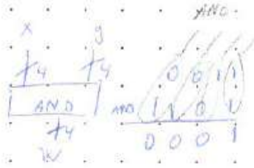
Blocs Aritmètics per N

Altres

AND bit a bit

Hi ha tantes portes AND com bits d'entrada. (n=4) doncs

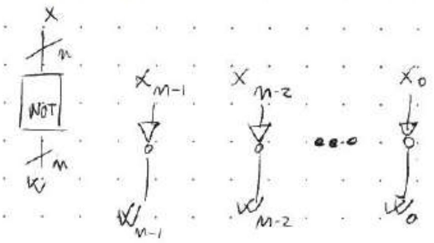


OR bit a bit

Tantes portes OR com bits té entrada i sortida.

NOT bit a bit

Tantes portes NOT com bits entrada.

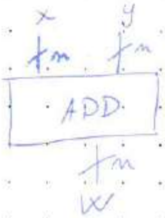


Sumador

$X_u = \sum_{i=0}^{n-1} x_i \cdot 2^i$ $\quad W_u = X_u + Y_u$ Construim un CLC que sumi 2 bit + carry

$Y_u = \sum_{i=0}^{n-1} y_i \cdot 2^i$

$W_u = \sum_{i=0}^{n-1} w_i \cdot 2^i$

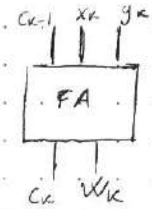


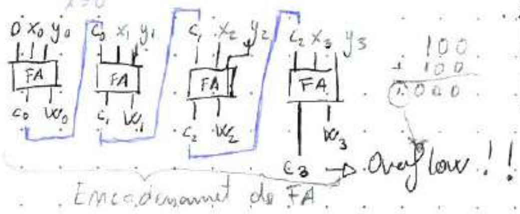
Del cas anterior

#El carry és el "m'emporto..."

C_{k-1}	x_k	y_k	C_k	W_k
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

FULL ADDER
Cultura General





$C_k = C_{out} = C_{k-1} \oplus x_k \oplus y_k$

$W_k = Sum = (x_k \cdot y_k) + (C_{k-1} \cdot (A \oplus B))$

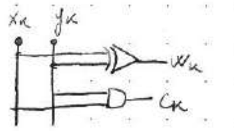
Half Adder

Només treballa bit a bit (Vist en apartat Altres) i no té carry de sumes passades, però sí que genera carry.

x_k	y_k	W_k	C_k
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

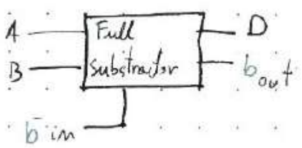
$W_k = x_k \oplus y_k$ XOR

$C_k = x_k \cdot y_k$ AND



Restador

Treballa bit a bit. Té les dues entrades + Bin (IN), i genera Resta + Bout (OUT). B = Borrow, D = Difference



A	B	bin	D	bout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

$$\begin{array}{r} 1\,1\,1\,0 \\ -\,0\,0\,1\,1 \\ \hline 1\,0\,1\,1 \end{array}$$

① 0-1 ≠ -1 doncs fiquem 1
② La que ens portem la sumem avall 1-2 = 0-1 ≠ -1
③ Repetir.

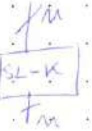
$D = A \oplus B \oplus b_{in}$

$b_{out} = (\bar{A}B) + (B_{in}(\overline{A \oplus B}))$

Desplaçador de bits

\# Shift Left

Desplaça k bits cap a l'esquerra



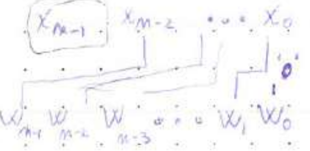
Fica "0" a les noves possicions. #Es com multiplicar per 10 en Decimal (que afegim '0')

$k=2$, $n=4$ → 0010 → 10?? → 1000

Aquest es perd

$2 \cdot 2^2 = 2^3 = 8$

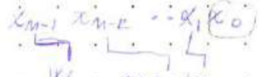
Pot ser que hi hagin bits que es perden.
\# Hi haurà overflow quan els perduts son '1'

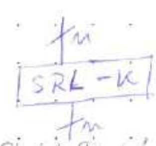


Multiplica per 2^k

En els pitjors casos, és necessita n+1 bits per representar resultat.
Si $X_{n-1} = 1$ → NO REPRESENTABLE

