Introducció als Computadors

Tema 7: Processadors de Propòsit Específic (PPE's)

http://personals.ac.upc.edu/enricm/Docencia/IC/IC7a.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

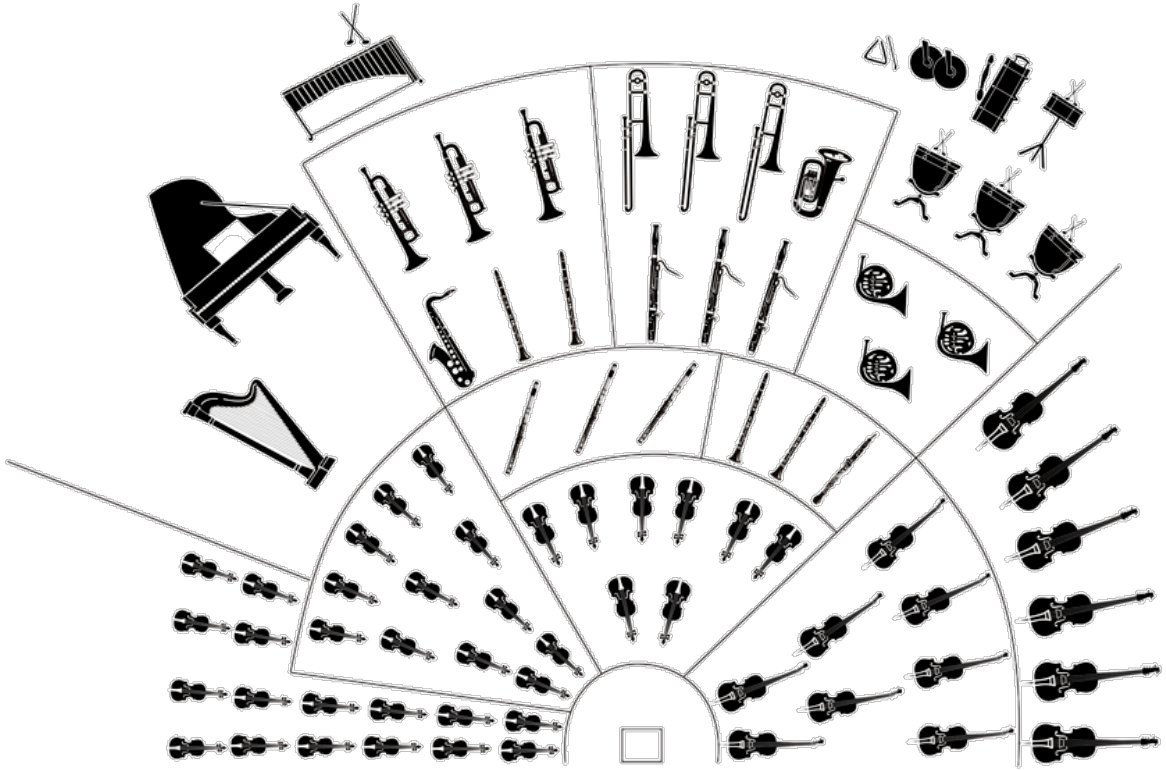
2020-21, 1^{er} quad.

PPE's dividits en UC i UP

Unitat de Control (UC)

[1]

Unitat de Procés (UP)



[2]

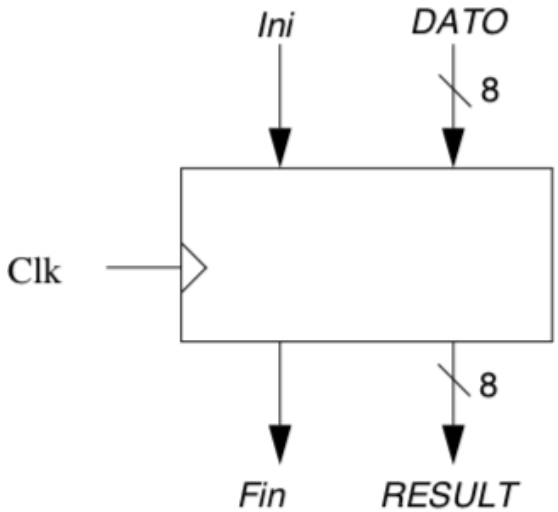
Índex

Processadors de propòsit específic (PPE's)

- Tractarem CLS's de major complexitat que els fets fins ara
  - PPE's: Processadors de propòsit específic
  - Processaran dades típicament de $n = 8$ o $n = 16$ bits
    - Fer el graf d'estats és inviable
  - Anàleg al que vàrem fer amb els CLC's
- Separarem la funcionalitat dels PPE's en dos CLS's
  - Unitat de Procés (UP)
    - Processarà les dades de n bits
    - També anomenat *data path*
    - Disseny ad-hoc
  - Unitat de Control (UC)
    - Generarà els senyals de control per a l'UP:
      Senyals de selecció per a multiplexors, senyals de càrrega per als registres, ...
    - També s'encarregarà dels senyals d'entrada/sortida de control del PPE
    - Tindrà pocs estats
    - Serà similar als CLS's del tema anterior

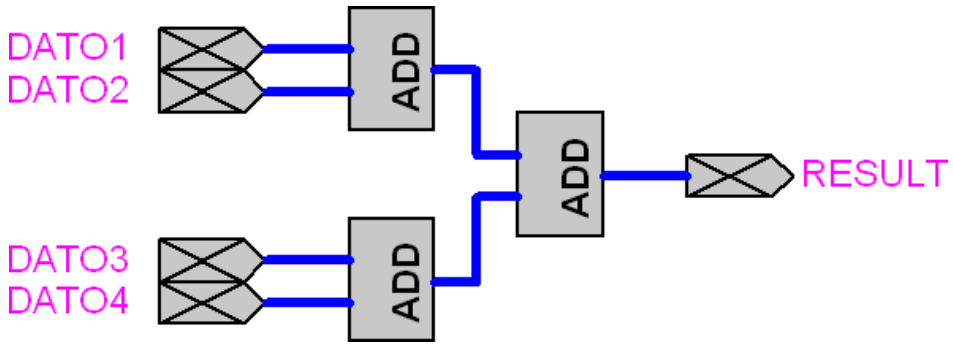
Enunciat d'exemple Suma-4

- Especificació:
  - Dissenyar un PPE que realitzi la suma modular mòdul 2^8 d'una seqüència de quatre nombres naturals codificats amb 8 bits.
  - Els nombres a sumar arriben per l'entrada *DATO* de 8 bits a raó d'un nombre per cicle, començant pel cicle on el senyal d'entrada *Ini* val "1".
  - Un cop terminat el càlcul, el resultat estarà disponible a la sortida de 8 bits *RESULT* durant un cicle, en el qual la sortida *Fin* valdrà "1"
    - El senyal *RESULT* només és significatiu quan *Fin* val "1"
- Encapsulat i cronograma simplificat d'exemple



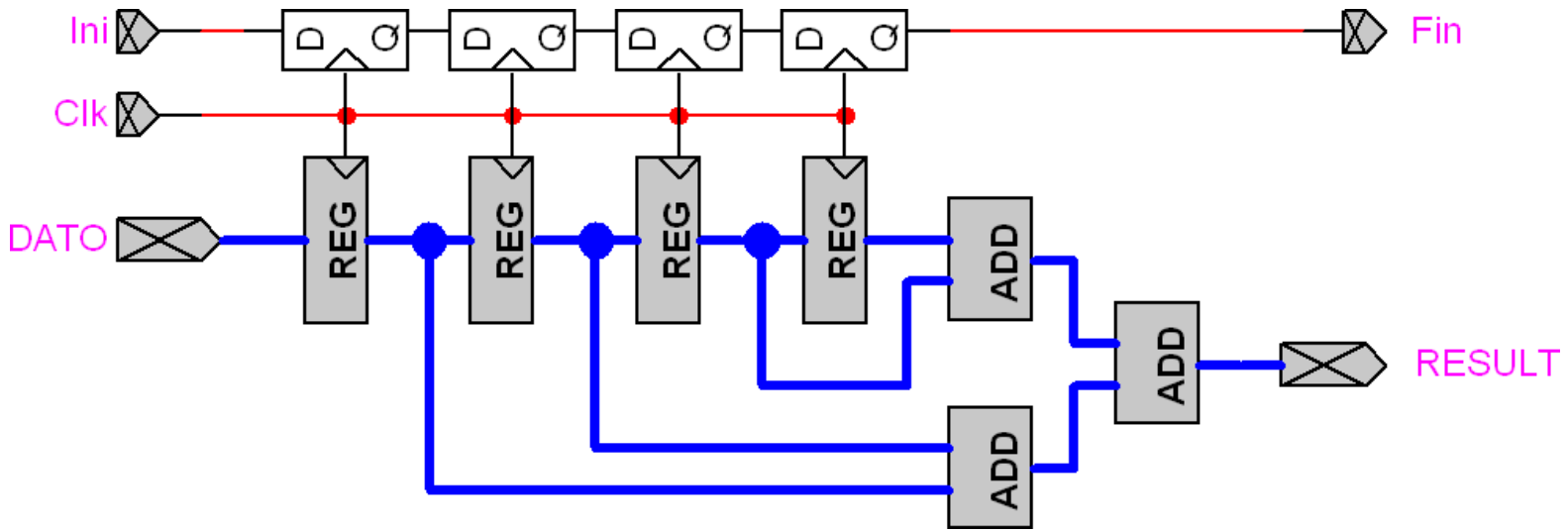
Cicle	0	1	2	3	4	5	6
Ini	0	1	0	0	0	0	0
DATO (dec)	x	23	5	12	18	x	x
RESULT (dec)	x	x	x	x	x	58	x
Fin	0	0	0	0	0	1	0

Primera aproximació a Suma-4



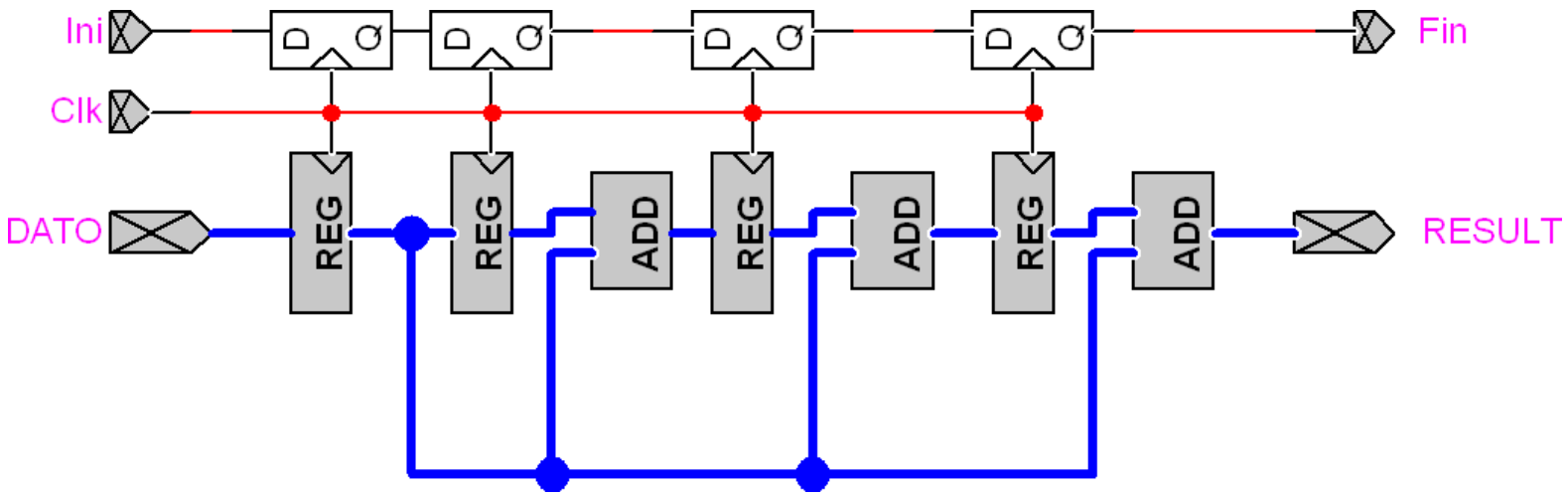
- Implementat com a CLC:
  - Però ens han demanat un CLS amb una temporització determinada!
  - El nombre d'entrades tampoc s'ajusta al demanat
- **No compleix l'especificació**

