Introducció als Computadors

Tema 5: Nombres enters

http://personals.ac.upc.edu/enricm/Docencia/IC/IC5.pdf

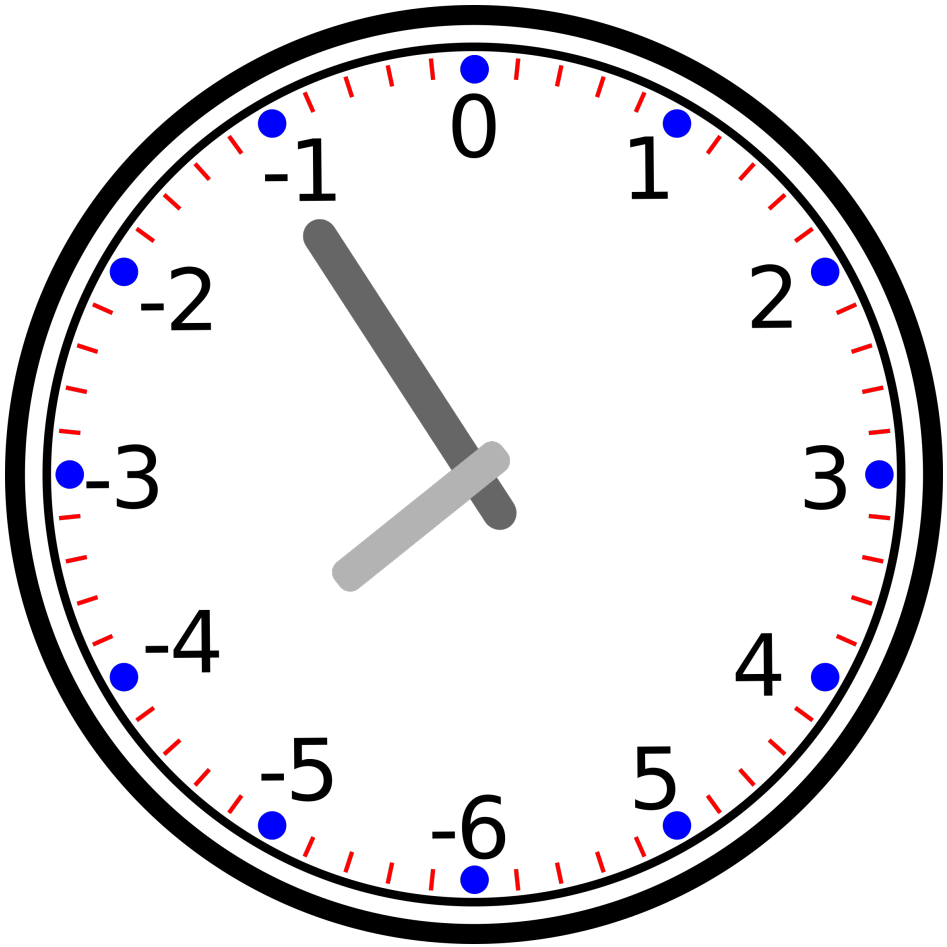
Enric Morancho
(enricm@ac.upc.edu)

Departament d'Arquitectura de Computadors
Facultat d'Informàtica de Barcelona
Universitat Politècnica de Catalunya

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

2020-21, 1^{er} quad.

Aritmètica modular amb nombres negatius



[1]

Índex

Introducció

- Veurem com representar i operar amb els nombres enters
  - $\mathbb{Z} = \{-\infty, \ldots, -2, -1, 0, 1, 2, \ldots, +\infty\}$
  - *Signed integers*
- Com al cas dels naturals, dedicarem n bits per a representar els enters
  - Només podrem representar un subconjunt finit de $\mathbb{Z}$
    - El subconjunt tindrà 2^n elements
    - A IC, com n=16, des de -32.768 fins a 32.767
- Els blocs aritmètics d'enters també implementen aritmètica modular
  - $(4 + 7) \mod 8 = (4 + (-1)) \mod 8 = 3$
  - Molts blocs tindran una sortida per indicar si el resultat de l'aritmètica modular coincideix amb el de l'aritmètica convencional

Repàs representació convencional $\mathbb{N}$

- Vector de n dígits binaris (bits)
  - $X = x_{n-1}x_{n-2}\ldots x_1x_0, \forall i\ x_i \in \{0,1\}$
- Valor decimal equivalent:
  - $X_u = \sum_{i=0}^{n-1} x_i \cdot 2^i$
- Rang de valors representable:
  - $0 \leq X_u \leq 2^n - 1$
- Extensió de rang:
  - Afegir 0's per l'esquerra
- Bloc aritmètic ADD implementa la suma amb aritmètica modular 2^n
  - $W = ADD(X, Y)$
    - $W_u = (X_u + Y_u) \mod 2^n$
    - Retorna els n bits de menys pes de la suma convencional
    - W no inclou el bit de més pes w_n de la suma convencional
  - La sortida c (*carry*) indica que el resultat de la suma convencional no és representable amb n bits

Índex

Índex

Signe i magnitud

- La codificació seria anàloga a la que fem en decimal:
  - El bit de més pes representa el signe (0 positiu, 1 negatiu)
  - La resta de bits representa el valor absolut del nombre (magnitud)
- Donat un vector de bits X , X_{sm} és el valor enter en decimal de X interpretat com a enter codificat en signe i magnitud
  - $X_{sm} = (-1)^{x_{n-1}} \sum_{i=0}^{n-2} x_i \cdot 2^i$
- Exemple per a $n = 3$

x_2	x_1	x_0	X_u	X_{sm}
0	0	0	0	0
0	0	1	1	1
0	1	0	2	2
0	1	1	3	3
1	0	0	4	0
1	0	1	5	-1
1	1	0	6	-2
1	1	1	7	-3

Doble representació del nombre 0

- **No la utilitzarem a IC**
  - Algorisme de suma més complexe que per als naturals
  - A EC s'utilitza per representar nombres en coma flotant

Índex

- **Buscarem una representació tal que la suma d'enters es pugui fer amb el mateix bloc ADD que realitza la suma de naturals**
  - El zero i els enters positius els representarem com al cas dels naturals
  - Representarem cada enter negatiu amb un vector de bits tal que al sumar-lo modularment amb la representació del propi enter en valor absolut el resultat sigui zero
    - Per a $n = 3$, com l'enter 1 es codifica amb $X = 001 \implies$ codificarem -1 amb un vector Y tal que $ADD(X, Y) = 000 \implies Y = 111$
    - Com l'enter 2 es codifica amb $X = 010 \implies$ codificarem -2 amb un vector Y tal que $ADD(X, Y) = 000 \implies Y = 110$
    - ...

