

Introducció als Computadors

Tema 3: Circuits Lògics Combinacionals (CLC)

<http://personals.ac.upc.edu/enricm/Docencia/IC/IC3b.pdf>

Enric Morancho
(enricm@ac.upc.edu)

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

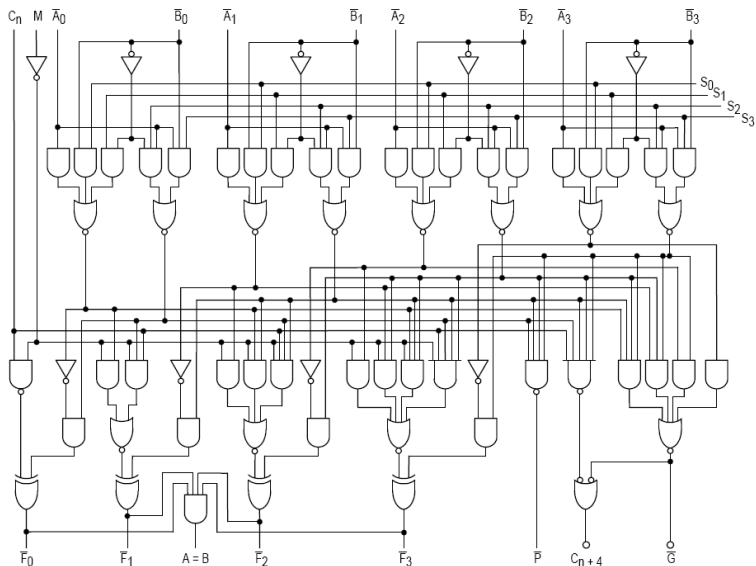


UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Facultat d'Informàtica de Barcelona

2020-21, 1^{er} quad.

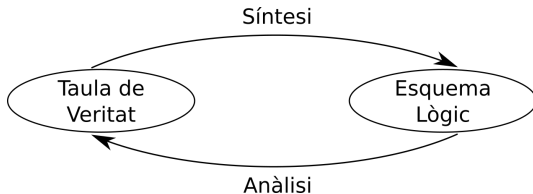
Presentació publicada sota llicència Creative Commons 4.0



[1]

- Síntesi de CLC's
 - Introducció
 - Síntesi mitjançant suma de funcions *minterms*: cas senzill
 - Generalització de les portes AND i OR a n entrades
 - Síntesi mitjançant suma de funcions *minterms*: cas general
 - Síntesi mitjançant descodificador i portes OR
 - Síntesi mitjançant ROM
- Exercicis
- Conclusions
- Miscel·lània

- Síntesi de CLC's
 - Introducció
 - Síntesi mitjançant suma de funcions *minterms*: cas senzill
 - Generalització de les portes AND i OR a n entrades
 - Síntesi mitjançant suma de funcions *minterms*: cas general
 - Síntesi mitjançant descodificador i portes OR
 - Síntesi mitjançant ROM
 - Exercicis
 - Conclusions
 - Miscel·lània



- Donada l'especificació d'un CLC, obtenir un esquema lògic del circuit
 - L'especificació vindrà donada per la Taula de Veritat
 - Poden existir diversos esquemes lògics per a la TV
 - Ja vàrem veure dos CLC's que implementaven la funció XOR
- Veurem tres mètodes de síntesi de CLC's
 - Mitjançant descomposició en suma de *minterms*
 - Cas senzill (portes AND-2, OR-2 i NOT)
 - Cas general (portes AND i OR amb suficients entrades i portes NOT)
 - Mitjançant decodificador i porta OR
 - Mitjançant ROM
- Els mètodes no garanteixen optimalitat sota cap criteri
 - Àrea, cost, energia consumida, retard

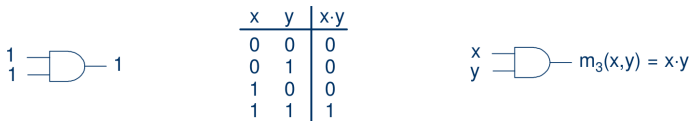
- Síntesi de CLC's
 - Introducció
 - Síntesi mitjançant suma de funcions *minterms*: cas senzill
 - Generalització de les portes AND i OR a n entrades
 - Síntesi mitjançant suma de funcions *minterms*: cas general
 - Síntesi mitjançant descodificador i portes OR
 - Síntesi mitjançant ROM
- Exercicis
- Conclusions
- Miscel·lània

