Introducció als Computadors

Tema 4: Blocs aritmètics combinacionals per a naturals

http://personals.ac.upc.edu/enricm/Docencia/IC/IC4.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 del rellotge (o modular)

[1]

Índex

Introducció

- Hem vist com sintetitzar CLC's a partir de la seva taula de veritat
  - Descomposició de la funció en suma de *minterms*
- Però aquest mètode no serveix per a circuits amb moltes entrades
  - Ex: un sumador que rebi dos enters de 16 bits té 32 entrades
    - La seva taula de veritat tindria $2^{32} \approx 4.000$ milions de files
  - Amb dades de 32 o 64 bits, encara seria molt pitjor
- Dissenyarem CLC's que realitzin càlculs aritmètics amb naturals
  - Per fer un sumador binari adaptarem a base 2 el que sabeu de base 10
  - Respecte als altres blocs, farem dissenys ad-hoc per a cada cas
    - Intentarem fer dissenys modulars, basant-nos en blocs més senzills

Aritmètica convencional vs. computacional

- Aritmètica convencional: el conjunt dels naturals és infinit
  - Si cal, el resultat de les operacions pot tenir més dígits que els operands
- Aritmètica computacional: utilitza subconjunt finit dels naturals
  - Els naturals es representen amb n bits (a IC, n=16)
    - El màxim natural representable és $2^n - 1$ (a IC, $2^{16} - 1 = 65.535$)
    - El resultat de les operacions no pot tenir més dígits que els operands
  - Penseu en el comptaquilòmetres d'un cotxe o en un rellotge de busques
    - Després del quilòmetre 999.999 vindrà el 000.000
    - Després de les 12:59 vindrà la 1:00

[2]

[3]

- Aritmètica modular

Índex

Índex

Repassem suma en base 10

- Siguin X i Y dos vectors de n dígits decimals
- Volem calcular un vector W tal que $W_u = X_u + Y_u$
  - Quants dígits ha de tenir el resultat?
    - Com $0 \leq X_u, Y_u \leq 10^n - 1 \implies 0 \leq X_u + Y_u \leq 2 \cdot 10^n - 2 < 10^{n+1}$
    - Per tant, $X_u + Y_u$ pot necessitar $n + 1$ dígits decimals
- Algorisme de suma
  - Sumem els dígits de les unitats $t_0 = x_0 + y_0$
    - w_0 pren per valor el dígit de les unitats de t_0
    - Si $t_0 \geq 10$, me'n porto 1 per a la propera suma
  - Sumem els dígits de les desenes $t_1 = x_1 + y_1$
    - Incremento t_1 en 1 si me'n porto 1 de la suma anterior
    - w_1 pren per valor el dígit de les unitats de t_1
    - Si $t_1 \geq 10$, me'n porto una per a la propera suma
  - Repetim el procés amb centenes,... fins a processar els n dígits
    - Si al fer la darrera suma me'n porto una, el resultat tindrà $n + 1$ dígits

Algorisme de suma en base 10

- $X = x_{n-1}x_{n-2}\ldots x_1x_0$ $\quad x_i \in \{0,1,\ldots,9\}$
- $Y = y_{n-1}y_{n-2}\ldots y_1y_0$ $\quad y_i \in \{0,1,\ldots,9\}$
- $W = w_n w_{n-1}w_{n-2}\ldots w_1w_0$ $\quad w_i \in \{0,1,\ldots,9\}$
  - Volem que $W_u = X_u + Y_u$, on W té $n+1$ dígits, X i Y en tenen n

```
c0=0;    // A la suma de les unitats no em porto res
for (k=0; k<n; k=k+1) {
     wk = xk + yk + ck; // 0 ≤ wk ≤ (9+9+1) = 19
     if (wk >= 10) {
          wk = wk - 10; // Em quedo amb xifra de les unitats
          ck+1 = 1;// Me'n porto una (carry) per a propera suma
     }
     else ck+1 = 0;
}
wn = cn; // Val 1 si de la darrera suma me'n porto una
```

Generalització: algorisme en base b

- $X = x_{n-1}x_{n-2}\ldots x_1x_0$ $\quad x_i \in \{0,1,\ldots,b-1\}$
- $Y = y_{n-1}y_{n-2}\ldots y_1y_0$ $\quad y_i \in \{0,1,\ldots,b-1\}$
- Quants dígits té la suma?
  - Com $0 \le X_u, Y_u \le b^n - 1 \implies 0 \le X_u + Y_u \le 2 \cdot b^n - 2 < b^{n+1}$
    $\implies$ la suma pot necessitar $n+1$ dígits
- $W = w_n w_{n-1} w_{n-2} \ldots w_1 w_0$ $\quad w_i \in \{0,1,\ldots,b-1\}$
  - Volem que $W_u = X_u + Y_u$, on W té $n+1$ dígits, X i Y en tenen n

```
c0=0;    // A la suma de les unitats no em porto res
for (k=0; k<n; k=k+1) {
     wk = xk + yk + ck; // 0 ≤ wk ≤ (b-1+b-1+1) = 2b-1
     if (wk >= b) {
          wk = wk - b; // Em quedo amb xifra de les unitats
          ck+1 = 1;// Me'n porto una (carry) per a propera suma
     }
     else ck+1 = 0;
}
wn = cn; // Val 1 si de la darrera suma me'n porto una
```

Exercici: sumes en diverses bases

- Base 2:

$$\begin{array}{rcccc} & 1 & 1 & 0 & 1 \\ + & 1 & 1 & 0 & 1 \\ \hline \end{array}$$

- Base 8:

$$\begin{array}{rcccc} & 3 & 6 & 0 & 4 \\ + & 3 & 3 & 7 & 2 \\ \hline \end{array}$$

- Base 16:

$$\begin{array}{rcccc} & A & 3 & B & 5 \\ + & C & 7 & 2 & F \\ \hline \end{array}$$

Exercici: sumes en diverses bases

- Base 2:

$$\begin{array}{rcccc} & 1 & 1 & 0 & 1 \\ + & 1 & 1 & 0 & 1 \\ \hline 1 & 1 & 0 & 1 & 0 \end{array}$$

- Base 8:

$$\begin{array}{rcccc} & 3 & 6 & 0 & 4 \\ + & 3 & 3 & 7 & 2 \\ \hline & 7 & 1 & 7 & 6 \end{array}$$

- Base 16:

$$\begin{array}{rcccc} & A & 3 & B & 5 \\ + & C & 7 & 2 & F \\ \hline 1 & 6 & A & E & 4 \end{array}$$

"Paradoxes" de la suma en base 2

$$1 + 1 = 10$$
$$10 + 1 = 11$$
$$11 + 1 = 100$$

Got it?

[4]

Índex

Algorisme de suma en base 2

- $X = x_{n-1}x_{n-2}\ldots x_1x_0$ $\quad x_i \in \{0,1\}$
- $Y = y_{n-1}y_{n-2}\ldots y_1y_0$ $\quad y_i \in \{0,1\}$
- $W = w_n w_{n-1}w_{n-2}\ldots w_1w_0$ $\quad w_i \in \{0,1\}$
  - Volem que $W_u = X_u + Y_u$, on W té $n+1$ dígits, X i Y en tenen n

```
c0=0;    // A la suma de les unitats no em porto res
for (k=0; k<n; k=k+1) {
     wk = xk + yk + ck; // 0 ≤ wk ≤ 3
     if (wk >= 2) {
          wk = wk - 2; // Em quedo amb xifra de les unitats
          ck+1 = 1;// Me'n porto una (carry) per a propera suma
     }
     else ck+1 = 0;
}
wn = cn; // Val 1 si de la darrera suma me'n porto una
```

Una iteració del bucle de suma


```
w_k = x_k + y_k + c_k; // 0 ≤ w_k ≤ 3
if (w_k >= 2) {
    w_k = w_k - 2; // Em quedo amb xifra de les unitats
    c_{k+1} = 1;   // Me'n porto una per a propera suma
}
else c_{k+1} = 0;
```