Segona aproximació a Suma-4



- Utilitza quatre biestables i quatre registres per retenir les entrades dels quatre darrers cicles i 3 sumadors per sumar-les de cop
  - *Shift register*
  - Compleix l'especificació funcional
  - Podria ser vàlid per sumar 4 nombres però, i per sumar-ne $N = 100$?
    - El T_c es dispararia (la solució no és escalable)
    - El determina el camí b-s. Un sumador en arbre tindria $\lceil log_2 N \rceil$ nivells
- Tampoc la considerem vàlida

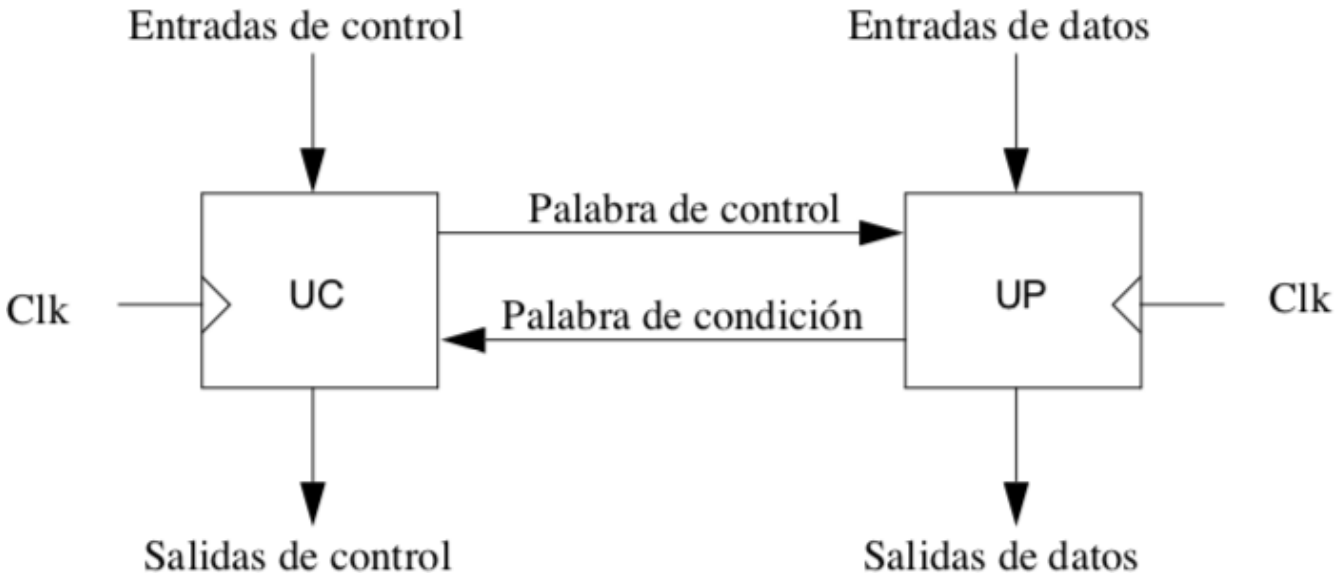
Tercera aproximació a Suma-4



- Utilitza quatre biestables i quatre registres per retenir les entrades dels quatre darrers cicles i i 3 sumadors fer sumes parcials cada cicle
  - Compleix l'especificació funcional
  - El T_p dels camis està més equilibrat que abans
  - Però l'ús de recursos segueix sent molt alt
    - Suma-N necessitaria $N-1$ sumadors, N registres i N biestables
- Tampoc la considerem vàlida
  - Sumarem les dades a mesura que es rebin, sense esperar tenir-les totes
    - El PPE només utilitzarà un bloc sumador i un registre

Índex

PPE dividit en UC i UP



- **Cada cicle**: comunicació entre UC i UP
  - La UC genera, a partir del seu estat actual, les sortides de control i la "paraula de control" que governarà la UP
    - Exemple: senyals de selecció dels multiplexors
  - La UP fa el seu processament, genera sortides de dades i retorna a la UC una "paraula de condició" que notifica esdeveniments rellevants
    - Exemple: que el resultat d'una operació ha estat 0
  - La UC calcula el seu estat futur a partir del seu estat actual, la "paraula de condició" i les entrades de control del PPE.

Unitat de Procés (UP) (*data path*)

- Estarà format pels blocs seqüencials (registres, biestables) interconnectats amb els blocs combinacionals (sumadors, comparadors, shifters, multiplexors, portes lògiques,...) necessaris per a realitzar el processament a implementar
- La UP enviarà a la UC la informació necessària perquè aquesta pugui prendre les decisions relatives al control per al cicle següent
  - Paraula de condició
  - "Avaluació de bucles i estructures condicionals"
- Entrades de la UP:
  - Entrades de dades del PPE
  - Paraula de control generada per la UC
- Sortides de la UP:
  - Sortides de dades del PPE
  - Paraula de condició

UP de Suma-4

- Podem expressar la tasca del PPE com a pseucodi:

```
if (Ini(t) == 1) {
    RESULT(t+4) = ∑_{i=0}^{3} DATO(t+i) mod 2^8;
    Fin(t+4) = 1;
}
```

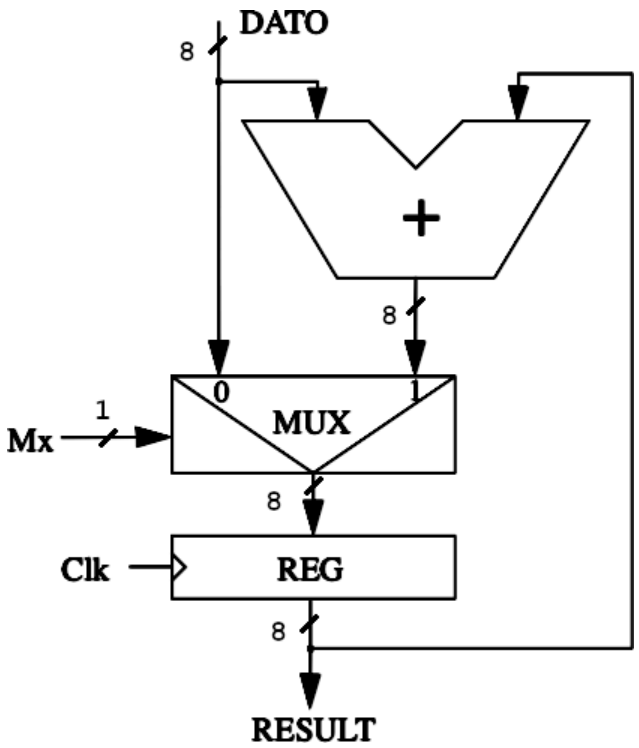
- Si ens centrem en el processament de les dades:

$$
\begin{aligned}
&tmp = DATO(t);\\
&tmp = (tmp + DATO(t+1)) \mod 2^8;\\
&tmp = (tmp + DATO(t+2)) \mod 2^8;\\
&tmp = (tmp + DATO(t+3)) \mod 2^8;\\
&RESULT(t+4) = tmp;
\end{aligned}
$$

  - La variable *tmp* haurà d'emmagatzemar-se a un registre
  - Necessitarem un sumador
  - *tmp* s'ha de poder actualitzar des de *DATO* o des del sumador
    - Multiplexor a l'entrada de REG amb un senyal de control
  - Faltarà generar el senyal *Fin* i el senyal de control del multiplexor com toca, però això és feina de la UC

UP de Suma-4

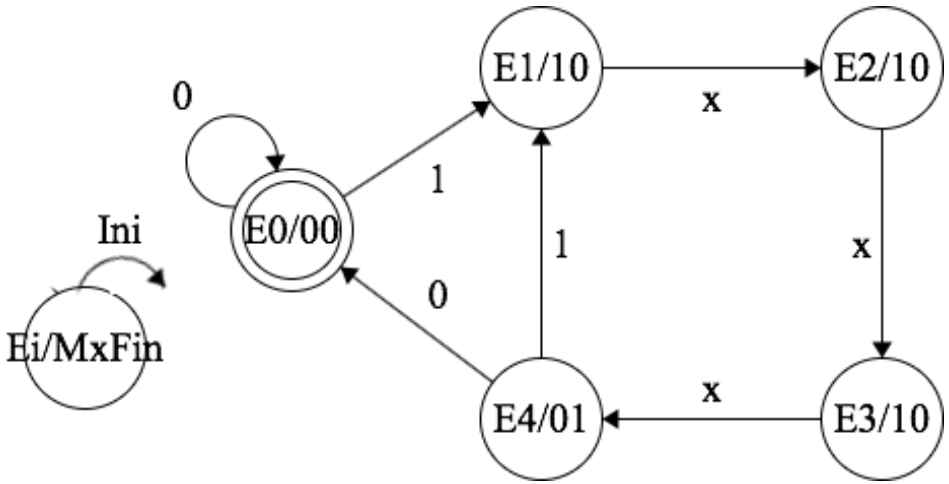
- Necessita:
  - Un registre per a emmagatzemar el valor acumulat fins ara de la suma
  - Un bloc sumador
  - Un multiplexor per a tractar el primer nombre
- Es mostra un possible disseny de la UP
  - La UC haurà de generar el senyal de control *Mx* (i *Fin*)



Unitat de Control (UC)