Ca2: tres possibilitats

- Exemple de tres possibles representacions seguint la idea ($n = 3$)

x_2	x_1	x_0	X_u	X_a	X_b	X_c
0	0	0	0	0	0	0
0	0	1	1	1	1	1
0	1	0	2	2	2	2
0	1	1	3	3	-5	3
1	0	0	4	4	-4	-4
1	0	1	5	-3	-3	-3
1	1	0	6	-2	-2	-2
1	1	1	7	-1	-1	-1

  - En cap cas, l'interval d'enters representat és simètric
    - No podrà ser perfectament simètrica perquè 2^n és parell i, un cop representat el 0, queda un nombre senar de nombres a representar
  - A la interpretació X_c, el bit x_2 ens indica el signe del nombre
    - Formalitzarem la generalització d'aquesta representació a n bits

- X_s és el valor enter, en decimal, representat pel vector de bits X
  - $X_s = -x_{n-1} \cdot 2^{n-1} + \sum_{i=0}^{n-2} x_i \cdot 2^i$
  - Similar a l'expressió del valor natural, llevat el bit de més pes
    - $X_u = \sum_{i=0}^{n-1} x_i \cdot 2^i$
- Rang d'enters representable:
  - $-2^{n-1} \leq X_s \leq 2^{n-1} - 1$
    - $n = 3 \implies -4 \leq X_s \leq 3$
    - $n = 4 \implies -8 \leq X_s \leq 7$
    - $n = 16 \implies -32.768 \leq X_s \leq 32.767$
- $\forall X, Y \; X_s = Y_s + 1 \implies X_u = (Y_u + 1) \mod 2^n$

Ca2: exemple $n = 3$

x_2	x_1	x_0	X_u	X_s
0	0	0	0	$0 = 0 + 0$
0	0	1	1	$1 = 0 + 1$
0	1	0	2	$2 = 0 + 2$
0	1	1	3	$3 = 0 + 3$
1	0	0	4	$-4 = -4 + 0$
1	0	1	5	$-3 = -4 + 1$
1	1	0	6	$-2 = -4 + 2$
1	1	1	7	$-1 = -4 + 3$

Ca2: exemple $n = 4$

x_3	x_2	x_1	x_0	X_u	X_s
0	0	0	0	0	$0 = 0 + 0$
0	0	0	1	1	$1 = 0 + 1$
0	0	1	0	2	$2 = 0 + 2$
0	0	1	1	3	$3 = 0 + 3$
0	1	0	0	4	$4 = 0 + 4$
0	1	0	1	5	$5 = 0 + 5$
0	1	1	0	6	$6 = 0 + 6$
0	1	1	1	7	$7 = 0 + 7$
1	0	0	0	8	$-8 = -8 + 0$
1	0	0	1	9	$-7 = -8 + 1$
1	0	1	0	10	$-6 = -8 + 2$
1	0	1	1	11	$-5 = -8 + 3$
1	1	0	0	12	$-4 = -8 + 4$
1	1	0	1	13	$-3 = -8 + 5$
1	1	1	0	14	$-2 = -8 + 6$
1	1	1	1	15	$-1 = -8 + 7$

Ca2: Extensió de rang (*Sign extension*, SE)

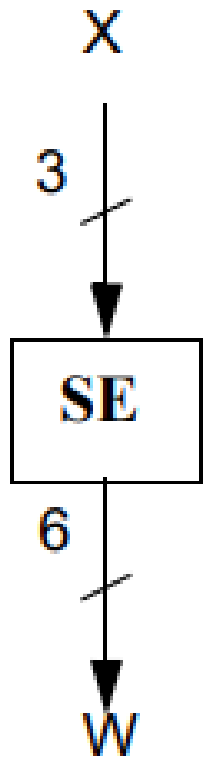
Donat X vector de n bits, trobar W de $n+1$ bits tal que $X_s = W_s$

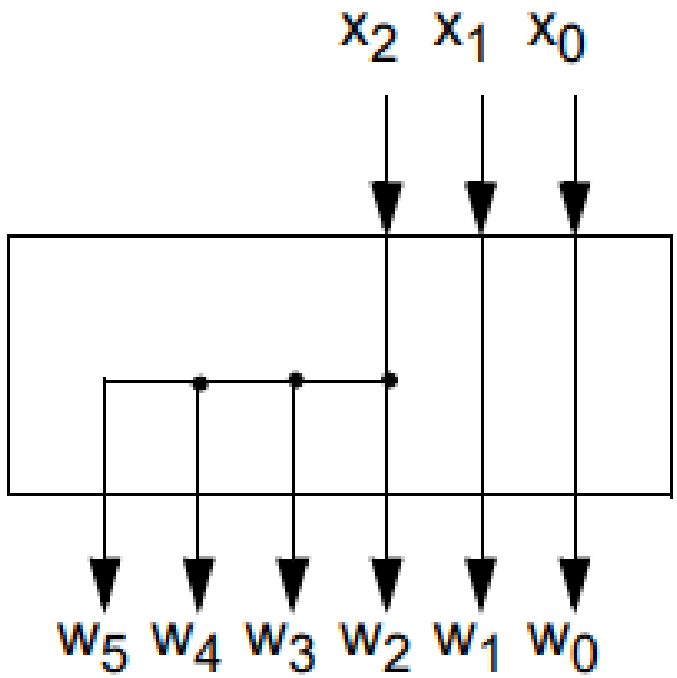
- Als naturals afegíem un zero per l'esquerra
- Als enters, això també valdrà pels positius
- I pels negatius?
  - Si afegim un "0" el convertiríem en un número positiu.
  - En aquest cas cal afegir un "1" per l'esquerra
- En resum, cal replicar el bit de més pes de X
  - *Sign extension*
- Exemple:
  - $X = 10001 \implies X_s = -16 + 1 = -15$
  - $Y = 110001 \implies Y_s = -32 + 16 + 1 = -15$
  - $Z = 11110001 \implies Z_s = -128 + 64 + 32 + 16 + 1 = -15$

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

- Implementació d'un bloc d'extensió de rang de 3 a 6 bits
  - Bloc implementat sense portes lògiques!

$W_s = X_s$





Conversió de Ca2 (negatiu) a decimal

- Aplicant la fórmula de X_s
  - $X_s = -x_{n-1} \cdot 2^{n-1} + \sum_{i=0}^{n-2} x_i \cdot 2^i$
    - $X = 11001010 \implies X_s = -128 + 64 + 8 + 2 = -128 + 74 = -54$
  - Podem simplificar-ho si substituïm tots els "1"'s consecutius a l'esquerra per un únic "1"
    - "Desfer" l'extensió de rang
    - $Y = 1001010 \implies Y_s = -64 + 8 + 2 = -54$
- Canviant el signe
  - Veurem que si $W = (!X + 1) \bmod 2^n \implies W_s = -X_s$
    - $!X$ és el vector de bits resultat d'aplicar la NOT a tots els bits de X
    - Si $X = 11001010 \implies W = 00110101 + 1 = 00110110$
      $\implies W_s = W_u = 32 + 16 + 4 + 2 = 54 \implies X_s = -54$

Conversió de decimal (negatiu) a Ca2

- Dividint entre 2:

```
-5 |2          -5   =>   1 0 1 1
-6 -3 |2
 1 -4 -2 |2
    1 -2 -1 |2
       0  0  0
          -1
```

```
5 |2          5   =>   0 1 0 1
4 2 |2
1 2 1 |2
  0 0 0 |2
    1 0 0
      0
```

- Determinant el nombre de bits necessaris per representar-lo (n)
  - Necessitem n tal que $-2^{n-1} \leq X_s \leq 2^{n-1} - 1$
    Si $X_s = -54 \implies -64 \leq -54 \leq 63 \implies n - 1 = 6 \implies n = 7$
  - Ara, dues possibilitats:
    - Obtenim els $n-1$ bits baixos de X resolent $X_s = -2^{n-1} + L_u$
      $-54 = -64 + L_u \implies L_u = 10 \implies L = 001010$
      Finalment, concatenem el dígit de signe "1" a L $\implies X = 1001010$
    - Si $X_s < 0 \implies X_u + |X_s| = 2^n$
      $X_u + 54 = 128 \implies X_u = 74 \implies X = 1001010$