- Funció lògica amb n variables d'entrada i una de sortida que retorna "1" únicament per a una combinació de valors de les entrades
 - Per a totes les altres combinacions retorna "0"
- Amb n variables d'entrada existeixen exactament 2^n funcions *minterms*
 - Per conveni, les anomenarem $m_0, m_1, \dots, m_{2^n-1}$
 - m_i retorna "1" a la combinació de valors d'entrada X tal que $X_u = i$, és a dir, la fila i -èsima de la TV
 - Exemple per a $n = 2$

x	y	$m_3(x, y)$	$m_2(x, y)$	$m_1(x, y)$	$m_0(x, y)$
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

- $m_3(x, y)$ es comporta com la porta lògica AND-2
- La porta AND-2 també serà clau per sintetitzar la resta de *minterms*

- El *minterm* m_3 es sintetitza directament amb una porta AND-2



- Per sintetitzar m_2 cal utilitzar també una porta NOT



- La següent taula mostra la síntesi (algebraica) de tots els *minterms*

x y		$\neg x$	$\neg y$	$m_3(x, y)$ $x \cdot y$	$m_2(x, y)$ $x \cdot \neg y$	$m_1(x, y)$ $\neg x \cdot y$	$m_0(x, y)$ $\neg x \cdot \neg y$
0	0	1	1	0	0	0	1
0	1	1	0	0	0	1	0
1	0	0	1	0	1	0	0
1	1	0	0	1	0	0	0

- Qualsevol funció lògica de n variables d'entrada es pot descompondre com a suma lògica de funcions *minterms* de n variables
 - La "suma lògica" correspon a la funció lògica OR
 - També s'anomena "descomposició en suma de productes"
- Exemple:

x	y	w
0	0	1
0	1	0
1	0	0
1	1	1

x	y	m_3	m_0	w $m_0 + m_3$
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

- $w(x, y) = m_0(x, y) + m_3(x, y) = !x \cdot !y + x \cdot y$

- Aquest mètode té dos passos:
 - 1 Determinar les funcions *minterms* necessàries i sintetitzar-las
 - Utilitzarem portes AND i NOT
 - 2 Fer la suma de les funcions *minterms*
 - Utilitzarem una porta OR (per cada sortida del circuit)

- Taula de veritat, descomposició en suma de *minterms* i esquema lògic.

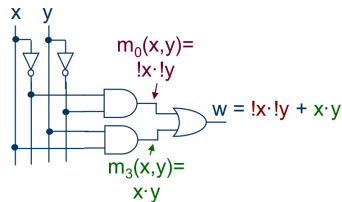
x	y	w
0	0	1
0	1	0
1	0	0
1	1	1

x	y	m_3	m_0	w $m_0 + m_3$
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

$!x \cdot !y$ (red arrow pointing to m_0)

$x \cdot y$ (green arrow pointing to m_3)

+



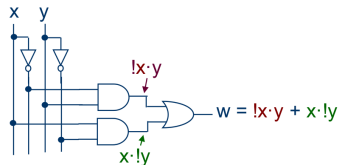
- Aquest circuit implementa la funció lògica NXOR-2
 - Utilitza dues portes NOT, dues AND-2 i una OR-2

- Taula de veritat, descomposició en suma de *minterms* i esquema lògic.

x	y	w
0	0	0
0	1	1
1	0	1
1	1	0

x	y	m_2	m_1	w
0	0	0	0	0
0	1	0	1	1
1	0	1	0	1
1	1	0	0	0

$!x \cdot y$
+
 $x \cdot !y$

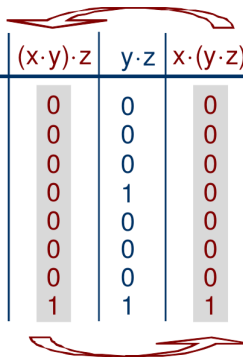


- $w(x, y) = m_1(x, y) + m_2(x, y)$
- Aquest circuit implementa la porta lògica XOR-2
 - Utilitza dues portes NOT, dues AND-2 i una OR-2

- Procediment totalment mecànic
 - Fàcil d'automatitzar
- Limitacions d'aquest mètode
 - Si treballem amb més de dues variables d'entrada, com construïm les funcions *minterms* amb portes AND-2?
 - Si el CLC es descompon en suma de més de dos funcions *minterms*, com fem la suma amb portes OR-2?
- Solució: generalitzar les funcions AND i OR a més de dues entrades

- Síntesi de CLC's
 - Introducció
 - Síntesi mitjançant suma de funcions *minterms*: cas senzill
 - Generalització de les portes AND i OR a n entrades
 - Síntesi mitjançant suma de funcions *minterms*: cas general
 - Síntesi mitjançant descodificador i portes OR
 - Síntesi mitjançant ROM
- Exercicis
- Conclusions
- Miscel·lània