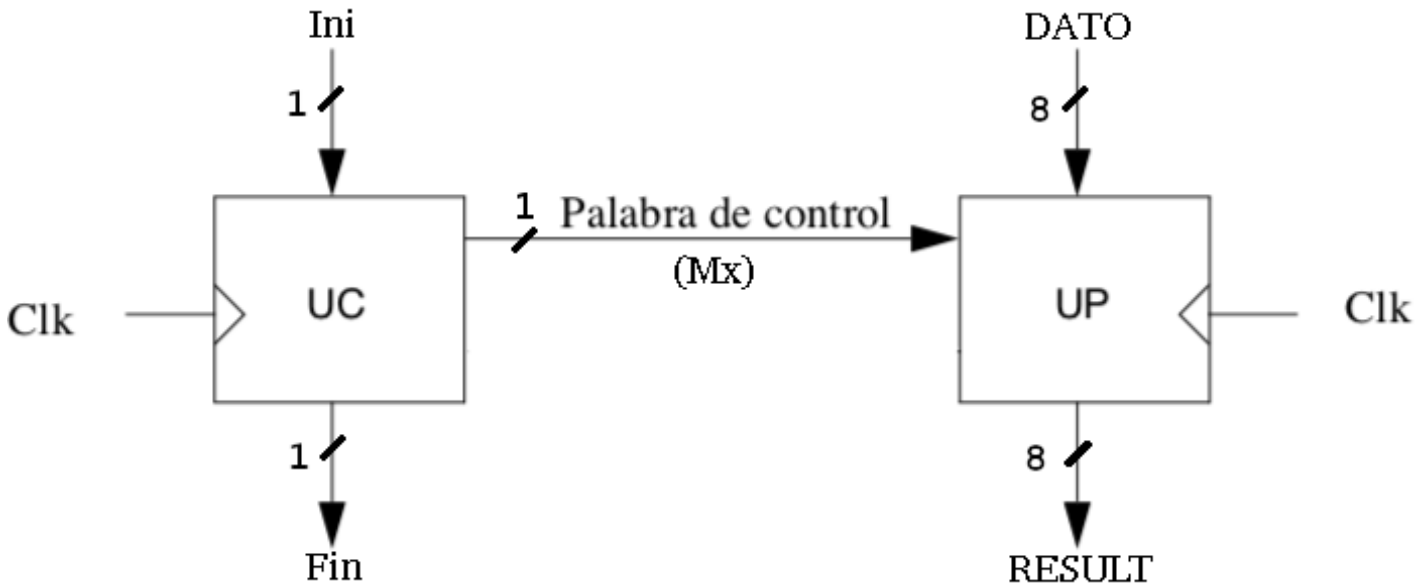
- CLS amb pocs senyals d'entrada i de sortida
  - Es podrà fer el graf d'estats com al tema anterior
  - Model de Moore
- Genera la "paraula de control" per governar la UP
- Senyals d'entrada de la UC:
  - Senyals de control del PPE
    - Típicament, validen les dades al bus d'entrada del PPE
  - "Paraula de condició" generada per la UP
- Senyals de sortida de la UC:
  - "Paraula de control" per a la UP
  - Senyals de control del PPE
    - Típicament, validen les dades al bus de sortida del PPE

- Serà un CLS amb una entrada (*Ini*) i dues sortides (*Fin* i *Mx*)
  - La paraula de control és un únic bit (*Mx*)
  - No hi ha paraula de condició
- Considerem cinc estats:
  - E0: carrega a registre *DATO*
  - E1: carrega a registre la suma de *DATO* (2^{on} operand) amb registre
  - E2: carrega a registre la suma de *DATO* (3^{er} operand) amb registre
  - E3: carrega a registre la suma de *DATO* (4^{rt} operand) amb registre
  - E4: valida *RESULT* (posa Fin="1") i carrega a registre *DATO*
- E0 i E4 carreguen *DATO* al registre per si al final del cicle *Ini*="1"



UC + UP de Suma-4

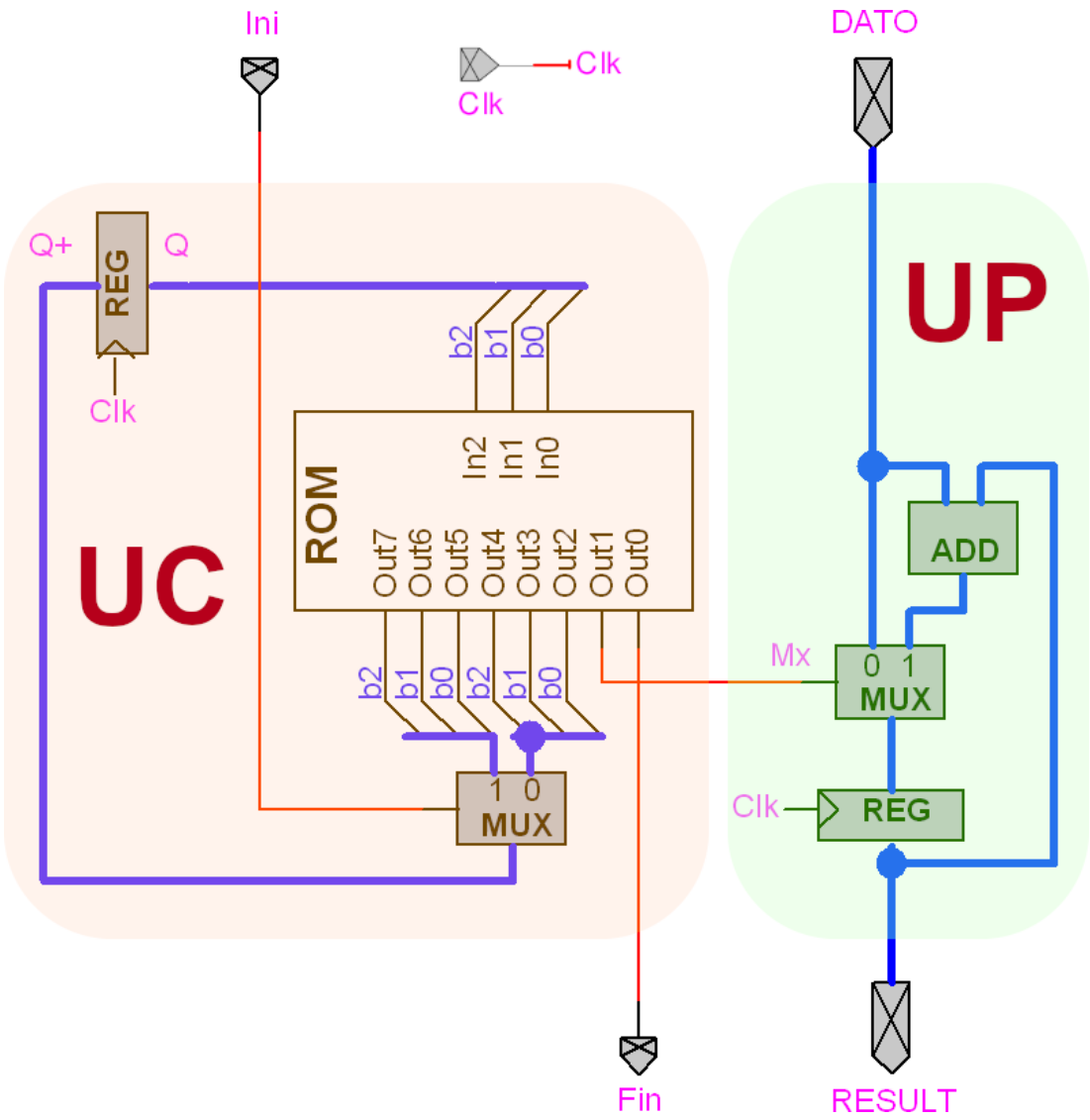
Cicle	0	1	2	3	4	5	6
Ini	0	1	0	0	0	0	0
DATO (dec)	10	23	5	12	18	30	x
Estat UC	E0	E0	E1	E2	E3	E4	E0
Mx	0	0	1	1	1	0	0
RESULT (dec)	x	10	23	28	40	58	30
Fin	0	0	0	0	0	1	0



Esquema lògic complet SUMA-4

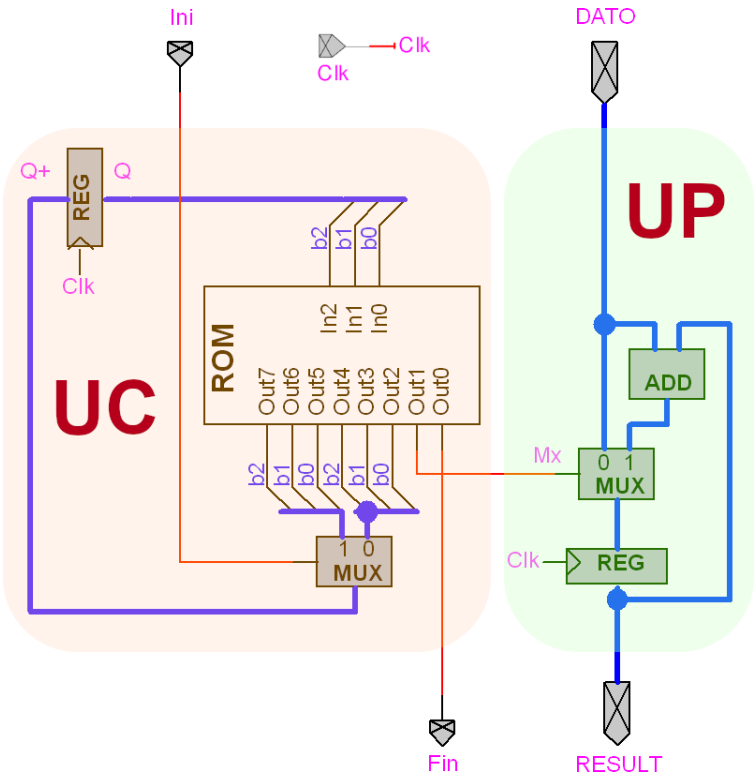
- Parametrització UC:
  - n=1 (*Ini*)
  - m=2 (*Fin*, *Mx*)
  - k=$\lceil log_2 5 \rceil = 3$
  - Mida ROM: 64 bits
    - $2^k \cdot (2^n \cdot k + m)$
  - Contingut ROM:
    - Verifiqueu-lo!

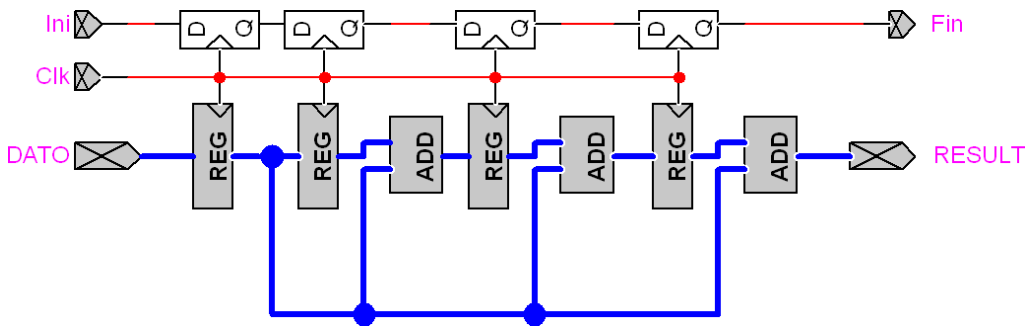
In			Out							
2	1	0	7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	0	0	0	0
0	0	1	0	1	0	0	1	0	1	0
0	1	0	0	1	1	0	1	1	1	0
0	1	1	1	0	0	1	0	0	1	0
1	0	0	0	0	1	0	0	0	0	1
1	0	1	x	x	x	x	x	x	x	x
1	1	0	x	x	x	x	x	x	x	x
1	1	1	x	x	x	x	x	x	x	x



- També es podrien encapsular UC i UP

SUMA-4 versus tercera aproximació





- Com processaria cada implementació aquest cronograma?