Exercici

- Ompliu cada fila de la taula a partir dels valors indicats:
  - $X_s = -x_{n-1} \cdot 2^{n-1} + \sum_{i=0}^{n-2} x_i \cdot 2^i$
  - $X_u = \sum_{i=0}^{n-1} x_i \cdot 2^i$
  - Quan hagueu d'especificar n, indiqueu el valor mínim possible.

n	X	X_s	X_u
8	11111011		
		-26	
6			51
7			51
		-5	
5		-30	

Exercici

- Ompliu cada fila de la taula a partir dels valors indicats:
  - $X_s = -x_{n-1} \cdot 2^{n-1} + \sum_{i=0}^{n-2} x_i \cdot 2^i$
  - $X_u = \sum_{i=0}^{n-1} x_i \cdot 2^i$
  - Quan hagueu d'especificar n, indiqueu el valor mínim possible.

n	X	X_s	X_u
8	11111011	-5	251
6	100110	-26	38
6	110011	-13	51
7	0110011	51	51
4	1011	-5	11
5	impossible	-30	impossible

Índex

Exemples representabilitat suma a Ca2

- Utilitzarem el mateix sumador que per a naturals $W = ADD(X, Y)$
  - Funcionarà perquè $\forall X, Y \; X_s = Y_s + 1 \implies X_u = (Y_u + 1) \mod 2^n$
- Però la detecció de no representabilitat serà diferent
- Fem algunes proves ($n = 4 \implies 0 \leq X_u \leq 15, \quad -8 \leq X_s \leq 7$):

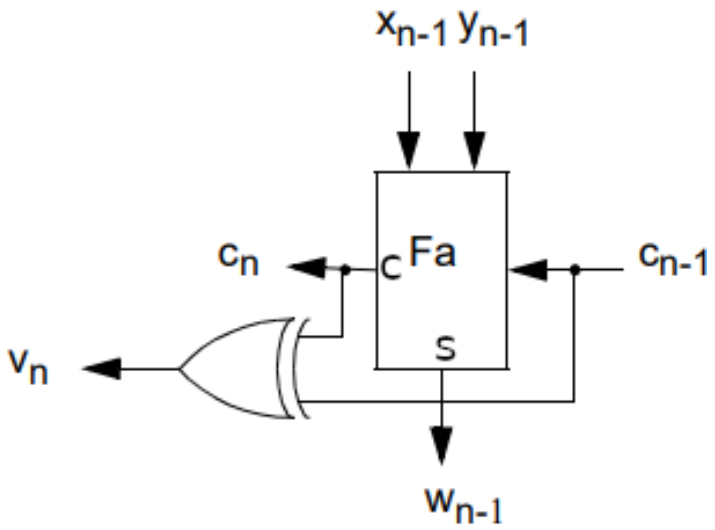
X	X_u	X_s	Y	Y_u	Y_s	$W = ADD(X, Y)$	W_u	W_s	$X_u + Y_u$	$X_s + Y_s$
1100	12	-4	0011	3	3	1111	15	-1	15	-1
0010	2	2	1111	15	-1	0001	1	1	17	1
1000	8	-8	1001	9	-7	0001	1	1	17	-15
0111	7	7	0010	2	2	1001	9	-7	9	9
0100	4	4	0011	3	3	0111	7	7	7	7
1100	12	-4	1111	15	-1	1011	11	-5	27	-5

- Quan no és representable el resultat de la suma d'enters?
  - Entrades de signe diferent: el resultat **sempre** serà representable
  - Entrades del mateix signe: haurem de comprovar que el signe del resultat coincideixi amb el signe de les entrades

No representativitat a la suma d'enters

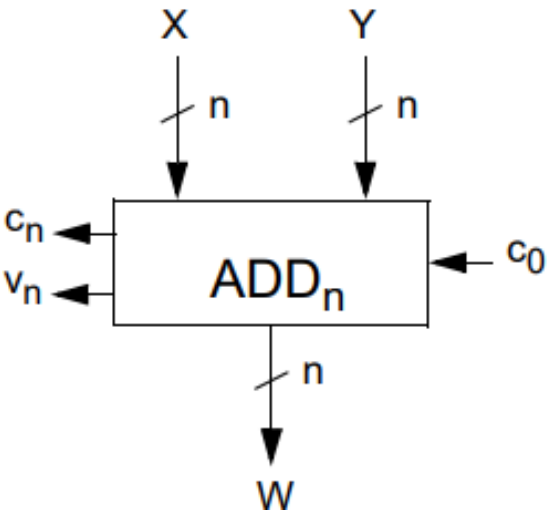
- Afegirem una nova sortida, v_n, al bloc ADD
  - $v_n = 1$ indicarà que el resultat de la resta d'enters amb aritmètica convencional no és representable en Ca2 de n bits
  - Cal detectar si els operands són del mateix signe però el resultat és de signe diferent
    - Els operands són positius però el resultat és negatiu: $!x_{n-1} \cdot !y_{n-1} \cdot w_{n-1}$
    - Els operands són negatius però el resultat és positiu: $x_{n-1} \cdot y_{n-1} \cdot !w_{n-1}$
  - $v_n = x_{n-1} \cdot y_{n-1} \cdot !w_{n-1} + !x_{n-1} \cdot !y_{n-1} \cdot w_{n-1}$
  - Amb la TV es pot demostrar que v_n és equivalent a c_n *XOR* c_{n-1}

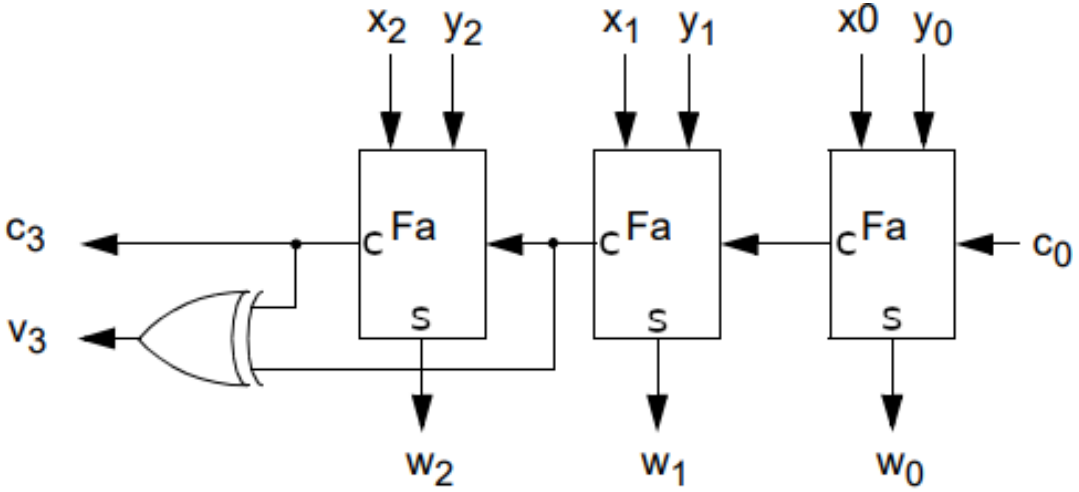
x_{n-1}	y_{n-1}	c_{n-1}	c_n	w_{n-1}	v_n
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	1	0
0	1	1	1	0	0
1	0	0	0	1	0
1	0	1	1	0	0
1	1	0	1	0	1
1	1	1	1	1	0



Sumador de naturals i d'enters

- Encapsulat i implementació per a $n = 3$
  - Afegim l'entrada c_0 perquè és serà útil més endavant
  - Per fer sumes, hi posarem el valor "0"





- A la documentació de l'assignatura teniu les expressions matemàtiques que indiquem els valors W_u i W_s en funció de X_u, Y_u, X_s i Y_s per a totes les casuístiques.