- Donats els tres bits x_k, y_k, c_k retorna el nombre binari $c_{k+1}w_k$ que indica quants d'aquests tres bits valen "1"
  - Casos possibles:

x_k	0	0	0	0	1	1	1	1
y_k	0	0	1	1	0	0	1	1
+ c_k	+ 0	+ 1	+ 0	+ 1	+ 0	+ 1	+ 0	+ 1
c_{k+1} w_k	0 0	0 1	0 1	1 0	0 1	1 0	1 0	1 1

- Podem implementar-la mitjançant un CLC amb 3 entrades i 2 sortides
  - *Full-adder* (FA)

Full-adder (FA)

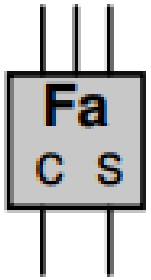
- Taula de veritat:

			c	s
x_k	y_k	c_k	c_{k+1}	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

$$x_k + y_k + c_k = 2 \cdot c_{k+1} + w_k$$

(+ i . aritmètiques, no lògiques)

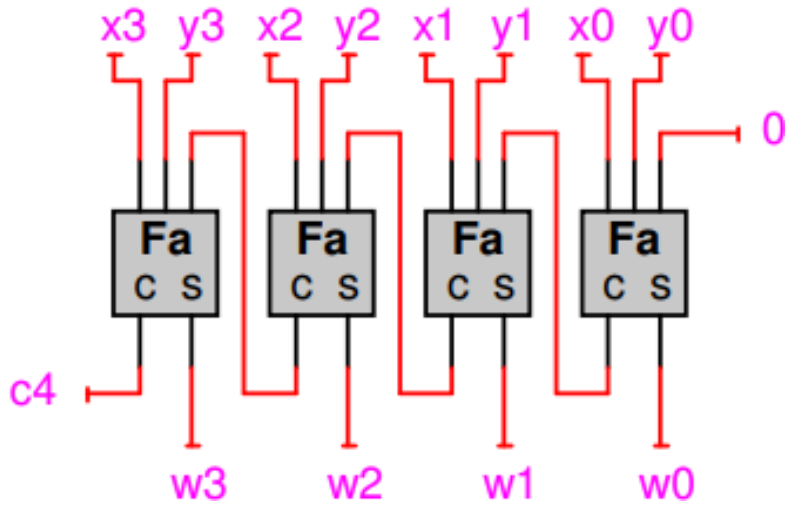
- Encapsulat:
  - Les entrades no estan etiquetades perquè són intercanviables



Construir sumador a partir de blocs FA

- Exemple $n = 4$
  - Al FA de la dreta li fem arribar un "0" com a una de les entrades perquè la suma de les unitats no pot portar-se res d'una suma anterior
  - Si c_4 val "1" indica que el resultat de la suma necessitaria 5 bits

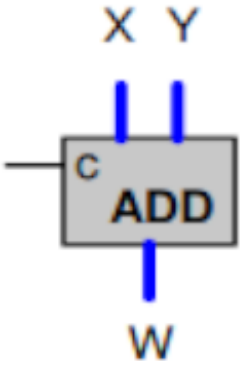
$$
\begin{aligned}
c_0 &= 0\ ; \\
(c_1, w_0) &= \mathbf{Fa}\,(x_0, y_0, c_0)\ ; \\
(c_2, w_1) &= \mathbf{Fa}\,(x_1, y_1, c_1)\ ; \\
(c_3, w_2) &= \mathbf{Fa}\,(x_2, y_2, c_2)\ ; \\
(c_4, w_3) &= \mathbf{Fa}\,(x_3, y_3, c_3)\ ;
\end{aligned}
$$

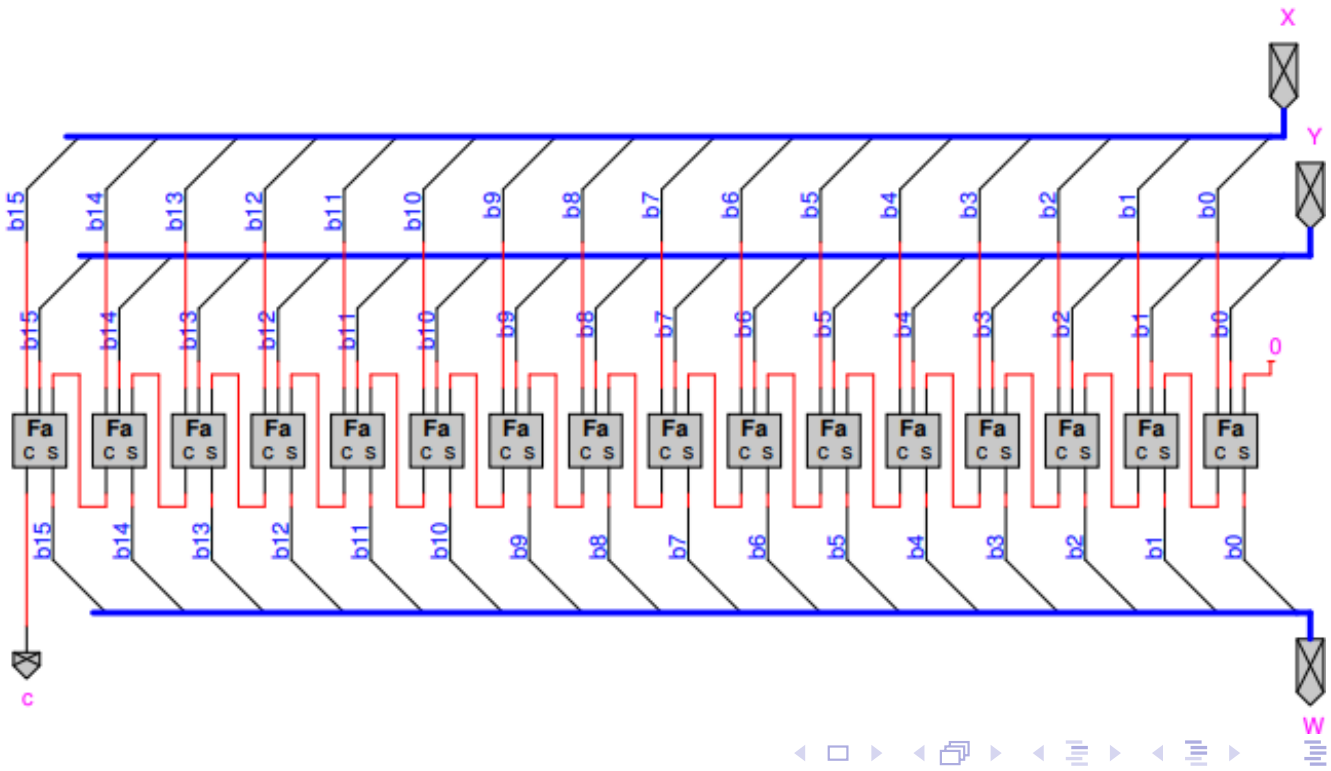


- *Ripple carry adder* (RCA)
  - Hi ha altres dissenys de sumadors amb millors prestacions

Bloc ADD

- Entrades: X, Y de $n = 16$ bits
  - Els cables gruixuts blaus representen busos (feixos) de n senyals
- Sortides: W de $n = 16$ bits i c (*carry*) d'un bit
  - $W_u = (X_u + Y_u) \mod 2^n$
  - $c = 1 \iff X_u + Y_u \geq 2^n$
    - $c = 1$ indica que el resultat necessitaria $n + 1$ bits (*overflow*)
    - La propagació del *carry* per tots els FA's determinarà el camí crític





ADD: suma no representable (*overflow*)