Cicle	0	1	2	3	4	5	6	7	8
Ini	0	1	0	1	0	0	0	0	0
DATO (dec)	x	23	5	12	18	22	7	x	x
RESULT (dec)									
Fin									

Índex

PPE Suma-N

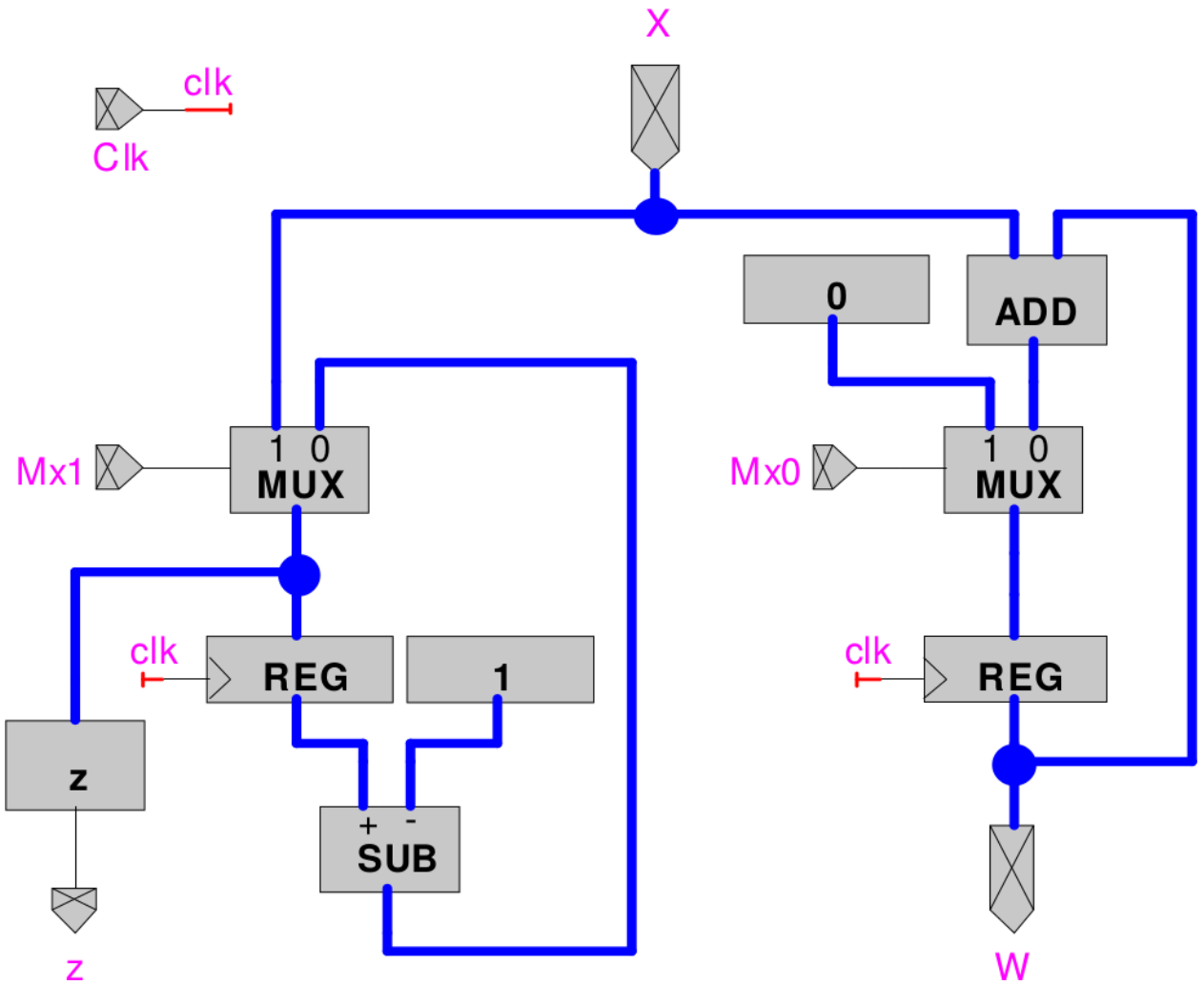
- Dissenyarem un PPE per a calcular la suma mòdul 2^8 de N nombres
- El PPE tindrà els senyals de control *Inici* i *Fi*, l'entrada de dades X i la sortida de dades W.
  - Primer rebrem el valor N per l'entrada X
    - Assumim $N \geq 1$
  - A continuació rebrem els N nombres per X, a raó d'un per cicle.
  - La suma s'ha de mostrar per W
- Veurem tres versions del PPE
  - La UP serà la mateixa als tres casos
  - La UC serà pròpia de cada cas
    - Es diferenciaran en la interpretació dels senyals de control
    - Sincronisme entre senyals de control del PPE i els busos d'entrada/sortida

Algorisme de Suma-N


```
num = X(t); // X(t) = N, la quantitat de nombres a sumar
suma = 0;
while (num > 0) {
    suma = suma + X(t+N+1-num); // X(t+1) -> primer nombre
    num = num - 1;
}
W = suma;
```

- Què necessita la UP per poder implementar l'algorisme?
  - Dos registres:
    - Un per emmagatzemar la suma i un altre per l'iterador del bucle
  - Dos blocs aritmètics
    - Per actualitzar la suma i el nombre d'iteracions
  - Un bloc z, perquè la UC sigui conscient del final del bucle
  - Multiplexors per poder triar entre inicialitzar o actualitzar els registres

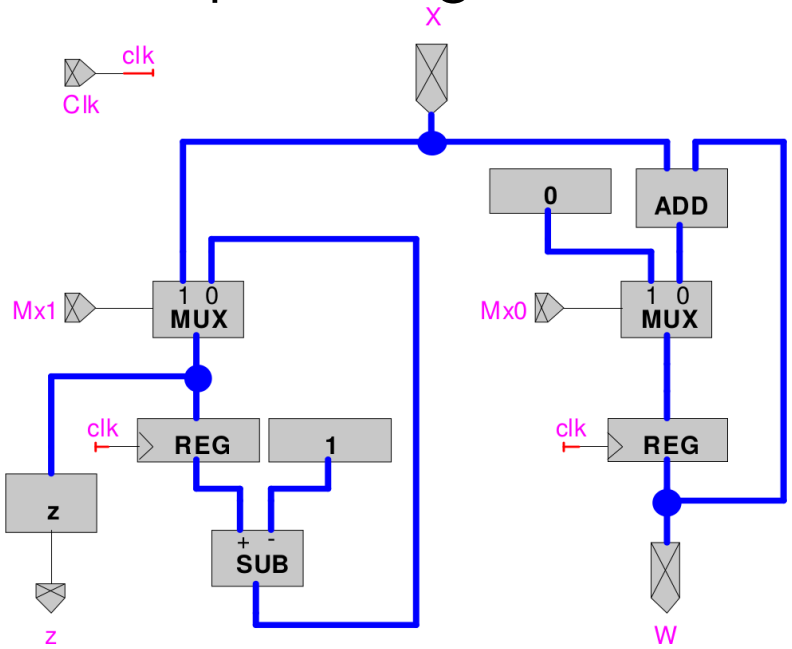
UP de Suma-N

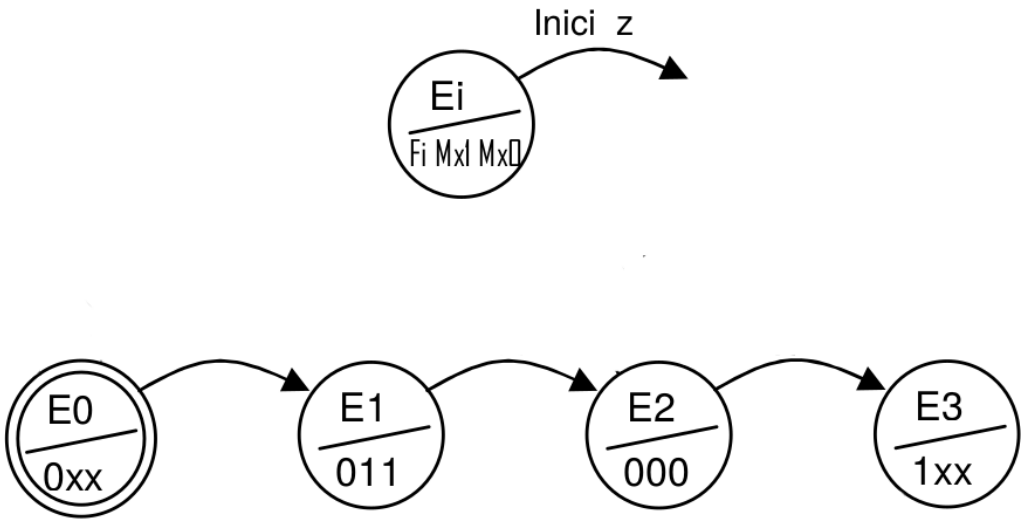


- La paraula de condició estarà formada pel senyal z
  - $z = 1 \iff$ el valor a l'entrada del registre és 0
- La paraula de control estarà formada pels senyals $Mx0$ i $Mx1$

Graf d'estats de la UC de Suma-*N* v1.0

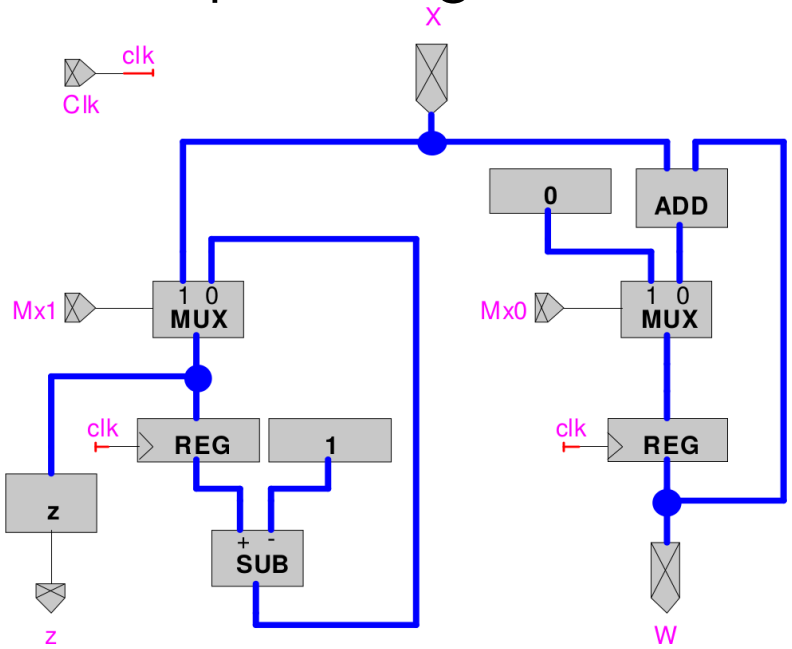
- Si Inici(c)=1 llavors X(c+1)=N, X(c+2)...X(c+N+1) són *N* nombres
  - W(c+N+2)=la suma i Fi(c+N+2)=1
  - *Fi* val 0 la resta de cicles
- Ignorarem el valor de *Inici* durant tot el càlcul, des de c+1 a c+N+2.
- Completeu el graf amb les etiquetes i transicions que falten