Desplaçador Drcta





\# Shift Right Logic

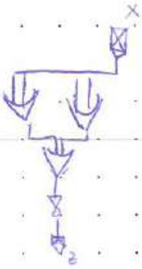
Funciona igual que Esquerra, però mou dreta

0110 (≫2, Afegim 0) = 0001 NO pot haver overflow **Divideix per 2^k**

Comparadors

Equivalent (Z)

Construïm un bloc que diu si $x == 0$ → $w = 1$
$x \,!= 0$ → $w = 0$



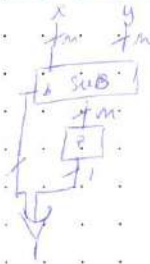
Si algun dels bits és 1, significa que resultat no serà 0, però volem que sigui 0, llavors fem NOT.

Less-than (LTU) # Less than Unsigned

Fem una resta de x-y i si y>x, llavors activarà el borrow i podem dir si és més petit.

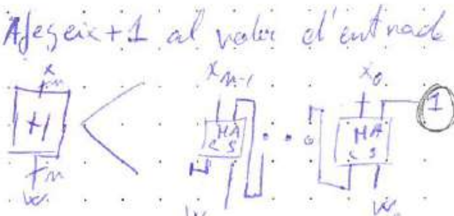
Menor o igual (LEU) # less equal Unsigned

Afegir les dues combinacions
(x≤y)



Incrementador

Afegeix +1 al valor d'entrada. Representa el 8 bits (pot no representar correctament.



→ Realment quan sumen, estem sumant a l'últim dígit. Per això li entra '1' només al últim.

1 0 1 0 0
\+ 1
→ Li sumem aquest, a l'últim, no a tots i no al primer. (Mateixa explicació que per canvi de signe en CA_2)

(4.1).

a) 10011111 + 01101111

```
 10011111
+01101111
----------
100001110   NR
```

b) 10101011 + 01010101

```
 10101011
+01010101
----------
100000000   NR
```

c) 01011101 + 01110111

```
 01011101
+01110111
----------
 10101100   R
```

(4.2).

a) 10101101 − 01011101

```
 10101101
-01011101
----------
 01010000   R
```

b) 10100000 − 10000001

```
 10100000
-10000001
----------
 00011111   R
```

c) 10100011 − 10111111 ✗

```
 10100011
-10111111
----------
                NR
```

(4.3).

a) 00010110 per 2^4

01100000 NR

b) 00101010 per 2^3

01010000 NR

c) 00000111 per 2^5

11100000 R

(4.4). Afegeixes tants '0' a l'esquerra com numerador.

a) 00000111 entre 2^1

00000011 NR

b) 00110101 entre 2^3

00011010 NR

c) 00010001 entre 2^0

00010001 R

d) 00101111 entre 2^7

00000000 NR

(4.5).

Nom

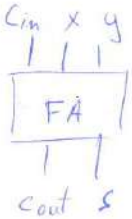
Dibuix

Taula Veritat

Half Adder

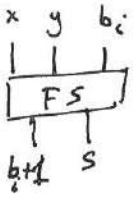
x	y	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Full Adder



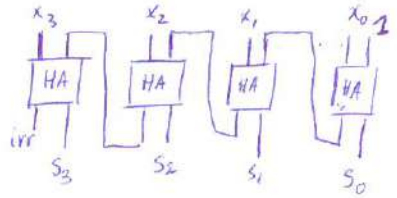
x	y	cin	Cout	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Full Substractor

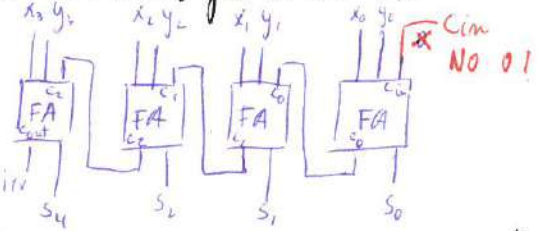


x	y	b_i	b_i+1	S
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

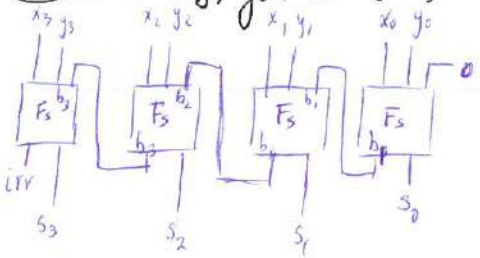
(4.6). Dibuixe el Bloc INC(x)