- El bloc aritmètic ADD **no** calcula la suma convencional $(X_u + Y_u)$
  - El bloc ADD calcula la suma mòdul 2^n
    - $W = ADD(X, Y)$
    - En general, $W_u = (X_u + Y_u) \mod 2^n \neq X_u + Y_u$
    - El que seria el bit w_n no forma part del resultat W
- En molts casos, el resultat de la suma convencional i el de la suma modular coincideixen
  - Quan no ho facin, $W_u \neq X_u + Y_u$, direm que **el resultat de la suma convencional no és representable amb n bits**
    - *Overflow* (sobreeximent)
    - La sortida c del bloc ADD tindrà el valor "1" per indicar-ho
  - Sempre es complirà que $X_u + Y_u = c \cdot 2^n + W_u$
    - c seria el bit w_n de la suma convencional
    - W conté els n bits baixos de la suma convencional

Exercici ($n = 4$)

- Donats vectors X i Y de 4 bits, determineu el resultat de la suma modular en 4 bits ($W = ADD(X, Y)$ i W_u), si s'ha produit *overflow* (c) i el resultat de la suma convencional ($X_u + Y_u$)

X	X_u	Y	Y_u	$W = ADD(X, Y)$	W_u	c	$X_u + Y_u$
1100	12	0011	3				
0010	2	1111	15				
1111	15	1111	15				

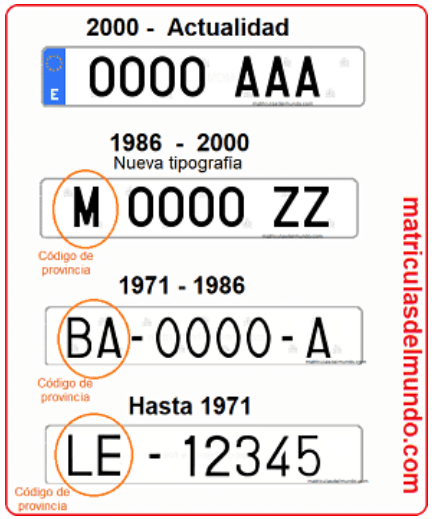
Exercici ($n = 4$)

- Donats vectors X i Y de 4 bits, determineu el resultat de la suma modular en 4 bits ($W = ADD(X, Y)$ i W_u), si s'ha produit *overflow* (c) i el resultat de la suma convencional ($X_u + Y_u$)

X	X_u	Y	Y_u	$W = ADD(X, Y)$	W_u	c	$X_u + Y_u$
1100	12	0011	3	1111	15	0	15
0010	2	1111	15	0001	1	1	17
1111	15	1111	15	1110	14	1	30

Overflows al món real

- Year 2000 problem [5]
- Gangnam Style overflows INT_MAX, forces YouTube to go 64-bit [6]
- GPS Week Number Rollover [7]
- IPv4 address exhaustion [8]
- Japan running out of credit card numbers [9]
- Year 2038 problem [10]
- Sistema de fitxers NTFS: data límit 28 de maig del 60.056 [11]
- Matrícules dels cotxes
- Marcadors esportius



[12]

[13]

[14]

Síntesi del *full-adder* (FA)

- Podem sintetitzar el *full-adder* a partir de la seva taula de veritat
  - Descomposició en suma de *minterms*
- Però el construirem a partir d'un CLC més senzill, el *half-adder*
  - Disseny modular

Bloc Half-Adder (HA)

- El *half-adder* calcularà les taules de sumar en base 2 (+ aritmètic)

x	y	$x+y$
0	0	0
0	1	1

x	y	$x+y$
1	0	1
1	1	10

  - Com un dels resultats necessita dos bits, codifiquem tots amb dos bits

x	y	$x+y$
0	0	00
0	1	01

x	y	$x+y$
1	0	01
1	1	10

- Half-Adder: CLC que implementa taules de sumar en base 2
  - Taula de veritat del HA (separant els bits de sortida)

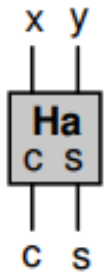
x	y	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

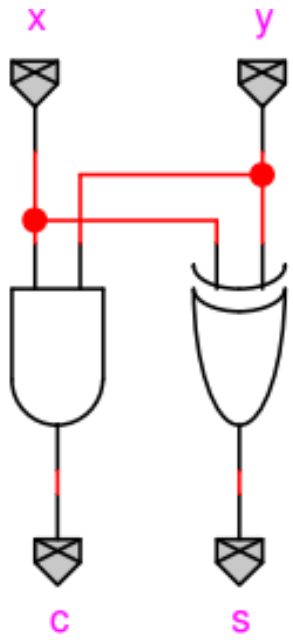
$$x + y = 2 \cdot c + s$$

(+ i . aritmètics, no lògics)

Síntesi del *Half-adder* (HA)

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

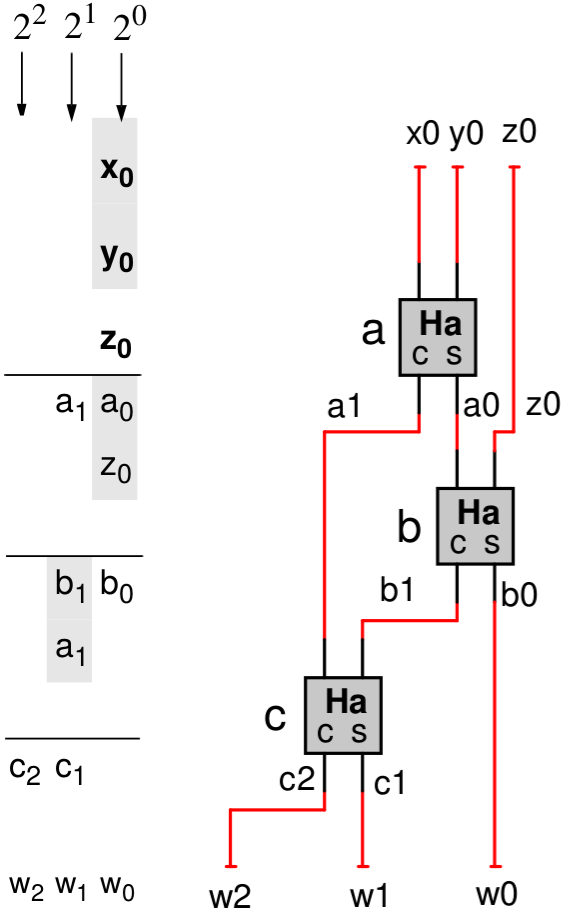




- Síntesi de cada sortida:
  - c: porta AND-2
  - x: porta XOR-2
- Encapsulat:
  - No cal identificar les entrades perquè són intercanviables
  - Sí que cal identificar les sortides c i s

Síntesi de *full-adder* amb blocs *half-adder*

- Utilitzant tres blocs HA (aprofitem associativitat i respectem pesos)

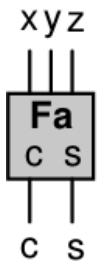


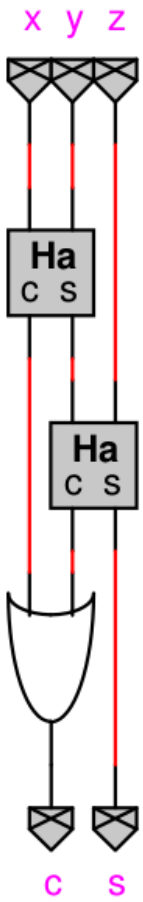
x_0	y_0	z_0	a_1	a_0	b_1	b_0	c_2	c_1	w_2	w_1	w_0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	1
0	1	0	0	1	0	1	0	0	0	0	1
0	1	1	0	1	1	0	0	1	0	1	0
1	0	0	0	1	0	1	0	0	0	0	1
1	0	1	0	1	1	0	0	1	0	1	0
1	1	0	1	0	0	0	0	1	0	1	0
1	1	1	1	0	0	1	0	1	0	1	1