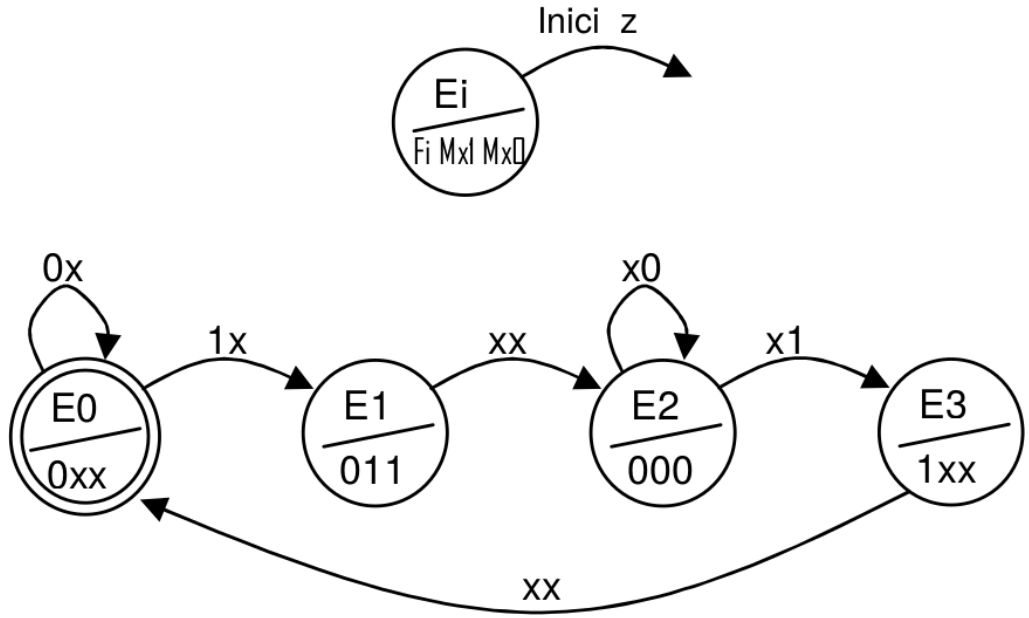




Graf d'estats de la UC de Suma-*N* v1.0

- Si Inici(c)=1 llavors X(c+1)=N, X(c+2)...X(c+N+1) són *N* nombres
  - W(c+N+2)=la suma i Fi(c+N+2)=1
  - *Fi* val 0 la resta de cicles
- Ignorarem el valor de *Inici* durant tot el càlcul, des de c+1 a c+N+2.
- Completeu el graf amb les etiquetes i transicions que falten

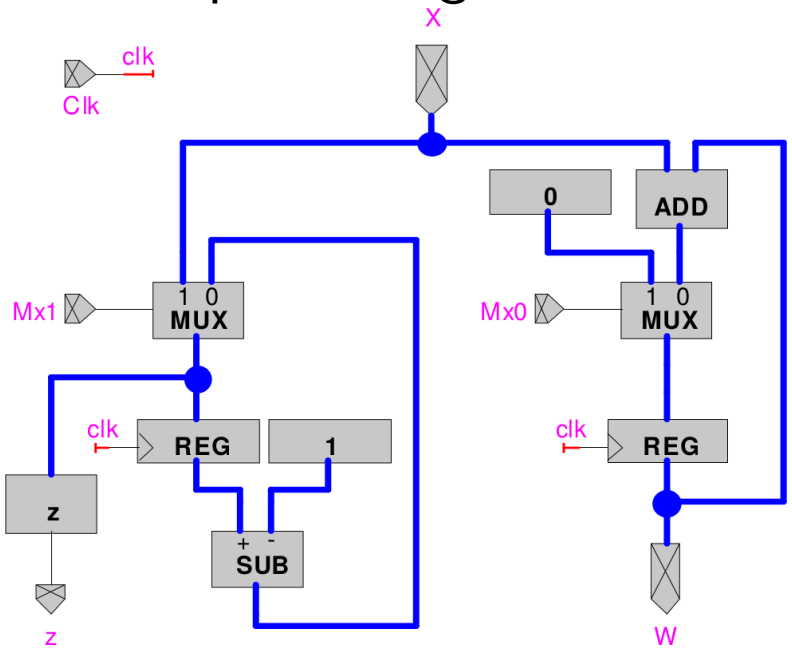


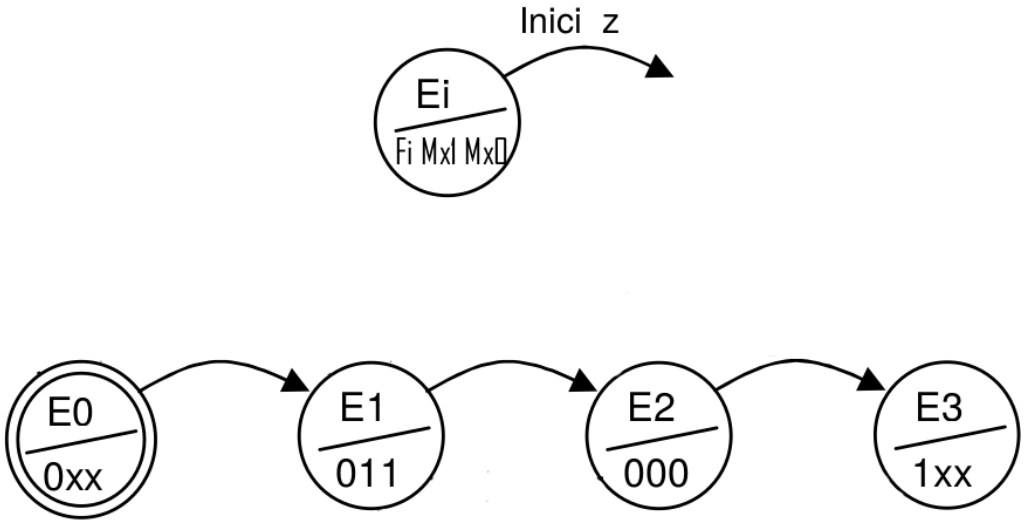


- Estats: E0: espera, E1: inicialitza regs., E2: bucle de càlcul, E3: presenta resultat

Graf d'estats de la UC de Suma-*N* v2.0

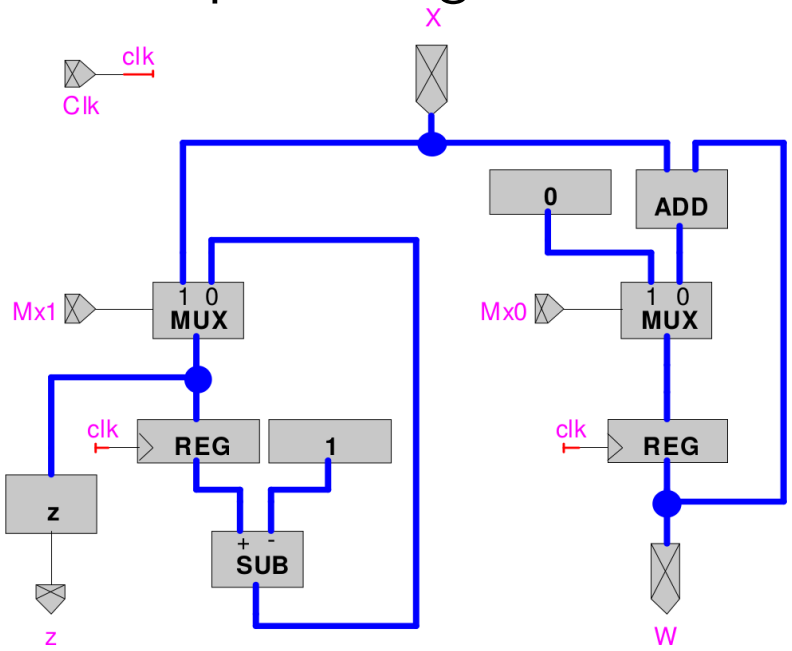
- Ara no ignorarem el valor de *Inici* mentre es faci el càlcul.
  - Si *Inici* val 1 en algun cicle entre c+1 i c+N+2, avortarem el càlcul.
- Completeu el graf amb les etiquetes i transicions que falten

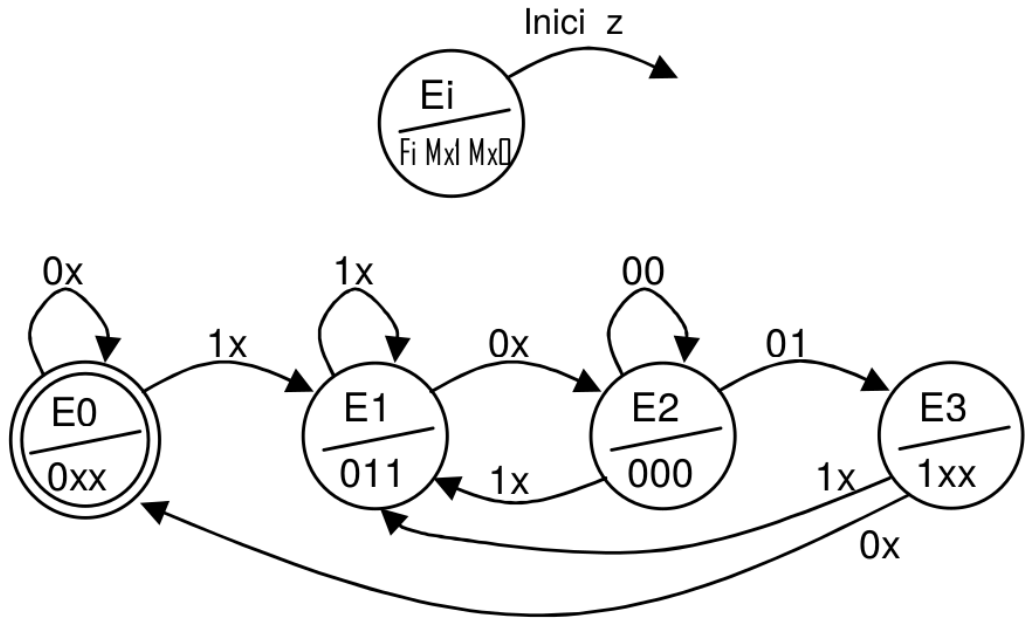




Graf d'estats de la UC de Suma-*N* v2.0

- Ara no ignorarem el valor de *Inici* mentre es faci el càlcul.
  - Si *Inici* val 1 en algun cicle entre c+1 i c+N+2, avortarem el càlcul.
- Completeu el graf amb les etiquetes i transicions que falten