Exercici

- Ompliu els espais buits de la taula ($n = 4$), $W = ADD(X, Y)$

X	X_u	X_s	Y	Y_u	Y_s	W	W_u	W_s	c_n	v_n	$X_u + Y_u$	$X_s + Y_s$
0011			0110									
1001			1111									
1011					-4							
1000				2								

Exercici

- Ompliu els espais buits de la taula ($n = 4$), $W = ADD(X, Y)$

X	X_u	X_s	Y	Y_u	Y_s	W	W_u	W_s	c_n	v_n	$X_u + Y_u$	$X_s + Y_s$
0011	3	3	0110	6	6	1001	9	-7	0	1	9	9
1001	9	-7	1111	15	-1	1000	8	-8	1	0	24	-8
1011	11	-5	1100	12	-4	0111	7	7	1	1	23	-9
1000	8	-8	0010	2	2	1010	10	-6	0	0	10	-6

Índex

Canvi de signe

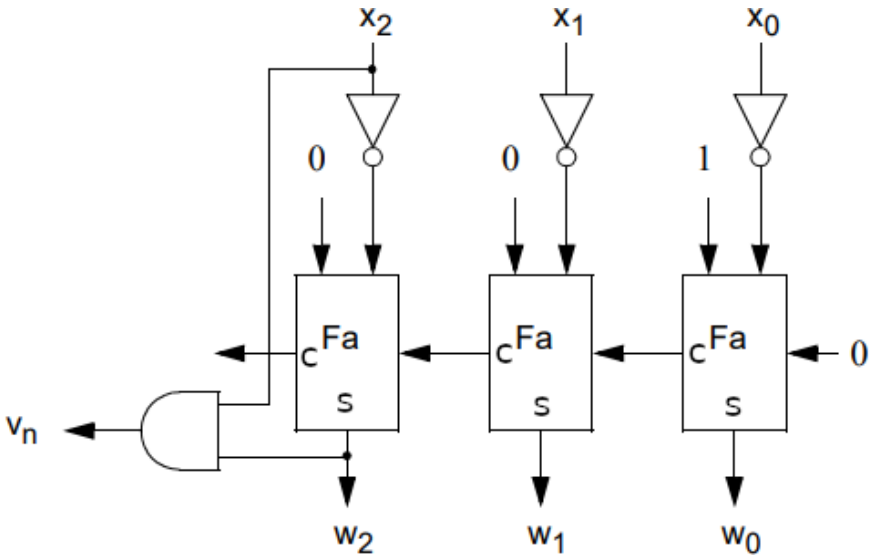
- Donat un vector X, trobar un vector W tal que $W_s = -X_s$
  - X i W vectors de n bits
- Considerem $W = (!X + 1) \bmod 2^n$
  - Estudiem alguns casos per a n=4

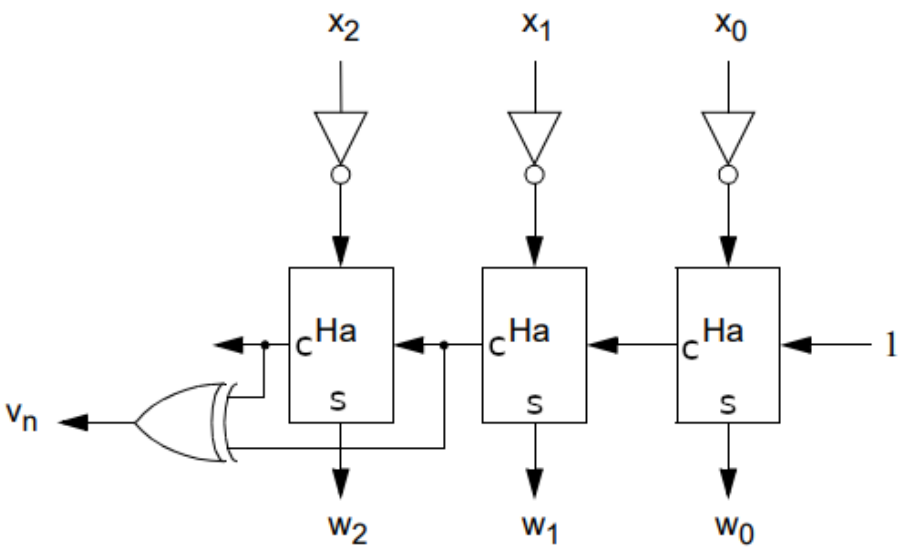
X	$!X$	$W = ADD(!X, 1)$	X_s	W_s
0001	1110	1111	1	-1
0011	1100	1101	3	-3
0111	1000	1001	7	-7
1100	0011	0100	-4	4
0000	1111	0000	0	0
1000	0111	1000	-8	-8

  - Veiem que es compleix sempre llevat $X = 1000$ ($X_s = -8$)
    - Era d'esperar perquè $-8 \leq X_S \leq 7$
    - Caldrà detectar no representativitat del resultat

Canvi de signe

- Detecció de no representativitat
  - Si l'entrada i la sortida tenen el signe negatiu
    - $v_n = x_{n-1} \cdot w_{n-1}$
  - A la documentació es demostra que és equivalent a:
    - $v_n = c_n \; XOR \; c_{n-1}$
    - Com la detecció de no representativitat a la suma
- Implementacions per a $n = 3$ amb n FA'a i amb n HA's