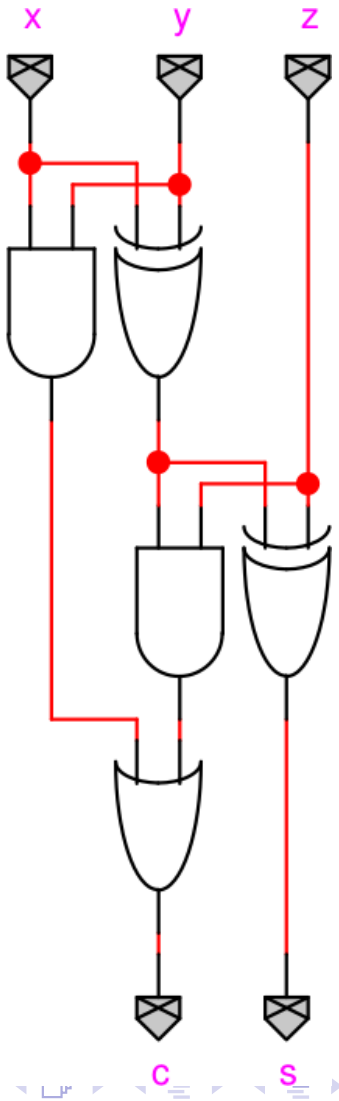
- Observem que w_2 sempre val "0"
  - Ja quadra, perquè hem vist que el resultat del FA només té 2 bits
- Simplificarem el càlcul de w_1 fent OR-2(a1, b1)

Síntesí de *full-adder* amb blocs *half-adder*

- Utilitzant dos blocs HA i una porta OR-2
  - També amb 1 porta OR-2, 2 portes AND-2 i 2 portes XOR-2







Índex

Multiplicar per potències de 2

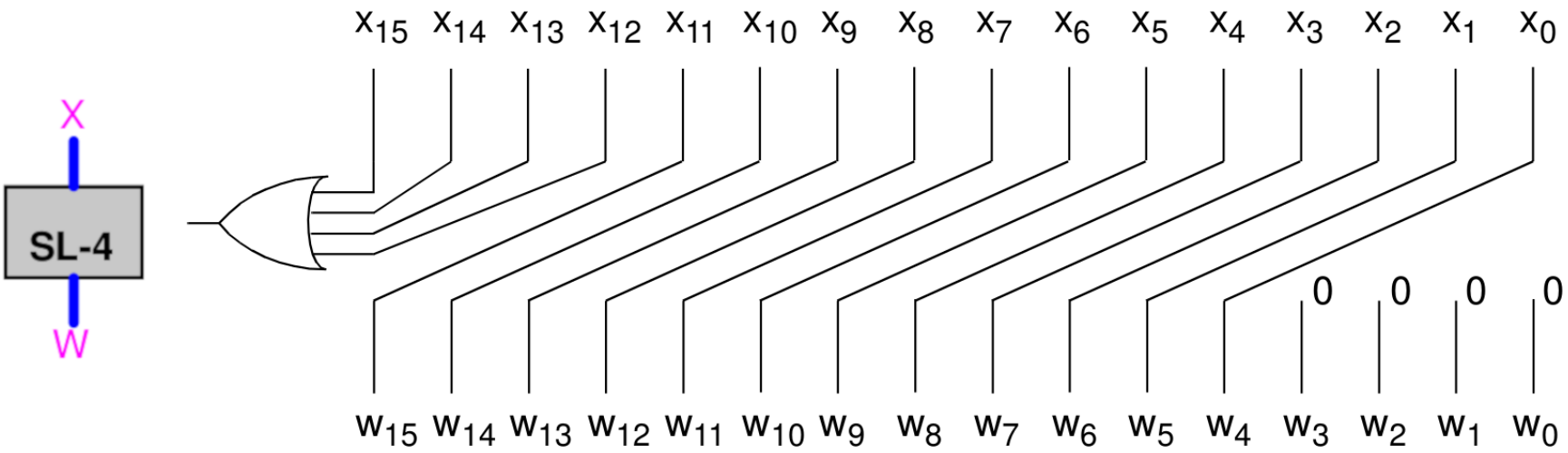
- En base b, multiplicar un número per b (10_b) és equivalent a desplaçar tots els dígits una posició a l'esquerra i afegir un 0 per la dreta
  - Exemples:
    - $25 \cdot 10 = 250$
    - $13 = 1101_2 \implies 13 \cdot 2 = 11010_2$
    - $75 = 4B_{16} \implies 75 \cdot 16 = 4B0_{16}$
- En base b, multiplicar un número per b^k ($10^k{}_b$) és equivalent a desplaçar els dígits k posicions a l'esquerra i afegir k 0's per la dreta
  - Exemples:
    - $25 \cdot 10^4 = 250.000$
    - $13 = 1101_2 \implies 13 \cdot 2^2 = 110100_2$
    - $75 = 4B_{16} \implies 75 \cdot 16^3 = 4B000_{16}$
- Demostració: a partir de la fórmula que calcula X_u
  - $X = x_{n-1}x_{n-2}\ldots x_1x_0, \quad x_i \in \{0,1,\ldots,b-1\} \implies X_u = \sum_{i=0}^{n-1} x_i \cdot \mathbf{b}^i$
    - Si $Y = x_{n-1}x_{n-2}\ldots x_1x_00 \implies Y_u = b \cdot X_u$
    - Si $Y = x_{n-1}x_{n-2}\ldots x_1x_0000 \implies Y_u = b^3 \cdot X_u$

Resultat no representable amb n bits

- A una implementació hardware, la mida de l'entrada i del resultat de l'operació estarà limitada a n bits
- A un nombre de n bits, desplaçar els dígits k posicions a l'esquerra provoca que es descartin els k bits de més pes de l'entrada
  - Exemple:
    - Si $n = 8$, desplaçar el nombre 10100011 tres posicions a l'esquerra resulta 00011000
    - Els bits "101" desapareixen del resultat
- Si algun d'aquests k bits de més pes val "1", el resultat del desplaçament no serà representable amb n bits

Implementació bloc SL-k

- SL-k: *Shift Left-k*
  - Implementació per a $k = 4$
    - El resultat serà no representable amb 16 bits si algun dels $k = 4$ bits de més pes de X val "1"
    - El bloc SL-4 mostrat no incorpora aquesta informació com a sortida