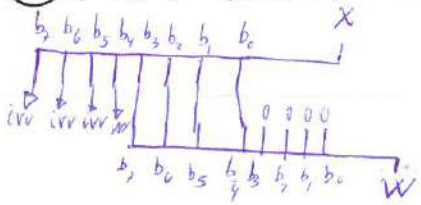
(4.7). ADD(x,y) fent ús de FA



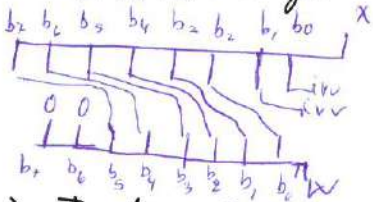
(4.8). SUB(x,y) fent ús (Fs)



(4.9). SL-4 (8 bits). $W_0 = X_0 \cdot 2^4$

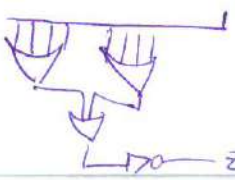


(4.10). SRL-2 (8 bits). Pregunta: [...]



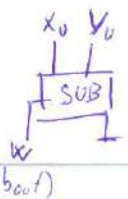
No pq. La divisió entre 2 d'un numero sempre serà més petit (el resultat representable) que el número a dividir. ▽IMPO

(4.12). Z(x). Indica si $X_0 = 0$ o no.
Que valdrà Z(x) si x = 1111 1111



Z(x) = 0
Pq no tots els num. són 0.

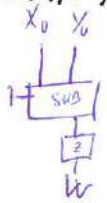
(4.13). Comparador LTU(x,y) ($X_0 < Y_0$)



x = 1000 1000
LTU(x,y) y = 0111 1111

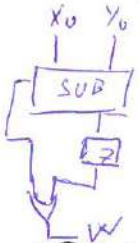
LTU = 0

(4.15). EQ(x,y) ($X_0 = Y_0$)
EQ(X,Y) = ? x = 10101010
y = 10101001



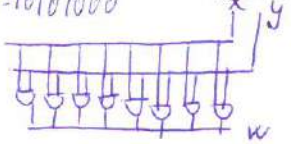
EQ = 0

(4.14). LEU(x,Y) ($X_0 \leq Y_0$)
LEU(x,y) = ? x = 10001000
Y = 1000 1001

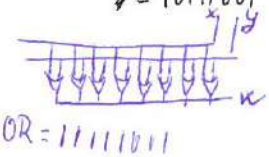


LEU = 1

(4.16). AND(x,y)
AND(x,x) = ? x = 11101010
y = 10111001
AND = 10101000

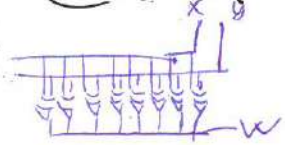


(4.17). OR(x,y)
OR = ?? x = 11101010
y = 10111001



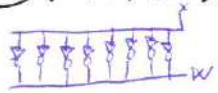
OR = 11111011

(4.18). XOR



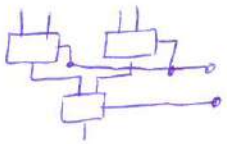
XOR = 01010011

(4.19). NOT(x)

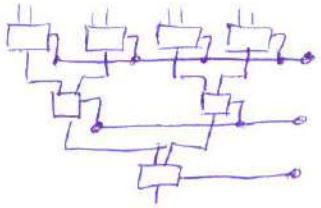


NOT = 00010101

(4.20). M_K-4-1



M_X-8-1



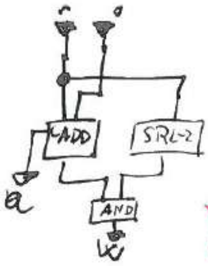
(4.21). Sort de a i w

$x = 00110111$

$y = 11000001$

$a = 0$

$W = 00001000$




```
 00110111        00001101
+11000001
0|11111000
 00001101  AND
 ??00?000
```

CUIDADO AMB el n bits sortida

$W = 8 \quad a = 1$

(4.22)