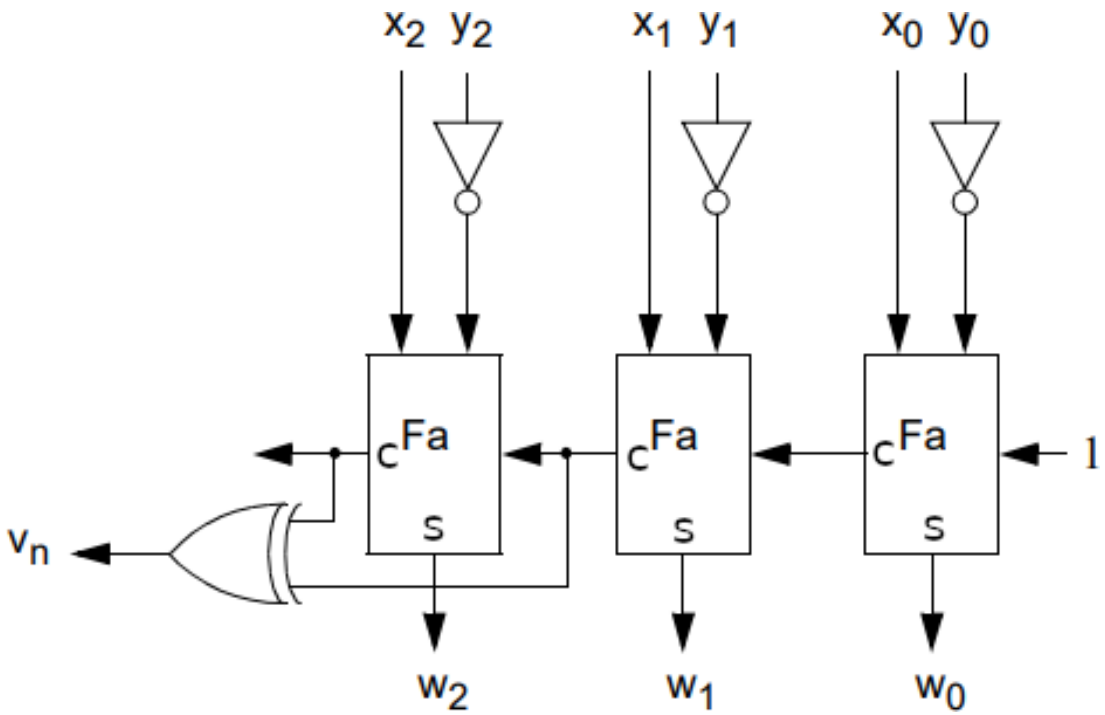




Índex

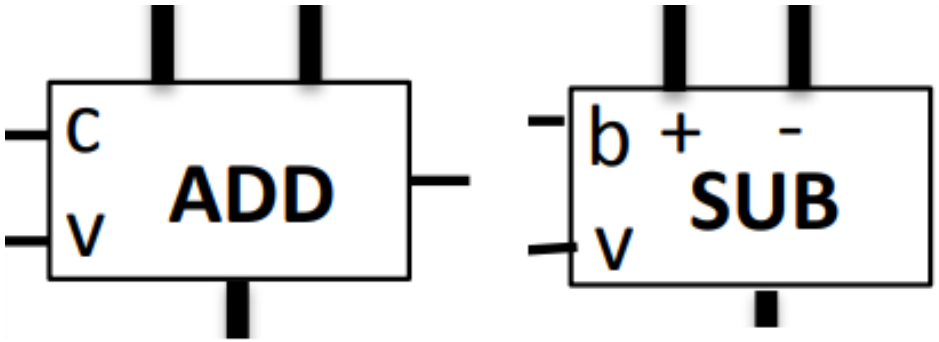
Resta

- Donats els vectors X, Y, trobar un vector W tal que $W_s = X_s - Y_s$
  - $W_s = X_s - Y_s = X_s + (-Y_s)$
    - Ja sabem canviar de signe
    - La detecció de no representabilitat ja la tenim feta
    - Alimentem amb "1" el primer *Full-adder*
- Implementació per a $n = 3$ amb n FA's



Blocs ADD i SUB que utilitzarem a IC

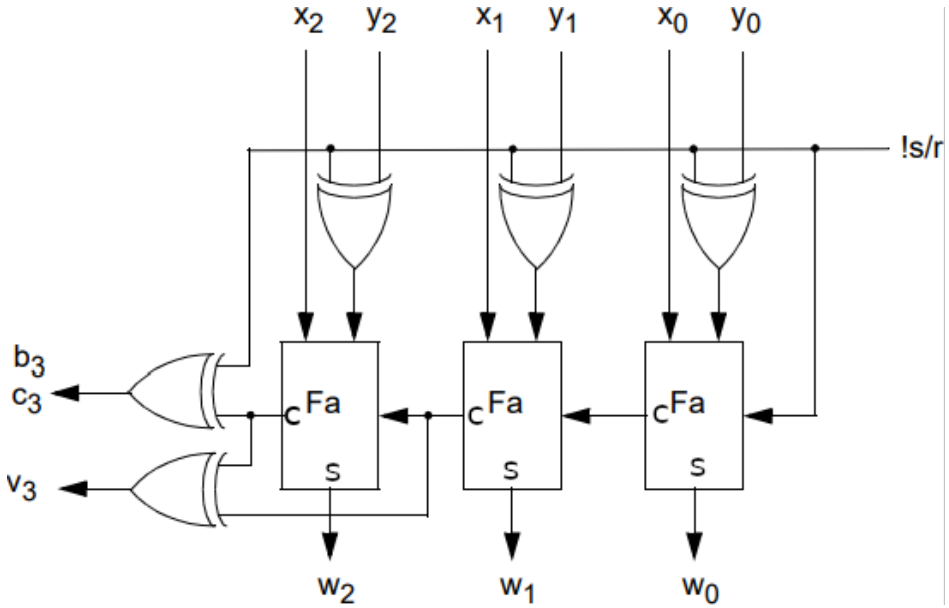
- ADD: Bloc sumador (per a enters i naturals)
  - Entrades: X, Y i el *carry-in* del *full-adder* dels bits de menys pes
    - Normalment, $carry\_in_0 = 0$
  - Sortides: $W = ADD(X, Y), c, v$
    - $c = 1 \iff$ resultat no representable amb interpretació naturals
    - $v = 1 \iff$ resultat no representable amb interpretació enters
- SUB: Bloc restador (per a enters i naturals)
  - Entrades: X, Y (+: minuend , -: subtrahend)
  - Sortides: $W = SUB(X, Y), b, v$
    - $b = 1 \iff$ resultat no representable amb interpretació naturals
    - $v = 1 \iff$ resultat no representable amb interpretació enters



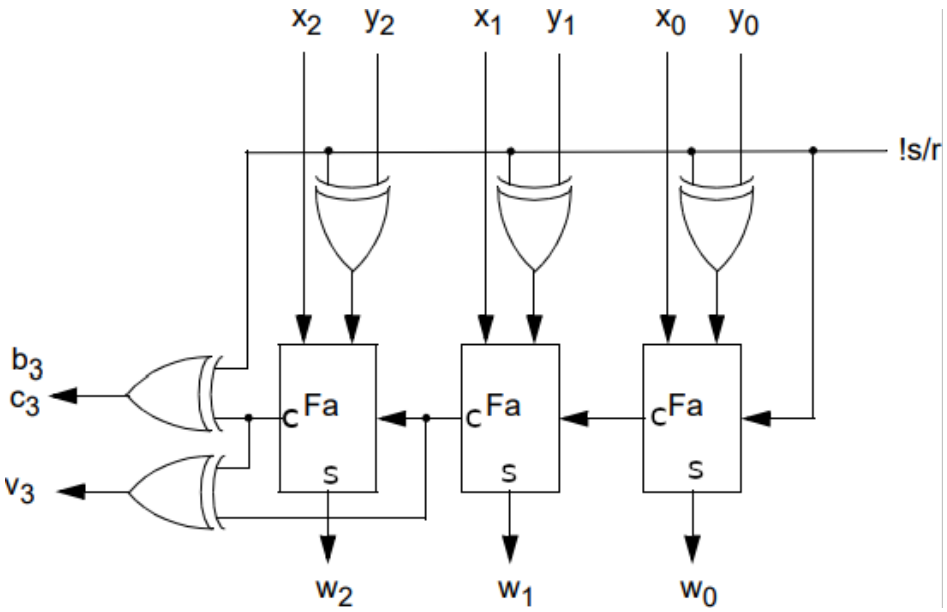
Índex

Sumador/Restador

- Combinarem el sumador i el restador a un únic CLC
- Necessitarà una entrada de control ($!s/r$)
  - $!s/r = 0$ indica que ha de sumar
    - Els FA's rebran Y
    - El primer FA rebrà un "0" a la tercera entrada
  - $!s/r = 1$ indica que ha de restar
    - Els FA's rebran $!Y$
    - El primer FA rebrà un "1" a la tercera entrada
- Implementació per a $n = 3$