- AND-2 i OR-2 són commutatives i associatives
 - Es pot demostrar, per exemple, amb taules de veritat
 - $(x \cdot y) \cdot z = x \cdot (y \cdot z)$



x	y	z	$x \cdot y$	$(x \cdot y) \cdot z$	$y \cdot z$	$x \cdot (y \cdot z)$
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	1	0
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	1	0	0	0
1	1	1	1	1	1	1

- Tenen element neutre
 - $x \cdot 1 = x$
 - $x + 0 = x$

- Generalització:

- AND- n retorna "1" \iff totes les entrades valen "1"
- OR- n retorna "1" \iff alguna entrada val "1"
- I la XOR- n ?
 - Retorna "1" \iff un nombre senar d'entrades val "1"

- Les representarem amb el mateix símbol que AND-2 i OR-2 afegint-hi les entrades necessàries



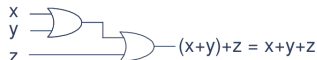
- Utilitzant portes AND-2/OR-2, aprofitant l'associativitat



x	y	z	$x \cdot y \cdot z$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

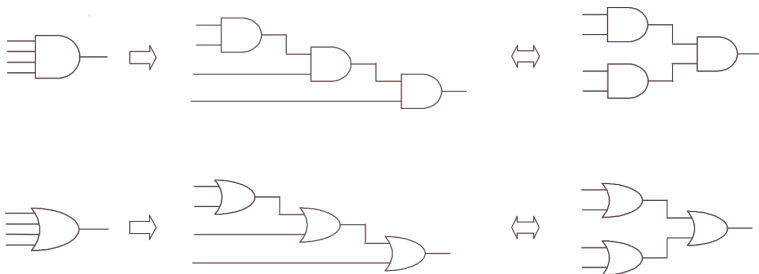


x	y	z	$x + y + z$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1



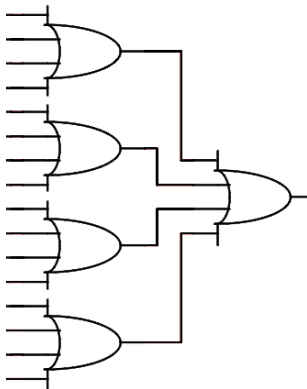
- Mostrem dues opcions utilitzant portes AND-2/OR-2

- Lineal
- En arbre

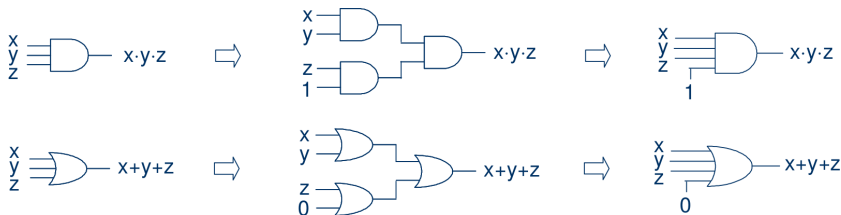


- Quina opció creieu que és millor?

- Arbres amb dos nivells de portes OR-4



- Què fer si necessito portes AND-3/OR-3 però tinc AND-4/OR-4?
 - No podem deixar entrades sense definir
 - Incompliríem regles de connexió de CLC's
 - Aprofitarem l'element neutre de les operacions \cdot i $+$
 - $x \cdot 1 = x$
 - $x + 0 = x$
 - Fem que les entrades innecessàries tinguin un valor constant igual a l'element neutre de l'operació



- Síntesi de CLC's
 - Introducció
 - Síntesi mitjançant suma de funcions *minterms*: cas senzill
 - Generalització de les portes AND i OR a n entrades
 - Síntesi mitjançant suma de funcions *minterms*: cas general
 - Síntesi mitjançant descodificador i portes OR
 - Síntesi mitjançant ROM
- Exercicis
- Conclusions
- Miscel·lània