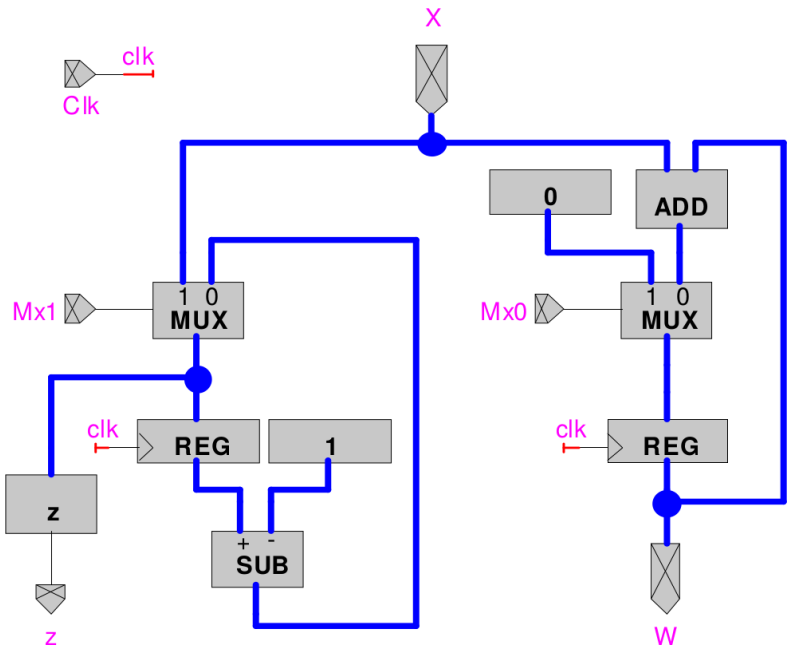


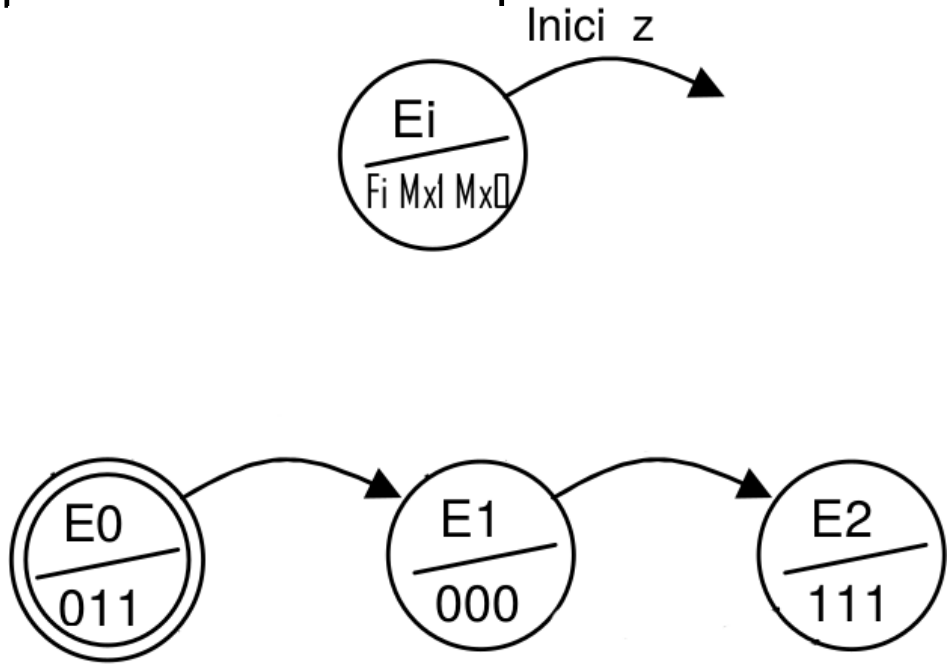


- Estats: E0: espera, E1: inicialitza regs., E2: bucle de càlcul, E3: presenta resultat

Graf d'estats de la UC de Suma-*N* v3.0

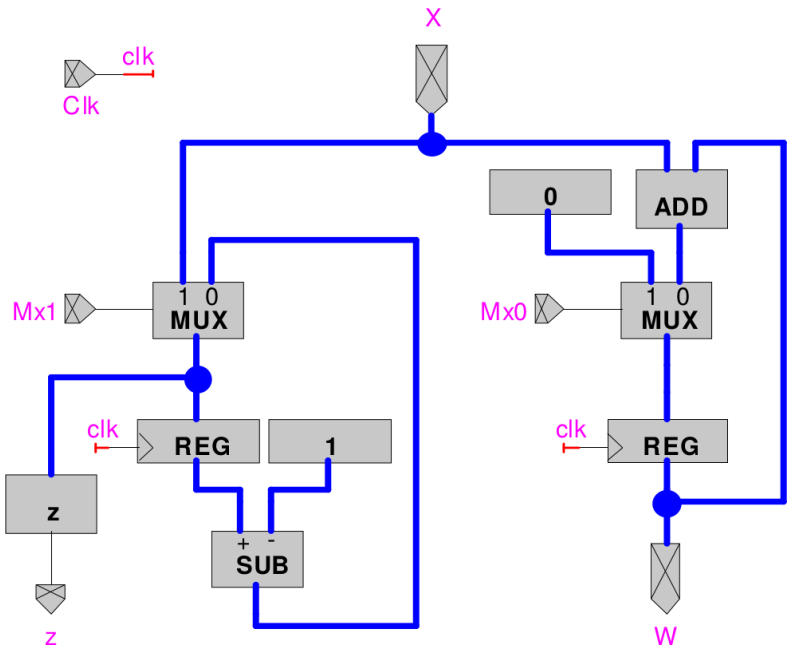
- Ara rebrem el valor N al cicle *c*, (en paral·lel amb *Inici*=1)
  - El resultat es mostrarà al cicle c+N+1
- Ignorarem el valor d'*Inici* durant tot el càlcul
- Completeu el graf amb les etiquetes i transicions que falten

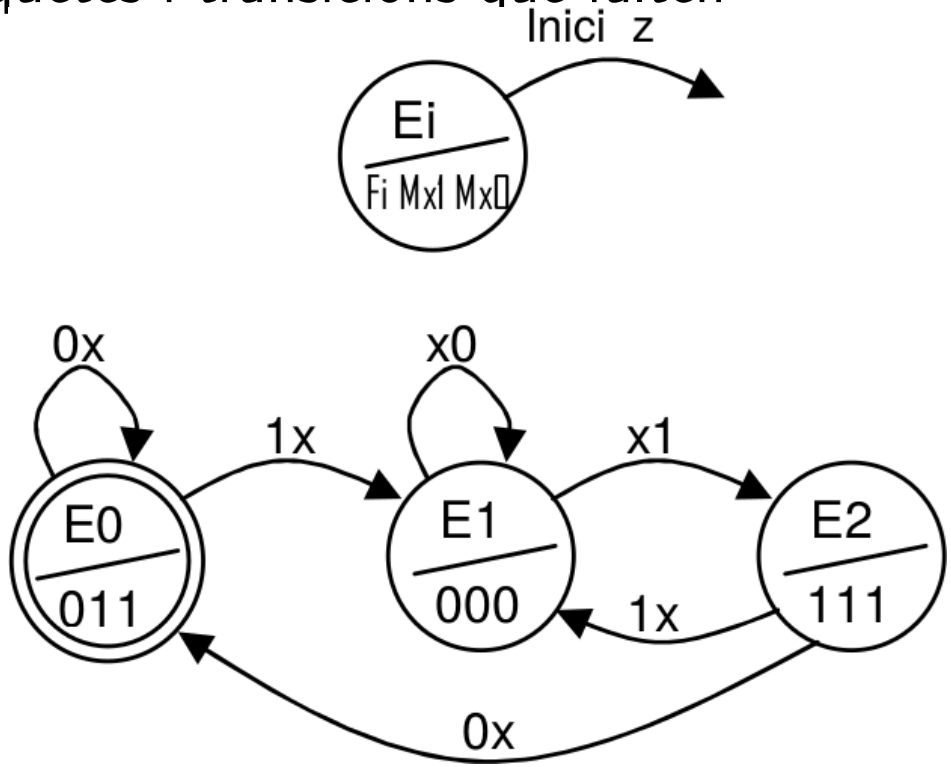




Graf d'estats de la UC de Suma-*N* v3.0

- Ara rebrem el valor N al cicle *c*, (en paral·lel amb *Inici*=1)
  - El resultat es mostrarà al cicle c+N+1
- Ignorarem el valor d'*Inici* durant tot el càlcul
- Completeu el graf amb les etiquetes i transicions que falten

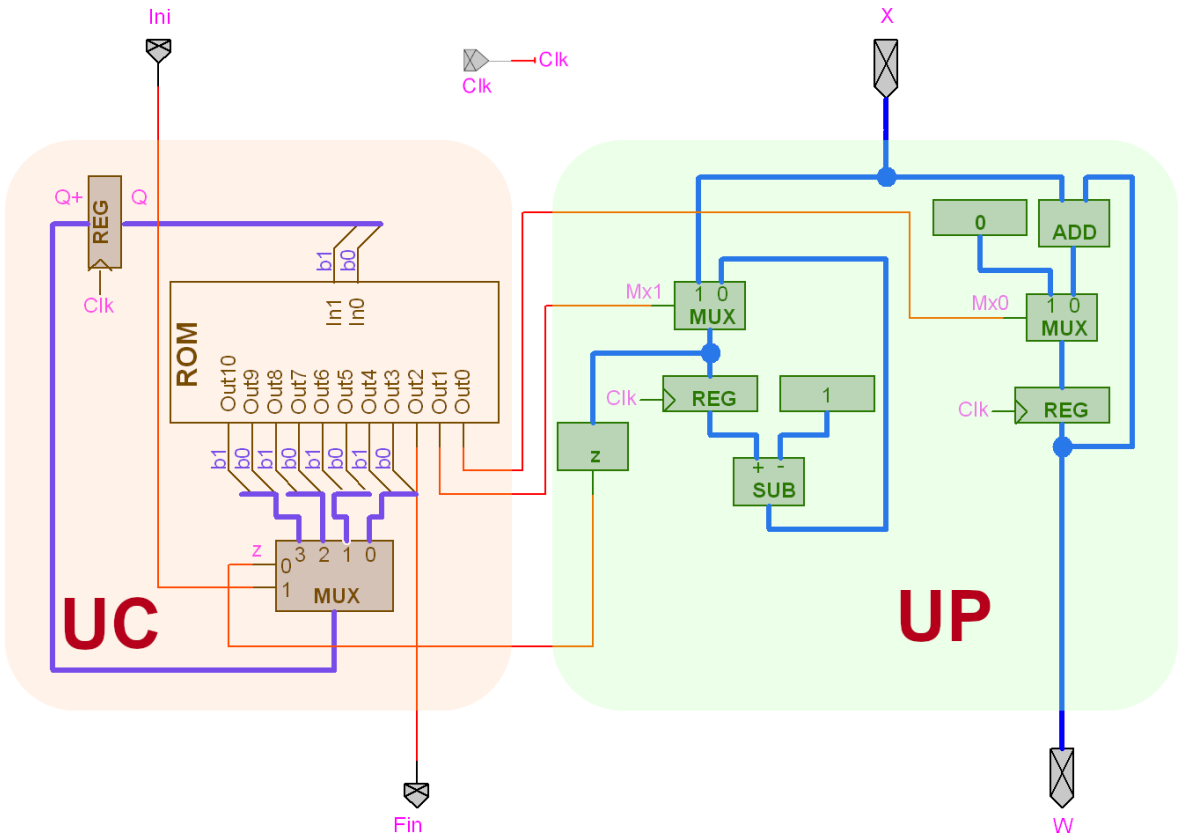




- Estats: E0: espera i inicialitza regs., E1: bucle de càlcul, E2: presenta resultat