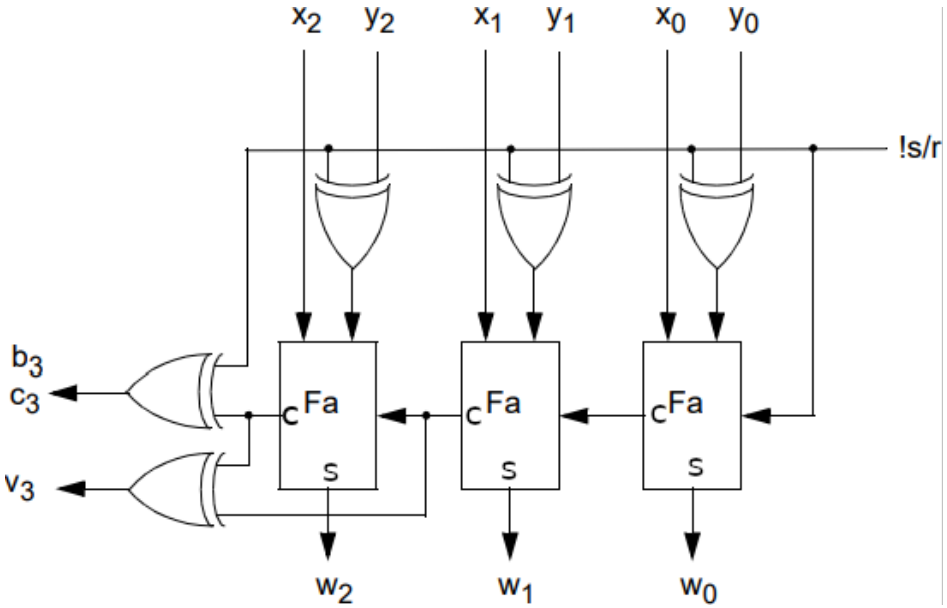


Sumador/Restador: entrada $!s/r$



- $!s/r = 0$ indica que ha de sumar
  - Com 0 *XOR* $x = x$, els FA's rebran Y
  - El primer FA rebrà un "0" a la tercera entrada
- $!s/r = 1$ indica que ha de restar
  - Com 1 *XOR* $x = !x$, els FA's rebran $!Y$
  - El primer FA rebrà un "1" a la tercera entrada

Sumador/Restador: No representativitat



- No representativitat per a cada operació i interpretació

	+	-
$\mathbb{N}$	c_n	b_n
$\mathbb{Z}$	v_n	v_n

  - A la resta de naturals, la no representativitat es detecta amb $!c_n$
    - Al circuit, $b_n = XOR(c_n, !s/r)$, com $!s/r = 1 \implies b_n = !c_n$

Índex

Multiplicació per potències de 2

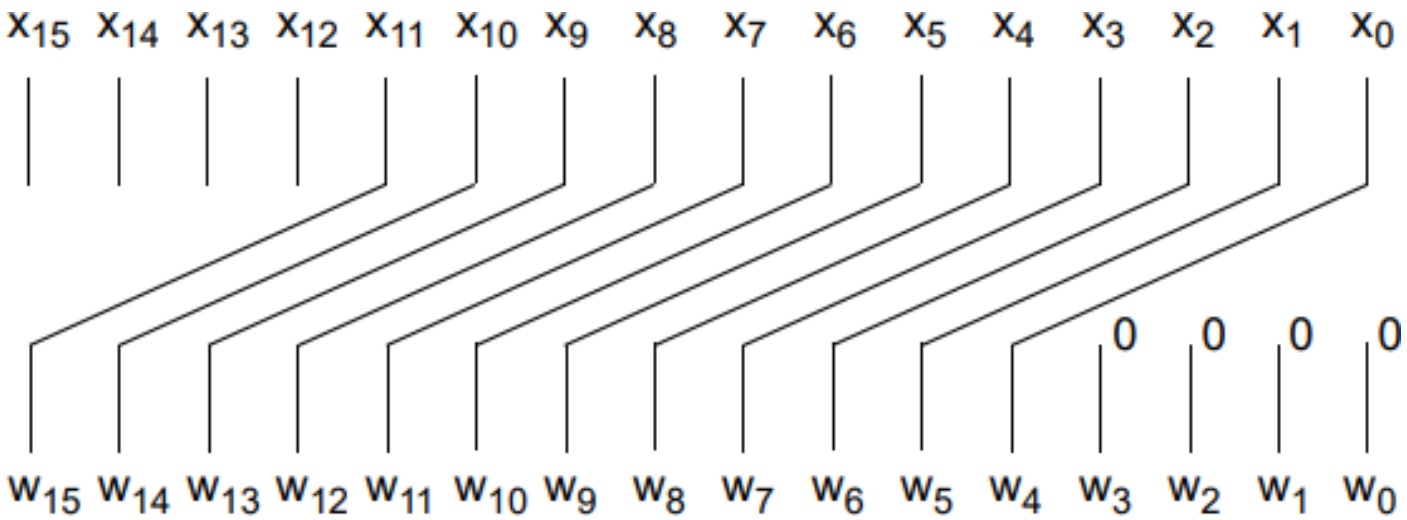
- Multiplicar per 2
  - Com amb naturals
    - Desplaçar els bits una posició a l'esquerra i posar un "0" a la posició 0
    - *Shift Left*-1 (SL-1)
  - No representativitat: si el signe de l'entrada és diferent del del resultat
    - Els dos bits alts de X no tenen el mateix valor
  - Exemples ($n = 4$): W=SL-1(X)

X	X_s	W	W_s
0011	3	0110	6
1111	-1	1110	-2
0111	7	1110	-2
1011	-5	0110	6

- Multiplicar per 2^k (SL-k)
  - Com als naturals, desplaçar k posicions a l'esquerra
  - No representativitat:
    - Si els $\mathbf{k+1}$ dígits de més pes de X no tenen el mateix valor

Implementació SL-k

- S'utilitza per a naturals i per a enters
- Exemple per a $n = 16, k = 4$
  - No detecta la no representativitat