- $W_u = (X_u \cdot 2^k) \mod 2^n$

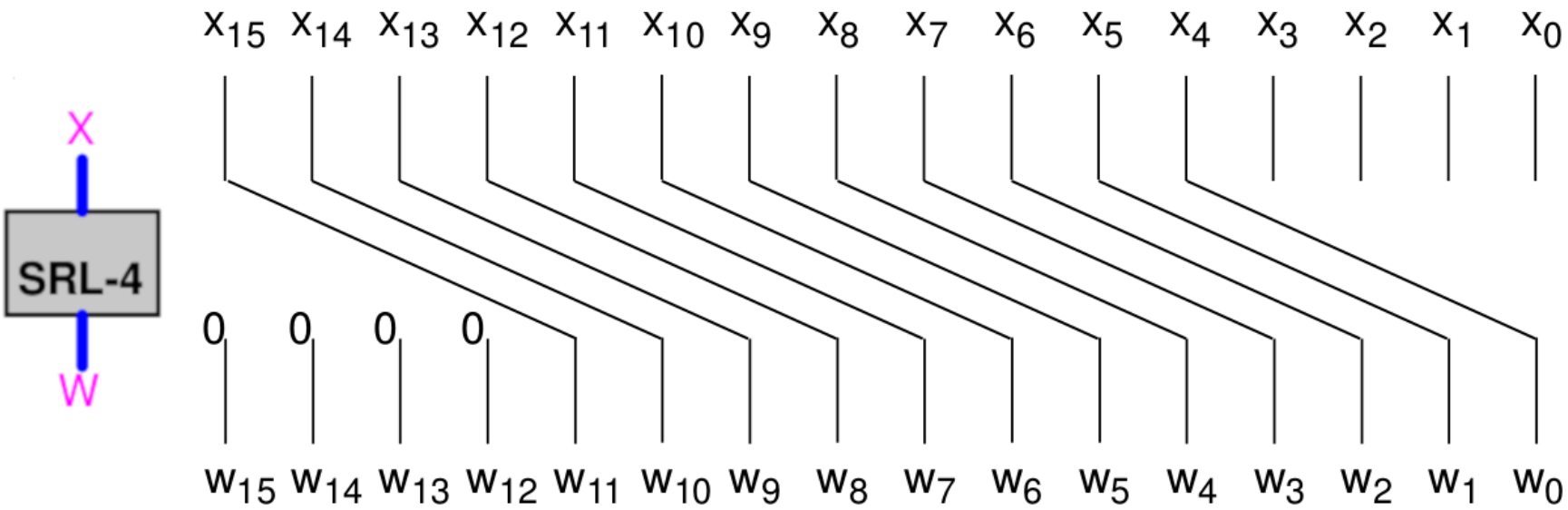
Índex

Dividir entre potències de 2

- En base b, la divisió entera d'un número entre b (10_b), arrodonint per defecte, és equivalent a descartar el dígit de menys pes i desplaçar tots els dígits una posició a la dreta
  - Exemples:
    - $25/10 = 2$
    - $13 = 1101_2 \implies 13/2 = 110_2$
    - $75 = 4B_{16} \implies 75/16 = 4_{16}$
- En base b, la divisió entera d'un número entre b^k, arrodonint per defecte, és equivalent a descartar els k dígits de menys pes i desplaçar els dígits k posicions a la dreta
  - Exemples:
    - $25.000/10^4 = 2$
    - $13 = 1101_2 \implies 13/2^3 = 1_2$
    - $331 = 14B_{16} \implies 331/16^2 = 1_{16}$
- Demostració: a partir de la fórmula que calcula X_u
  - $X = x_{n-1}x_{n-2}\ldots x_1x_0, \quad x_i \in \{0,1,\ldots,b-1\} \implies X_u = \sum_{i=0}^{n-1} x_i \cdot \mathbf{b}^i$
    - Si $Y = x_{n-1}x_{n-2}\ldots x_1 \implies Y_u = \lfloor X_u/b \rfloor$
    - Si $Y = x_{n-1}x_{n-2}\ldots x_2 \implies Y_u = \lfloor X_u/b^2 \rfloor$

Implementació bloc SRL-k

- SRL-k: *Shift Right Logically-k*
  - Implementació per a $k = 4$



- $W_u = \lfloor X_u / 2^k \rfloor$
  - El resultat sempre és representable perquè és menor que l'entrada
- Quan estudiem la representació dels enters, veurem que el bloc SRL no serà utilitzable amb nombres enters mentre que el bloc SL serà vàlid tant per naturals com per a enters

Índex

Comparadors de nombres naturals

- Dissenyarem tres tipus de comparadors de naturals
  - *E*Qual: $w = EQ(X, Y)$

```
if (X==Y) w=1; else w=0;
```

  - *L*ess Than Unsigned: $w = LTU(X, Y)$

```
if (Xu<Yu) w=1; else w=0;
```

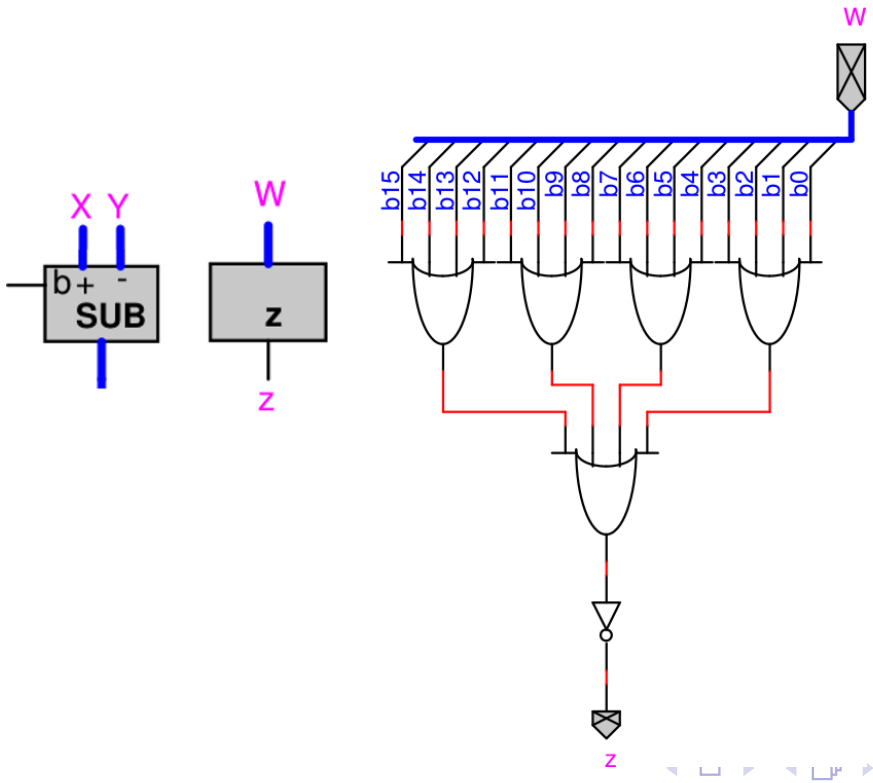
  - *L*ess or Equal Unsigned: $w = LEU(X, Y)$

```
if (Xu<=Yu) w=1; else w=0;
```

  - No farem comparadors $>$ i $\geq$ perque $A > B \iff B < A$ i $A \geq B \iff B \leq A$
- Entrades: dos nombres naturals codificats amb 16 bits
- Sortida: un bit
  - 0: la comparació és falsa
  - 1: la comparació és certa
- Els implementarem utilitzant CLC's ja vistos o més senzills
  - Disseny modular