Esquema lògic complet Suma-N

- Parametrització UC
  - n=2 (Ini, z)
  - m=3 $(Fi, Mx1, Mx0)$
  - k=$\lceil log_2 4 \rceil = 2$
  - mida ROM: 44 bits
    - 4 paraules (2^k)
    - 11 bits/paraula $(2^n \cdot k + m)$



- El contingut de la ROM (*programa?*) determina la versió de Suma-N

In	Out (v1.0)										
1 0	10	9	8	7	6	5	4	3	2	1	0
0 0	0	1	0	1	0	0	0	0	x	x	
0 1	1	0	1	0	1	0	1	0	0	1	1
1 0	1	1	1	0	1	1	1	0	0	0	0
1 1	0	0	0	0	0	0	0	1	x	x	

In	Out (v2.0)										
1 0	10	9	8	7	6	5	4	3	2	1	0
0 0	0	1	0	1	0	0	0	0	x	x	
0 1	0	1	0	1	1	0	1	0	0	1	1
1 0	0	1	0	1	1	1	1	0	0	0	0
1 1	0	1	0	1	0	0	0	1	x	x	

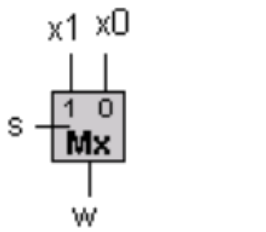
In	Out (v3.0)										
1 0	10	9	8	7	6	5	4	3	2	1	0
0 0	0	1	0	1	0	0	0	0	1	1	
0 1	1	0	0	1	1	0	0	1	0	0	0
1 0	0	1	0	1	0	0	0	1	1	1	
1 1	x	x	x	x	x	x	x	x	x	x	

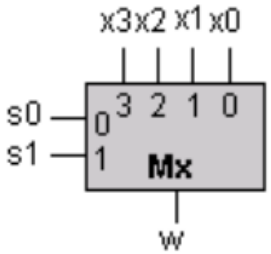
Observacions

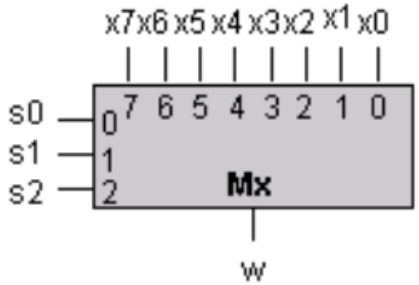
- La UC de Suma-4 té més estats que les de Suma-N !!
  - La UC de Suma-4 incorpora dins del seu estat la informació de quants operands s'han rebut
  - Les UCs de Suma-N deleguen mantenir aquesta informació a la UP de Suma-N
    - La UP notificarà a la UC que ja s'han rebut tots els operands
- Els senyals de control $Mx0$ i $Mx1$ es poden fusionar en un de sol perquè a cada cicle tenen el mateix valor
  - La ROM passaria a ser de 40 bits