Índex

Divisió entre potències de 2

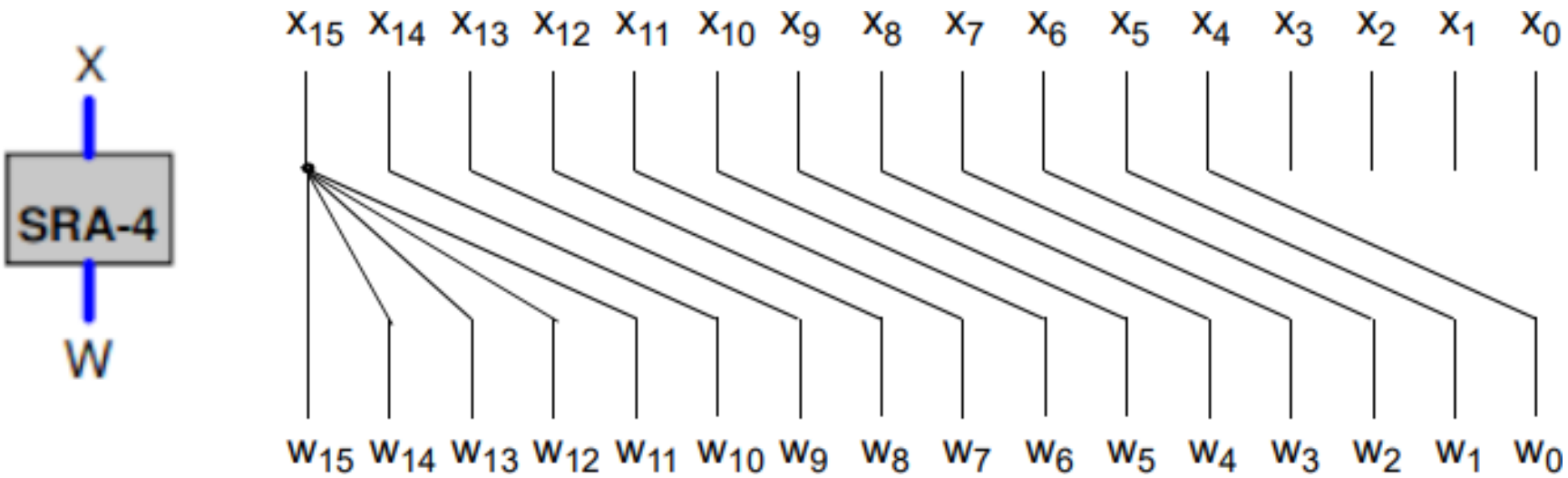
- Divisió entre 2
  - No podem fer-ho com als naturals
    - Desplaçar una posició a la dreta i posar un "0" al bit de més pes
    - *Shift-Right Logically-1* (SRL-1)
    - El resultat, interpretat en Ca2, seria sempre positiu!
  - A Ca2, cal desplaçar a la dreta i replicar el bit de més pes
    - *Shift-Right Arithmetically* (SRA-1)
    - Resultat sempre representable
  - Exemples ($n = 4$), W =SRA-1(X)

X	X_s	W	W_s
0011	3	0001	1
1101	-3	1110	-2
0111	7	0011	3
1000	-8	1100	-4

- Divisió entre 2^k
  - SRA-k: desplaçar a la dreta k posicions i replicar bit de més pes

Implementació SRA-k

- S'utilitza per a vectors de bits que volem interpretar com a enters
- Exemple per a $n = 16, k = 4$



- $W_s = \lfloor X_s/2^k \rfloor$
  - Sempre serà representable

Índex

Comparadors de nombres enters

- Dissenyarem dos tipus de comparadors d'enters
  - *L*ess Than: $w = LT(X, Y)$

```
if (Xs<Ys) w=1; else w=0;
```

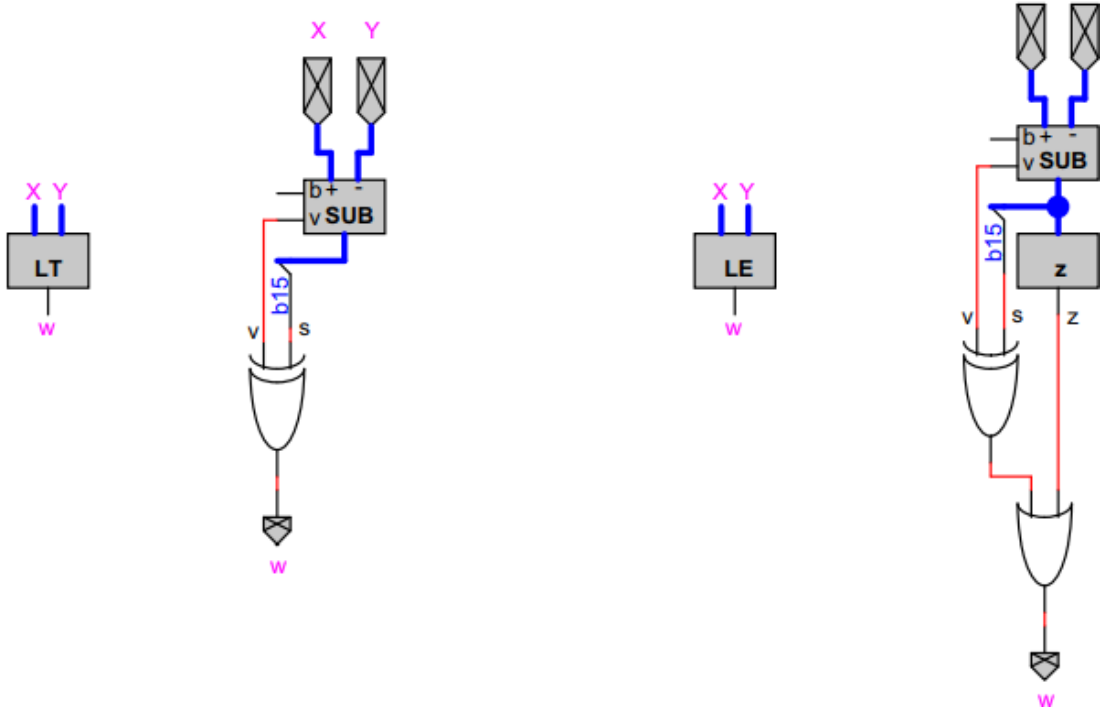
  - *L*ess or Equal: $w = LE(X, Y)$

```
if (Xs<=Ys) w=1; else w=0;
```

  - Són diferents dels comparadors de naturals, *LTU* i *LEU*
- La comparació d'igualtat la podrem fer amb el bloc *EQ* vist al tema 4
- La implementació calcula $X_s - Y_s$ i n'examina el signe del resultat
  - Si el resultat és negatiu $\implies X_s < Y_s$
  - I si el resultat de la resta és no representable com a enter?
    - Només necessitem saber el signe del resultat, no el valor concret
    - No representativitat detectada perquè el signe era incorrecte
    - Podem saber el signe correcte si invertim el signe obtingut

Blocs *LT* i *LE*

- *LT* (*Less Than*) s'implementarà amb el bloc *SUB*
  - $X_s < Y_s \iff X_s - Y_s < 0 \iff$
    $((v = 0) \cdot (w_{15} = 1)) + ((v = 1) \cdot (w_{15} = 0)) \iff v \ XOR \ w_{15}$
- *LE* (*Less or Equal*) s'implementarà amb els blocs *SUB* i *z*
  - $LE(X, Y)$ retorna "1" $\iff EQ(X, Y) \ OR \ LT(X, Y)$
- Encapsulat i implementació:



Exemple

- Donats vectors X i Y de 8 bits, determineu el resultat de les comparacions entre X i Y interpretant-los com a naturals i com a enters

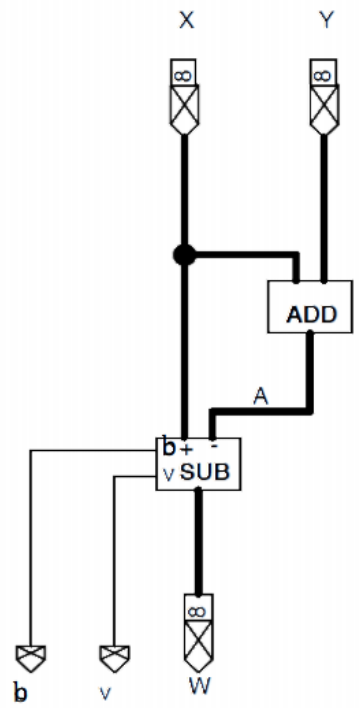
X	Y	$EQ(X,Y)$	$LTU(X,Y)$	$LEU(X,Y)$	$LT(X,Y)$	$LU(X,Y)$
10100101	01001111	0	0	0	1	1
00011010	01011100	0	1	1	1	1
10101100	10101100	1	0	1	0	1
01101010	10101000	0	1	1	0	0
00000001	00000000	0	0	0	0	0

Índex

Exercici 1516Q1E2

Dado el siguiente circuito CLC y los vectores de 8 bits de entrada X y Y, completad la siguiente tabla.