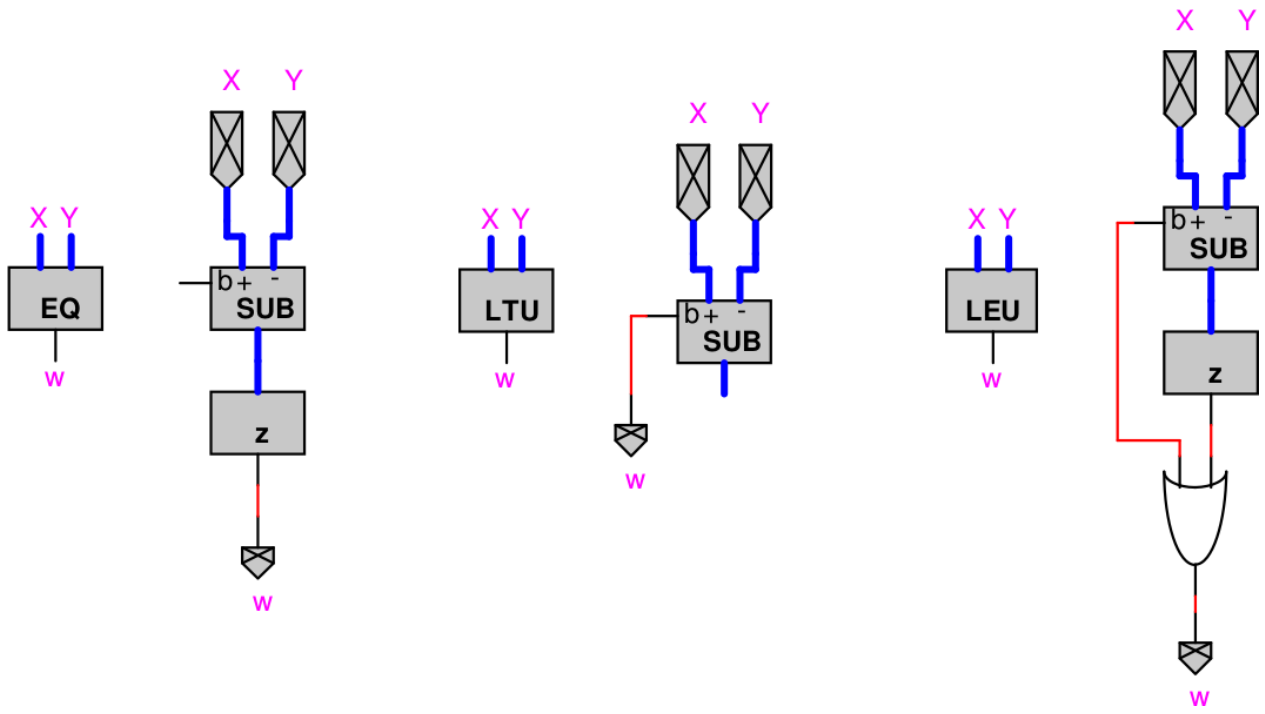
Comparadors: CLC's auxiliars

- $SUB(X, Y)$: bloc restador
  - El bloc SUB calcula $X_u - Y_u$
    - Si $X_u < Y_u$, la sortida b valdrà "1"
    - Indica que el resultat no és representable (resultat negatiu)
- $z(W)$: bloc z (zero)
  - El bloc z retorna "1" si els n bits de l'entrada valen "0"
    - Fa la OR-n de tots els bits del resultat i inverteix la sortida



Comparadors: implementació

- $EQ(X, Y)$ s'implementarà amb el bloc SUB i el bloc z
  - $X = Y \iff X_u - Y_u = 0$
- $LTU(X, Y)$ s'implementarà amb el bloc SUB
  - $X_u < Y_u \iff X_u - Y_u < 0 \iff b = 1$
- $LEU(X, Y)$ s'implementarà amb els blocs SUB i z
  - $X_u \leq Y_u \iff EQ(X, Y)$ o $LTU(X, Y)$



Exemple

- Donats vectors X i Y de 8 bits, determineu el resultat de les comparacions entre X i Y

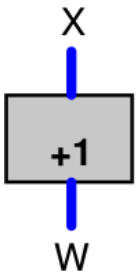
X	Y	$EQ(X,Y)$	$LTU(X,Y)$	$LEU(X,Y)$
10101100	01001010	0	0	0
00010111	01011101	0	1	1
10101010	10101010	1	0	1