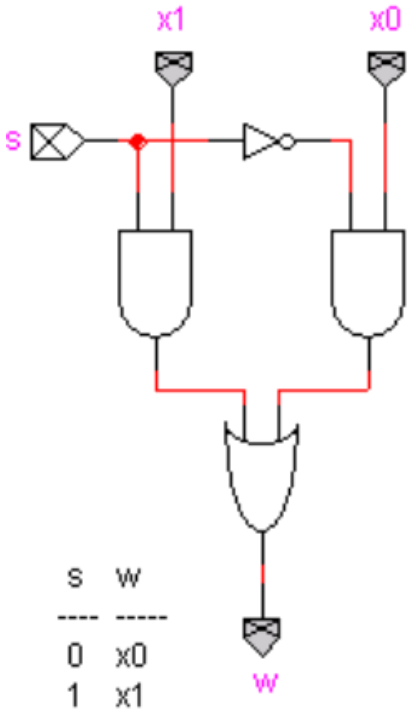
Índex

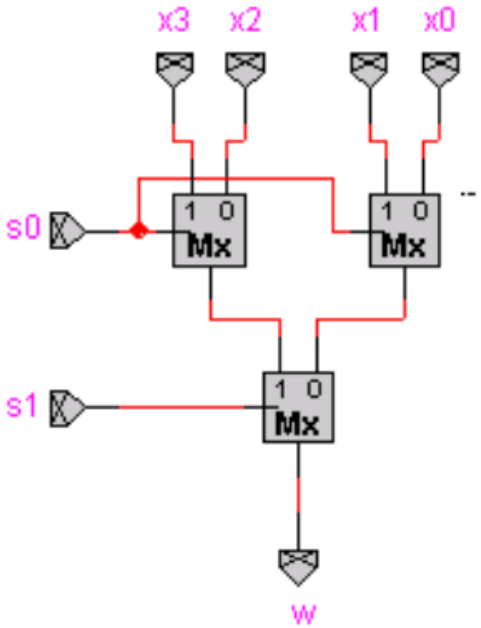
Multiplexors Mx-2-1, Mx-4-1, Mx-8-1

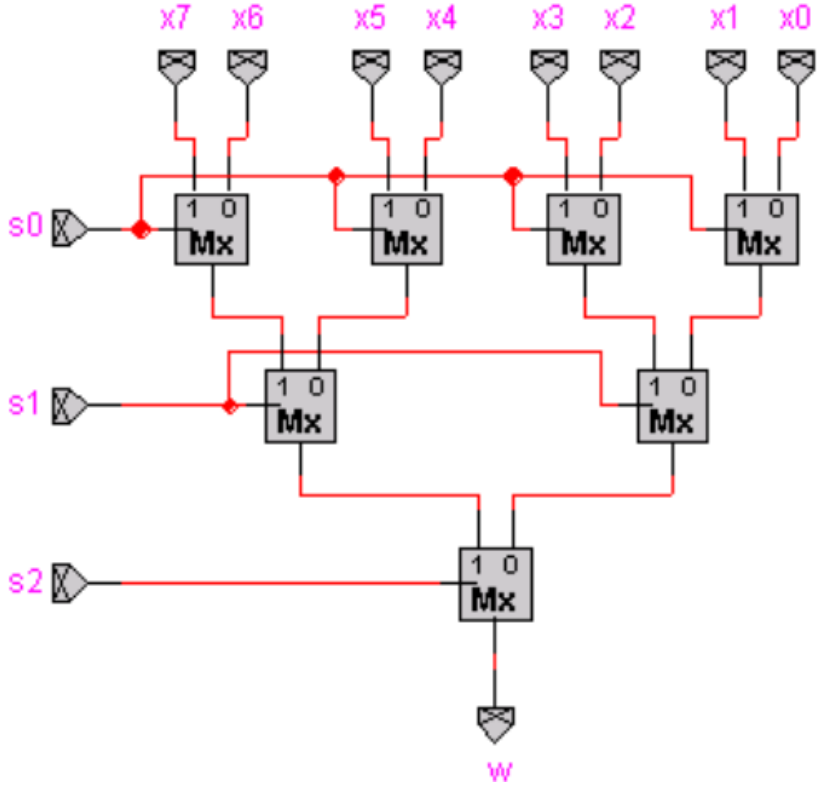






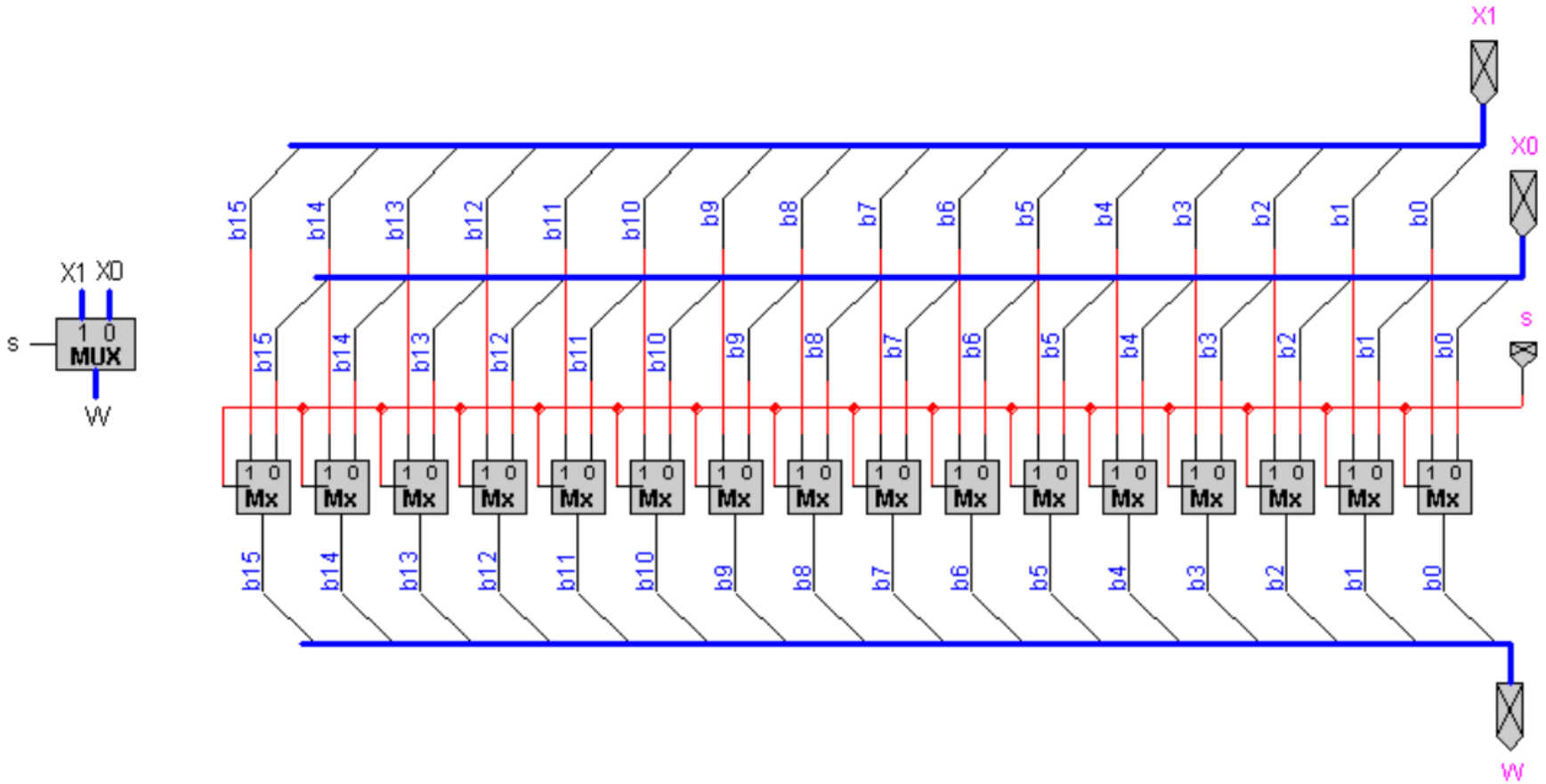






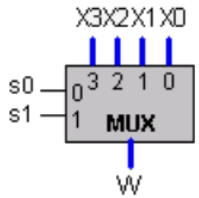
Multiplexors de bussos de 16 bits

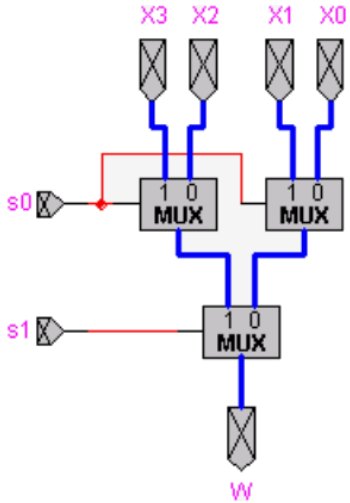
- MUX 2-1



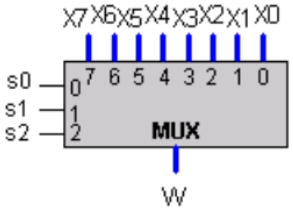
Multiplexors de bussos de 16 bits

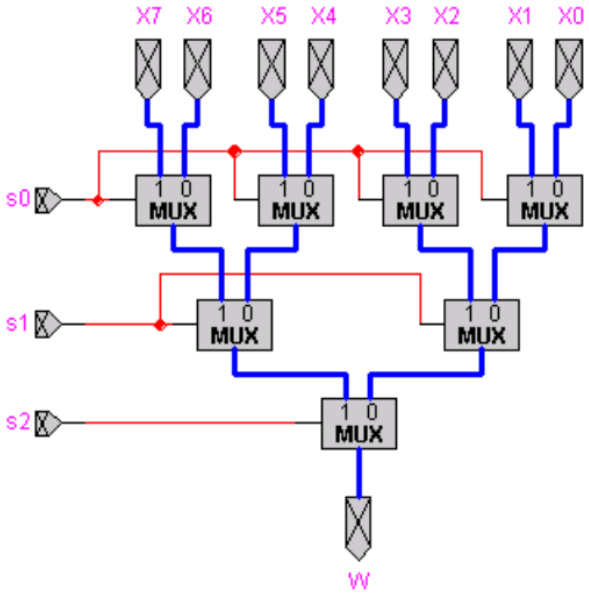
- MUX 4-1





- MUX 8-1

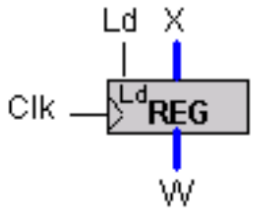


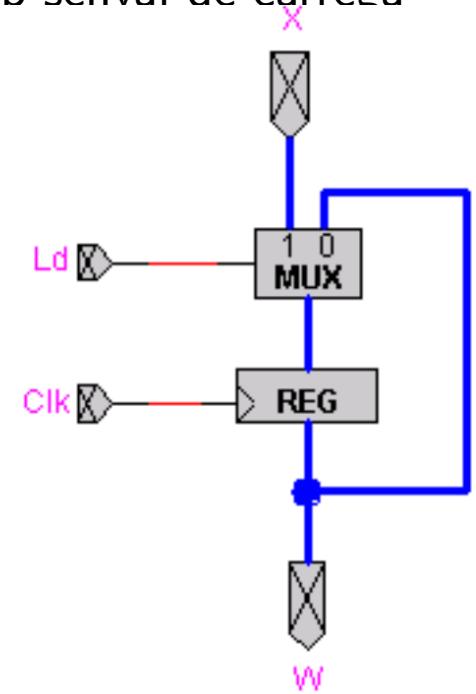


Registre amb senyal de càrrega

- Els registres actualitzen cada cicle el valor que s'hi emmagatzema
  - Alguns CLS's necessitaran retenir els valors varis cicles
    - Utilitzaran un bloc anomenat registre amb senyal de càrrega

```
if (Ld(t)==1)
  W(t+1) = X(t);
else
  W(t+1) = W(t);
```

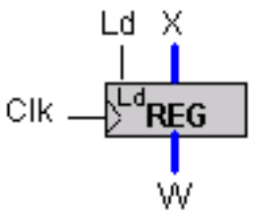


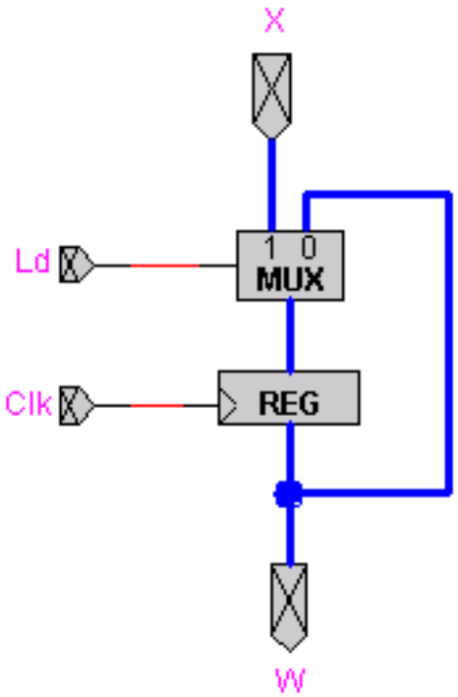


- Al calcular el T_p d'un camí que acabi a un registre amb senyal de càrrega, cal travessar el multiplexor abans d'arribar als biestables
- Es podria implementar sense fer servir el multiplexor i fent que l'entrada *Clk* del REG sigui el resultat d'una AND-2 de *Clk* i *Ld*?

Registre amb senyal de càrrega

```
if (Ld(t)==1)
  W(t+1) = X(t);
else
  W(t+1) = W(t);
```



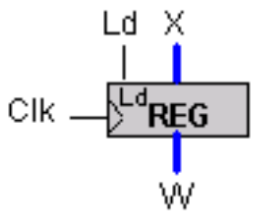


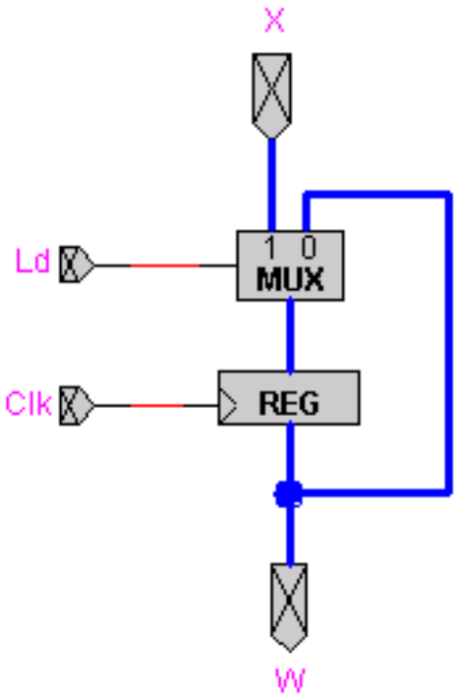
- Ompliu el cronograma amb el comportament d'un registre amb càrrega

Cicle	0	1	2	3	4	5	6
X	0x12	0x14	0x26	0x43	0x90	0x33	0x87
Ld	0	1	0	0	1	1	0
W	0x27						

Registre amb senyal de càrrega


```
if (Ld(t)==1)
  W(t+1) = X(t);
else
  W(t+1) = W(t);
```





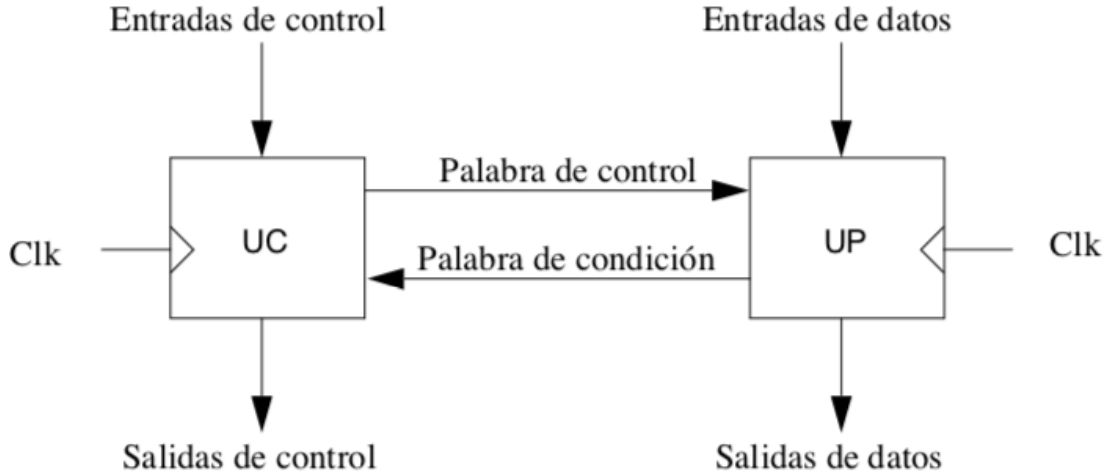
- Ompliu el cronograma amb el comportament d'un registre amb càrrega

Cicle	0	1	2	3	4	5	6
X	0x12	0x14	0x26	0x43	0x90	0x33	0x87
Ld	0	1	0	0	1	1	0
W	0x27	0x27	0x14	0x14	0x14	0x90	0x33

Índex

Conclusions

- Descomposarem els PPE's en dos CLS's:
  - Unitat de Control (UC) i Unitat de Procés (UP)



- Cada cicle:
  - La UC generarà, en funció del seu estat actual, la "paraula de control" que governarà la UP i les sortides de control
  - La UP farà el processament determinat per la "paraula de control" i generarà la "paraula de condició" i les sortides de dades
  - La UC calcularà el seu estat futur en funció del seu estat actual, la "paraula de condició" i les entrades de control
- Heu d'estudiar el document d'autoaprenentatge (slide 37)
- No oblideu realitzar el qüestionari d'Atenea ET7a

Índex

Autoaprenentatge

- Estudieu pel vostre compte el document d'Atenea "8 ejercicios resueltos de E/S síncrona"
  `https://atenea.upc.edu/mod/resource/view.php?id=1673000`
  - Mostra fins a 8 variacions del problema de sumar quatre nombres
  - S'introdueixen modificacions a la senyalització de l'inici d'arribada de dades, de la finalització del càlcul, així com la possibilitat d'avortar un càlcul en curs

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.centives.net/S/2012/what-does-it-take-to-be-an-orchestra-conductor/.

[2] [Online]. Available: http://tandemedicions.com/aplicaciones/siringa1Esp/teoria/vlc/la_orquesta.html.

Introducció als Computadors

Tema 7: Processadors de Propòsit Específic (PPE's)

http://personals.ac.upc.edu/enricm/Docencia/IC/IC7a.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.