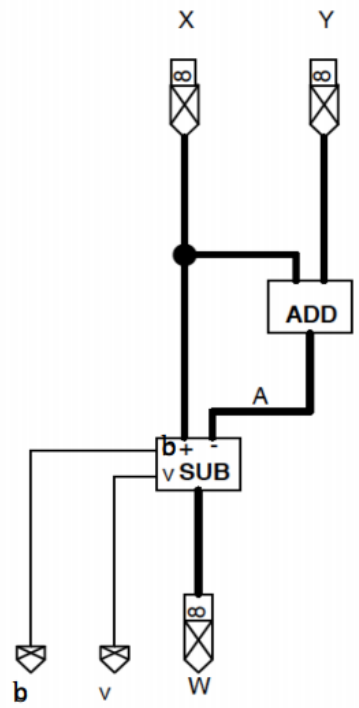
X	Y	A	W	Wu	Ws	b	v
00000000	11111111						
11111111	00000001						
10000000	10101010						



Exercici 1516Q1E2

Dado el siguiente circuito CLC y los vectores de 8 bits de entrada X y Y, completad la siguiente tabla.

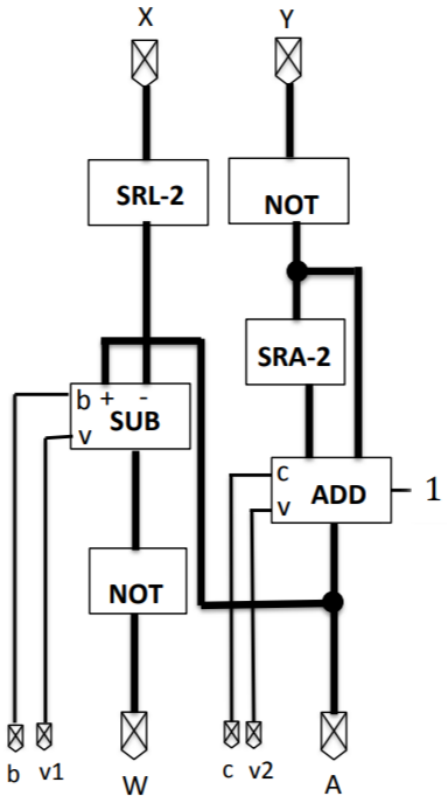
X	Y	A	W	Wu	Ws	b	v
00000000	11111111	11111111	00000001	1	1	1	0
11111111	00000001	00000000	11111111	255	-1	0	0
10000000	10101010	00101010	01010110	86	86	0	1



Exercici 1819Q1E2

Dado el esquema del CLC a bloques de la derecha, completad la siguiente tabla que indica el valor de las salidas del circuito para cada uno de los cuatro casos concretos de valores de las entradas (un caso por fila).

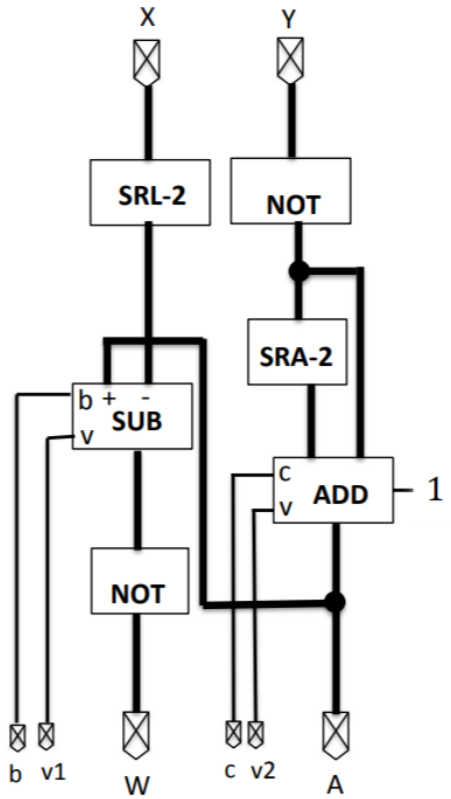
X	Y	b	v1	c	v2	W	A
0000 0000	1111 1111						
1000 0001	1111 1100						
1000 0000	0111 1111						
1111 1111	1000 0000						



Exercici 1819Q1E2

Dado el esquema del CLC a bloques de la derecha, completad la siguiente tabla que indica el valor de las salidas del circuito para cada uno de los cuatro casos concretos de valores de las entradas (un caso por fila).

X	Y	b	v1	c	v2	W	A
0000 0000	1111 1111	0	0	0	0	11111110	00000001
1000 0001	1111 1100	1	0	0	0	00011011	00000100
1000 0000	0111 1111	0	0	1	1	10111110	01100001
1111 1111	1000 0000	0	1	0	1	10011111	10011111



Índex

Conclusions

- Un mateix vector de bits pot ser interpretat de formes diferents
  - $X = 10011 \implies X_u = 19, X_s = -13$
    - Pel context, haurem de deduir com interpretar els vectors de bits
  - Ens passa el mateix amb les paraules
    - La paraula "son" té interpretacions diferents en funció de l'idioma
    - ensonyament (català), forma verbal del verb ésser (castellà), fill (anglès), so (francès)
- Hem vist complement a 2, una possible representació dels enters
  - Permet fer la suma d'enters i de naturals amb el mateix bloc CLC
  - L'extensió de rang s'implementa mitjançant la replicació del bit de signe
  - La detecció de no representativitat dels resultats es complica respecte els naturals però també és factible
- No oblideu fer l'ET5 a Atenea

Índex

- Altres formes de representar els enters
  - Complement a 1
  - Base -2
- Rangs de tipus natural i enter a Swift (similars als de C, ...)

Type	Minimum	Maximum
Int8	-128	127
Int16	-32,768	32,767
Int32	-2,147,483,648	2,147,483,647
Int64	- 9,223,372,036,854,775,808	9,223,372,036,854,775,807
Int	- 9,223,372,036,854,775,808	9,223,372,036,854,775,807
UInt8	0	255
UInt16	0	65,535
UInt32	0	4,294,967,295
UInt64	0	18,446,744,073,709,551,615
UInt	0	18,446,744,073,709,551,615

[2]

Referències I

Llevat que s'indiqui el contrari, les figures, esquemes, cronogrames i altre material gràfic o bé han estat extrets de la documentació de l'assignatura elaborada per Juanjo Navarro i Toni Juan, o corresponen a enunciats de problemes i exàmens de l'assignatura, o bé són d'elaboració pròpia.

[1] [Online]. Available: https://www.colorado.edu/ecenter/sites/default/files/styles/small/public/page/reusesymbol.png?itok=YUtLBOY1.

[2] [Online]. Available: https://www.topjavatutorial.com/mobile/swift/swift-data-types/.

Introducció als Computadors

Tema 5: Nombres enters

http://personals.ac.upc.edu/enricm/Docencia/IC/IC5.pdf

Enric Morancho
(enricm@ac.upc.edu)

Departament d'Arquitectura de Computadors
Facultat d'Informàtica de Barcelona
Universitat Politècnica de Catalunya

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

2020-21, 1^{er} quad.