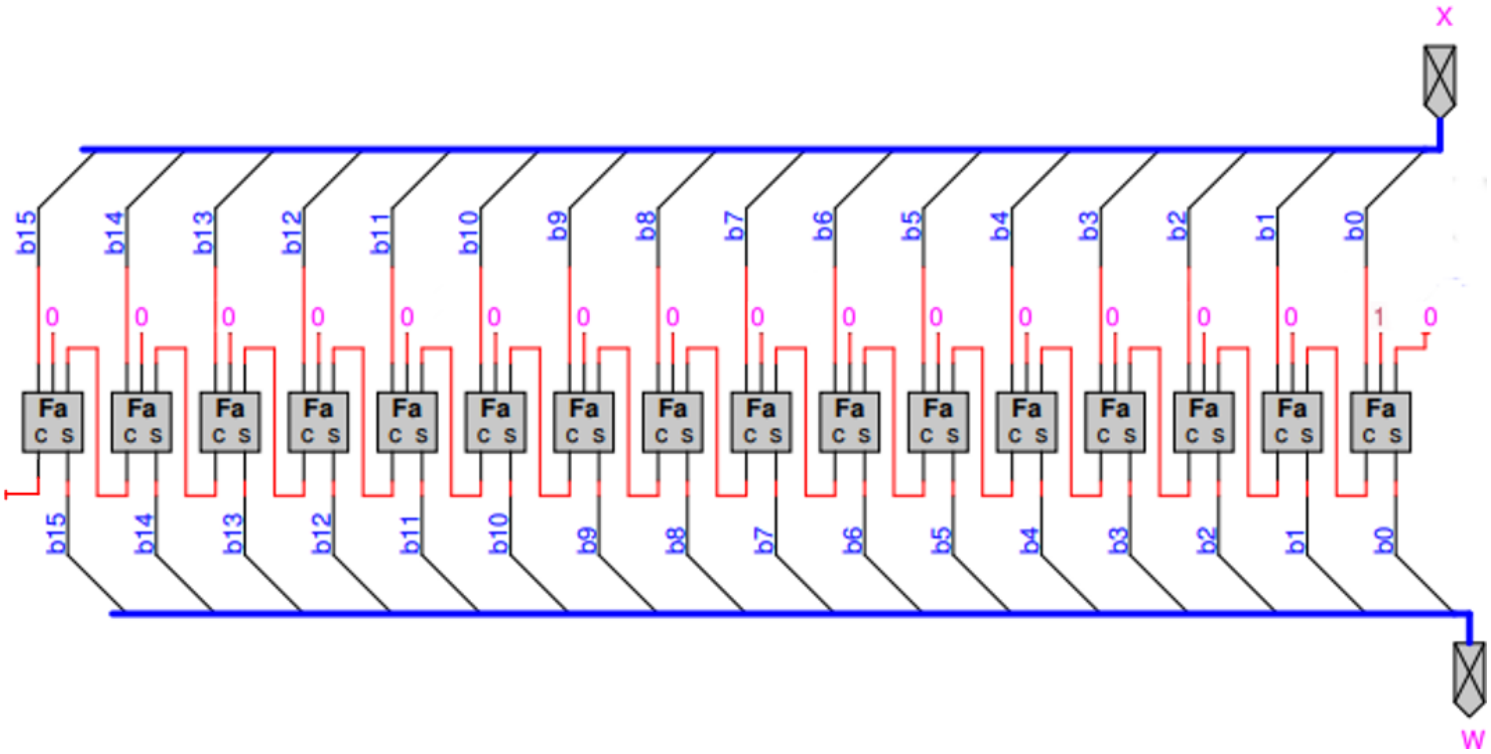
Índex

Índex

Incrementador amb n FA's

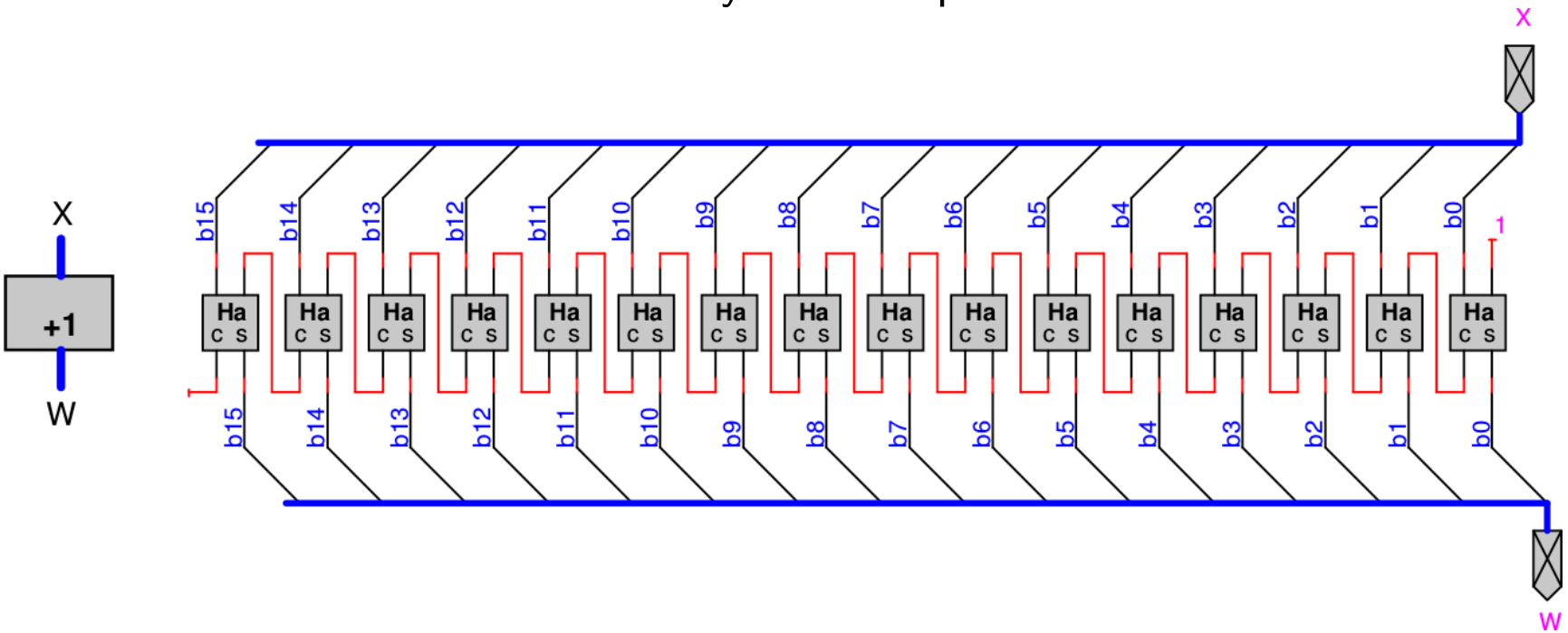
- Incrementador: CLC que rep d'entrada un bus X de n bits i retorna un bus W de n bits tal que $W_u = (X_u + 1) \mod 2^n$
- 1 bloc ADD de n bits o directament n blocs FA
  - Suma els vectors de n bits X i $000\ldots001$
  - Un dels operands dels FA sempre valdria "0"





Incrementador amb n HA's

- Incrementador: CLC que rep d'entrada un bus X de n bits i retorna un bus W de n bits tal que $W_u = (X_u + 1) \mod 2^n$
  - Cada HA rep un element de X
  - El primer HA també rep un "1"
  - La resta de HA's reben el carry calculat pel HA anterior



- Podem substituir el primer HA per una NOT i connectar b0 al segon

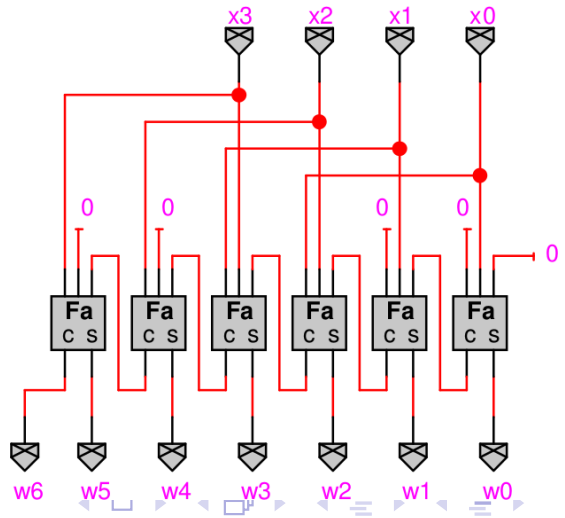
Índex

Multiplicador per 5

CLC que amb entrada X de n bits i sortida W de m bits on $W_u = 5 \cdot X_u$.
Determineu m per garantir que el resultat sempre sigui representable.

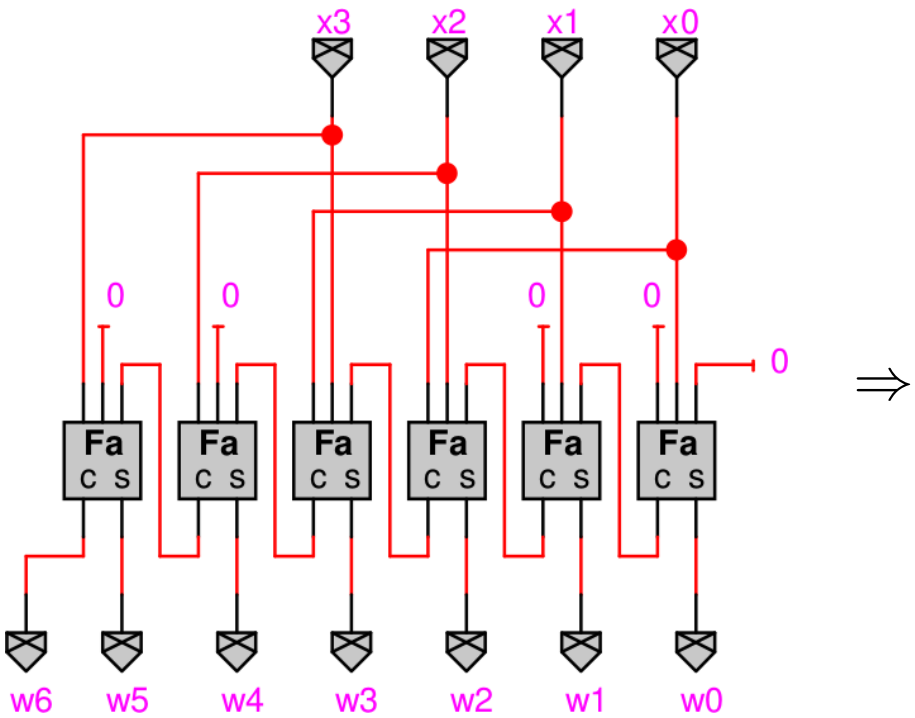
- $W_u = 5 \cdot X_u = 4 \cdot X_u + X_u = 2^2 \cdot X_u + X_u$
  - $X_u \cdot 2^2$ requereix $n + 2$ bits
  - Sumar una dada de $n + 2$ bits amb una de n bits genera $n + 3$ bits
  - Per tant, $m = n + 3$
- Possibles dissenys:
  - Amb un bloc SL-2 i un bloc ADD de $n + 2$ bits
  - Amb $n + 2$ FA's, fent el desplaçament a l'esquerra manualment
    - Exemple per $n = 4$ i $m = n + 3 = 7$

$$\begin{array}{cccccccc} & x_3 & x_2 & x_1 & x_0 & 0 & 0 \\ + & 0 & 0 & x_3 & x_2 & x_1 & x_0 \\ \hline w_6 & w_5 & w_4 & w_3 & w_2 & w_1 & w_0 \end{array} \Rightarrow$$