- *minterm*: funció lògica de n variables que retorna "1" únicament per a una de les combinacions de valors dels senyals d'entrada
 - Per conveni, $m_i(x_{n-1}, \dots, x_1, x_0)$ on $0 \leq i \leq 2^n - 1$ retorna "1" únicament a la combinació de valors X tq $X_u = i$
- Formalització algebraica:

$$m_i(x_{n-1}, \dots, x_1, x_0) = \prod_{k=0}^{n-1} L_k \quad L_k = \begin{cases} \neg x_k & \text{if } i_k = 0 \\ x_k, & \text{if } i_k = 1 \end{cases}$$

$$\text{on } i = (i_{n-1}, i_{n-2}, \dots, i_1, i_0)_2$$

- Exemples ($n = 4$):
 - $m_9(x_3, x_2, x_1, x_0) = x_3 \cdot \neg x_2 \cdot \neg x_1 \cdot x_0$
 - Com $9 = 1001_2 \implies L_3 = x_3, L_2 = \neg x_2, L_1 = \neg x_1, L_0 = x_0$
 - $m_7(x_3, x_2, x_1, x_0) = \neg x_3 \cdot x_2 \cdot x_1 \cdot x_0$
 - Com $7 = 0111_2 \implies L_3 = \neg x_3, L_2 = x_2, L_1 = x_1, L_0 = x_0$
- Qualsevol funció lògica es pot descompondre com a suma de *minterms*

- Taules de veritat de les 8 funcions *minterms* per a $n = 3$

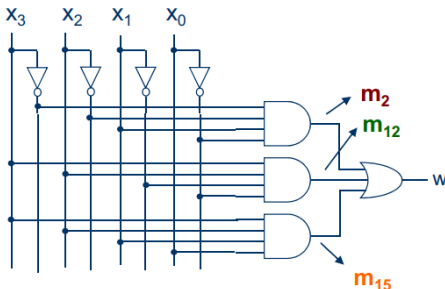
x_2	x_1	x_0	m_7	m_6	m_5	m_4	m_3	m_2	m_1	m_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

- Exemple: $m_3(x_2, x_1, x_0) = \neg x_2 \cdot x_1 \cdot x_0$
- La síntesi de cada funció *minterm* requereix una porta AND-3 i tantes portes NOT com variables calgui negar



Exemple: 4 entrades i 1 sortida

x_3	x_2	x_1	x_0	w
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1



- $w = m_2 + m_{12} + m_{15} = \neg x_3 \cdot \neg x_2 \cdot x_1 \cdot \neg x_0 + x_3 \cdot x_2 \cdot \neg x_1 \cdot \neg x_0 + x_3 \cdot x_2 \cdot x_1 \cdot x_0$
- El CLC utilitza 4 portes NOT, 3 portes AND-4 i 1 porta OR-3
- Què faríeu si a la TV hi hagués alguna sortida a X (*Don't care*)?

- Sintetitzem cada senyal de sortida per separat
 - Si algun *minterm* és comú, no cal sintetitzar-lo dos cops
 - Si una sortida només té un *minterm*, no cal la porta OR per fer la suma

x	y	z	w ₁	w ₀
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	0
1	1	1	0	0

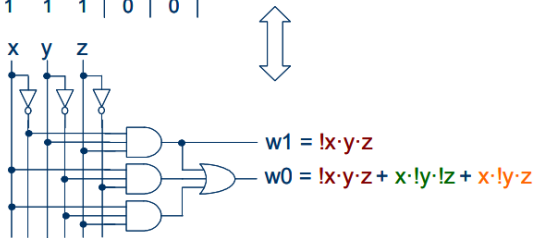
$$m_3(x,y,z) = \bar{x} \cdot y \cdot z$$

$$m_4(x,y,z) = x \cdot \bar{y} \cdot \bar{z}$$

$$m_5(x,y,z) = x \cdot \bar{y} \cdot z$$

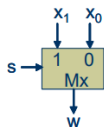
$$w_0 = \bar{x} \cdot y \cdot z + x \cdot \bar{y} \cdot \bar{z} + x \cdot \bar{y} \cdot z$$

$$w_1 = \bar{x} \cdot y \cdot z$$



- El CLC utilitza 3 portes NOT, 3 portes AND-3 i 1 porta OR-3

Exemple: Multiplexor 2-1



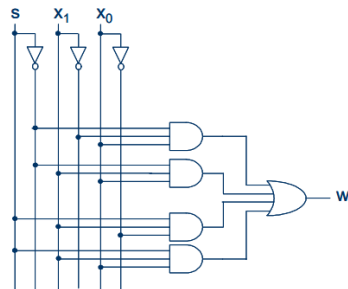
s	x ₁	x ₀	w
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

$\Rightarrow m_1 = !s \cdot !x_1 \cdot x_0 \Rightarrow$

$\Rightarrow m_3 = !s \cdot x_1 \cdot x_0 \Rightarrow$

$\Rightarrow m_6 = s \cdot x_1 \cdot !x_0 \Rightarrow$

$\Rightarrow m_7 = s \cdot x_1 \cdot x_0 \Rightarrow$



- $w = m_1 + m_3 + m_6 + m_7 = !x_2 \cdot !x_1 \cdot x_0 + !x_2 \cdot x_1 \cdot x_0 + x_2 \cdot x_1 \cdot !x_0 + x_2 \cdot x_1 \cdot x_0$
- El CLC utilitza 3 portes NOT, 4 portes AND-3 i 1 porta OR-4

