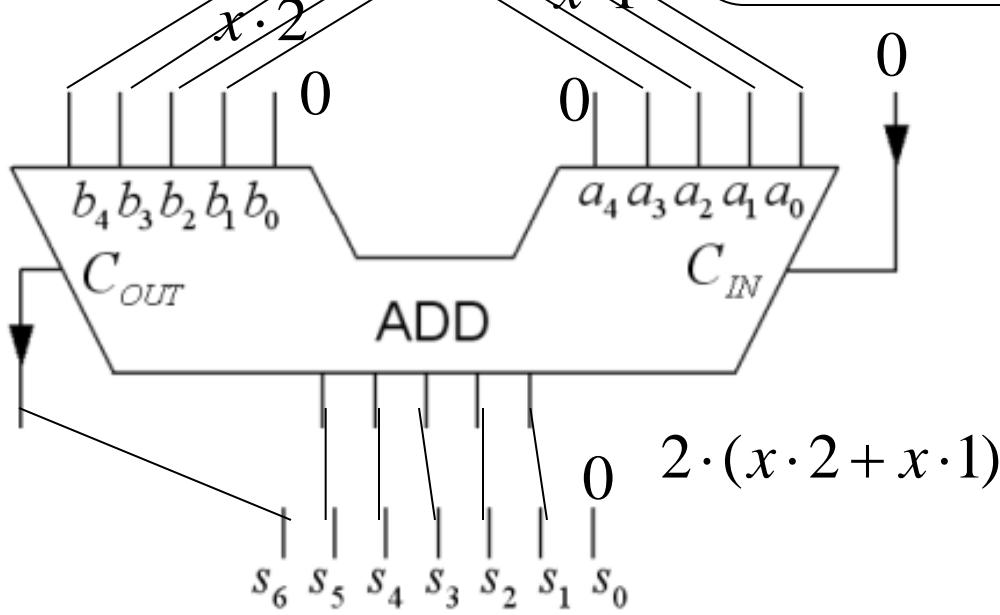


Ex 8.11 Multiply with 6 !

$$15 \cdot 6 = 90$$

$$1111 = 15$$

$$\begin{array}{r}
 \begin{array}{r}
 \begin{array}{r}
 1 & 1 & 1 & 1 & 0 & 0 \\
 0 & 1 & 1 & 1 & 1 & 0 \\
 + & 0 & 0 & 1 & 1 & 1 \\
 \hline
 1 & 0 & 1 & 1 & 0 & 1
 \end{array}
 & 15 \times 2 \\
 \hline
 1 & 0 & 1 & 1 & 0 & 1 & \times 2
 \end{array}$$



$$1011010 = 90$$

William Sandqvist william@kth.se