Multiplicador per 5: simplificació

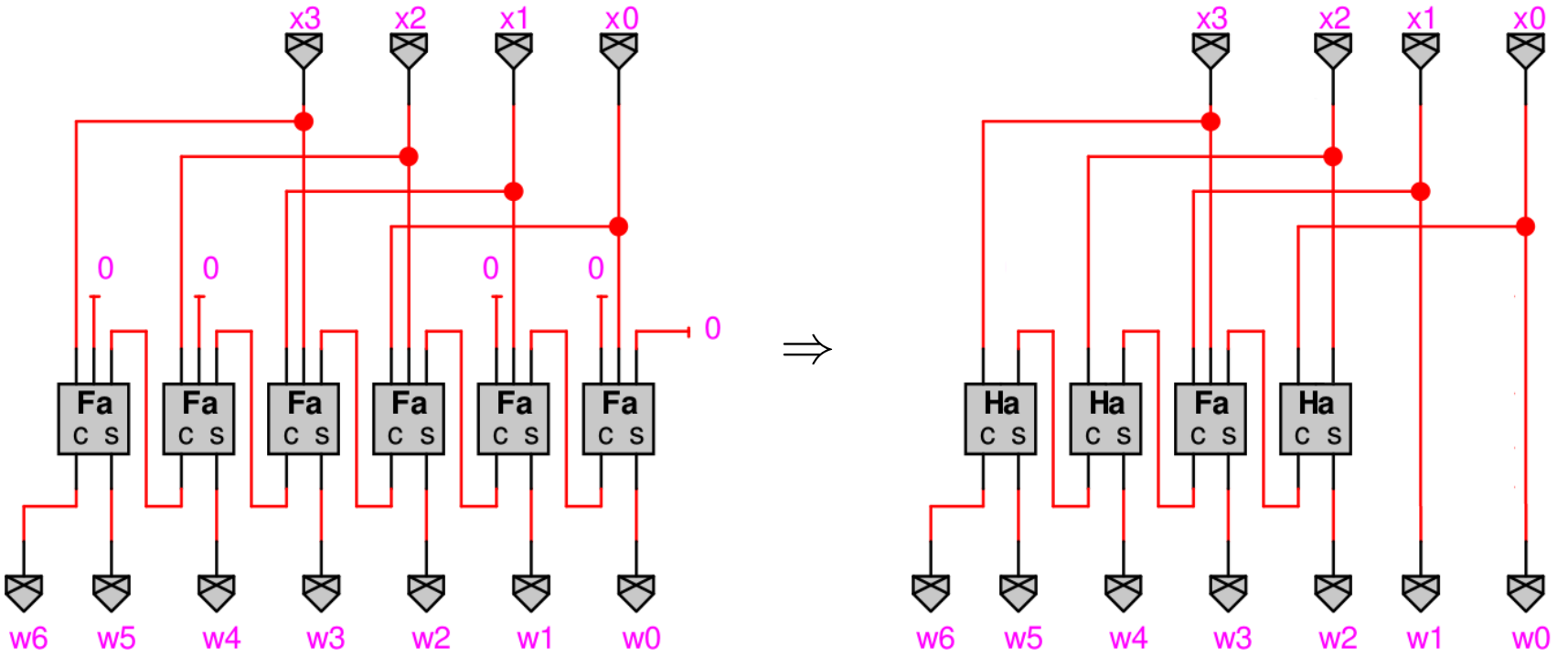
- Analitzant les entrades dels blocs FA, podem simplificar el disseny:



$\Rightarrow$

Multiplicador per 5: simplificació

- Analitzant les entrades dels blocs FA, podem simplificar el disseny:
  - Els FA que calculen $w0$ i $w1$ tenen dues entrades amb el valor "0"
    - Podem connectar directament $x0$ i $x1$ a $w0$ i $w1$ respectivament
  - Els que calculen $w2$, $w4$, $w5$ i $w6$ tenen un operand amb el valor "0"
    - Els podem substituir per blocs *Ha*
  - El que calcula $w3$ és necessari perquè pot haver de sumar tres valors diferents de "0"



Índex

Exercicis (entrega al racó fins dia 4/oct.)

Dibuixeu l'esquema lògic intern dels blocs combinacionals que es demanen a continuació. Per a fer-los, caldrà combinar els blocs vistos anteriorment.

- Valor absolut de X-Y. Bloc que calcula Wu=|Xu-Yu| on X, Y i W són busos de 8 bits.
  - Primer, utilitzant dos restadors SUB de 8 bits, un comparador LEU i un multiplexor de busos de 8 bits.
  - Després, a partir de dos restadors SUB de 8 bits, un comparador LEU, dos blocs AND, un sumador ADD (o un bloc OR), les portes lògiques que siguin necessàries (i sense utilitzar multiplexors).
- Majoria d'uns. Bloc amb una entrada X de 8 bits i sortida d'un sol bit. La sortida val 1 si X conté més 1's que 0's; altrament valdrà 0. Construïu el bloc a partir d'un Half-Adder, dos Full-Adders, dos blocs ADD, un bloc LEU i les portes lògiques que siguin necessàries.
  - Exemple: si l'entrada és 10101010_2 la sortida serà 0 però si l'entrada és 11001111_2 la sortida serà 1.

Índex

Conclusions

- Hem fet dissenys ad-hoc per a blocs aritmètics per a naturals
  - Síntesi basada en descomposició en suma de *minterms* és inviable
- Els blocs aritmètics fixen la mida dels operands d'entrada i sortida
  - A aquesta assignatura $n = 16$, però és comú $n = 32$ o $n = 64$
  - No és possible representar naturals majors o iguals a 2^n
    - L'aritmètica computacional implementa aritmètica modular 2^n
    - El resultat de l'aritmètica computacional pot no coincidir amb el de l'aritmètica convencional
- Alguns blocs tenen una sortida que indica que el resultat de l'aritmètica convencional no és representable amb n bits
  - El bloc ADD té la sortida *c carry* per indicar que el resultat de la suma amb aritmètica convencional requereix $n + 1$ bits
- No oblideu respondre l'ET4 a Atenea i fer autoaprenentatge dels temes indicats a continuació!

Índex

Autoaprenentatge

- Haureu d'estudiar pel vostre compte les següents seccions de la documentació de l'assignatura:
  - 4.3 Restador binari
    - És clau que us adoneu que el resultat de la resta de dos naturals serà no representable com a natural si el resultat de la resta és negatiu i com detectar-ho
  - 4.7 Operadors lògics bit a bit
    - Veureu els operadors AND, OR, XOR i NOT bit a bit.
    - Podreu resoldre *The Hamlet's dilemma*: *To be or not to be* = $0xFF$:-)
  - 4.8 Disseny de multiplexors
    - Construireu multiplexors grans a partir de multiplexors més petits, en particular, construir un Mx8-1 (multiplexor de 8 entrades i 1 sortida) a partir de varis Mx2-1 (multiplexors de 2 entrades i 1 sortida)
  - 4.9 Anàlisi de circuits amb blocs
    - Interpretar els senyals com a nombres naturals i representar la tasca dels blocs algebraicament.

Índex

Manualitats binàries

- Incrementador fet amb fusta
  `https://www.youtube.com/watch?v=zELAfmp3fXY`
- Sumador fet amb fusta i bales
  `https://www.youtube.com/watch?v=GcDshWmhF4A`

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.quantamagazine.org/smaller-is-better-why-finite-number-systems-pack-more-punch-20190211/.

[2] [Online]. Available: https://pixfeeds.com/images/auto/320-489978438-closeup-car-dashboard.jpg.

[3] [Online]. Available: https://boomlyshop.com/collections/clocks-and-watches/products/lucite-11-big-number-wall-clock.

[4] [Online]. Available: https://www.walmart.com/ip/Binary-Math-Computer-Science-Humor-Graphic-Framed-Print-Poster-Wall-or-Desk-Mount-Options/747826982.

[5] [Online]. Available: https://en.wikipedia.org/wiki/Year_2000_problem.

[6] [Online]. Available: https://arstechnica.com/information-technology/2014/12/gangnam-style-overflows-int_max-forces-youtube-to-go-64-bit/.

[7] [Online]. Available: https://en.wikipedia.org/wiki/GPS_Week_Number_Rollover.

[8] [Online]. Available: https://en.wikipedia.org/wiki/IPv4_address_exhaustion.

[9] [Online]. Available: https://www.theguardian.com/world/2020/aug/24/japan-running-out-of-credit-card-numbers-amid-online-shopping-boom.

[10] [Online]. Available: https://en.wikipedia.org/wiki/Year_2038_problem.

[11] [Online]. Available: https://en.wikipedia.org/wiki/NTFS.

[12] [Online]. Available: https://www.matriculasdelmundo.com/espana.html.

Referències II

[13] [Online]. Available: https://www.sport.es/es/noticias/futbol/amano-eldense-apuestas-segundab-mafia-5950546.

[14] [Online]. Available: https://upload.wikimedia.org/wikipedia/en/0/0a/Nadia_Comaneci_1.00.jpg.

Introducció als Computadors

Tema 4: Blocs aritmètics combinacionals per a naturals

http://personals.ac.upc.edu/enricm/Docencia/IC/IC4.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.