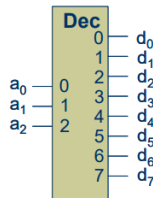
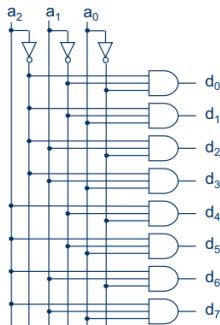
- Síntesi de CLC's
 - Introducció
 - Síntesi mitjançant suma de funcions *minterms*: cas senzill
 - Generalització de les portes AND i OR a n entrades
 - Síntesi mitjançant suma de funcions *minterms*: cas general
 - Síntesi mitjançant descodificador i portes OR
 - Síntesi mitjançant ROM
- Exercicis
- Conclusions
- Miscel·lània

- També es basa en la descomposició en suma de *minterms*
 - Però ens estalvia la síntesi dels *minterms* perquè ja ens la proporciona un bloc combinacional anomenat descodificador (*decoder*)
 - Dec- $n-2^n$: CLC amb n entrades i 2^n sortides
- Com al cas previ, amb la porta OR sumem les funcions *minterms*

- CLC que amb n senyals d'entrada i 2^n de sortida
 - n entrades: $A = a_{n-1}a_{n-2} \dots a_1a_0$
 - A rep el nom d'adreça (address) o direcció
 - 2^n sortides: $D = d_{2^n-1}d_{2^n-2} \dots d_1d_0$
- Què calcula?
 - $d_j = 1 \iff A_u = j$
 - Únicament una sortida tindrà el valor "1"
 - És la sortida corresponent a A_u
 - També podem visualitzar-lo com que retorna la TV del *minterm* m_{A_u}
- Implementació:
 - n portes NOT
 - 2^n portes AND de n entrades

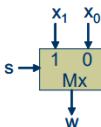
- Mostrem TV, implementació i encapsulat del Dec-3-8
 - Respecteu l'etiquetat de les entrades i sortides del Descodificador

a_2	a_1	a_0	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

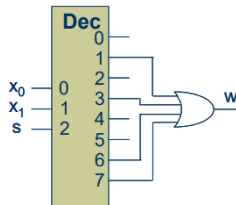


- Exemple: donat un nombre binari $A = a_2a_1a_0$, $d_j = "1"$ $\iff A_u = j$
 - $A = 110 \implies A_u = 6 \implies d_6 = 1$ i la resta de sortides a "0"

- L'ordre de les variables d'entrada a la TV ($s x_1 x_0$) ha de correspondre's amb el pes de les entrades al bloc Dec-3-8
 - s amb pes 2, x_1 amb pes 1, x_0 amb pes 0
- Per sintetitzar la sortida w cal una porta OR de quatre entrades
 - w es descompon com a suma de quatre *minterms*
 - Connectem les sortides del Dec-3-8 corresponents als *minterms* de la funció w amb la porta OR-4

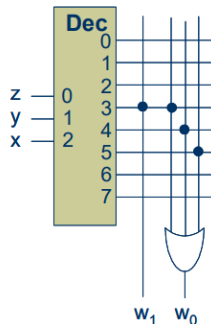


s	x_1	x_0	w
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1



- L'ordre de les variables d'entrada a la TV (xyz) ha de correspondre's amb el pes de les entrades al bloc Dec-3-8
 - x amb pes 2, y amb pes 1, z amb pes 0
- Sortides:
 - Com la sortida w_1 només utilitza un *minterm*, no cal porta OR per sintetitzar-lo
 - Per sintetitzar w_0 cal una porta OR-3

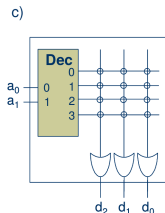
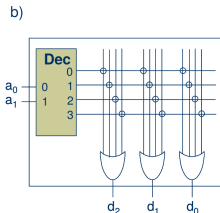
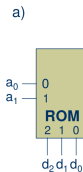
x	y	z	w_1	w_0
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	0
1	1	1	0	0



- Síntesi de CLC's
 - Introducció
 - Síntesi mitjançant suma de funcions *minterms*: cas senzill
 - Generalització de les portes AND i OR a n entrades
 - Síntesi mitjançant suma de funcions *minterms*: cas general
 - Síntesi mitjançant descodificador i portes OR
 - Síntesi mitjançant ROM
- Exercicis
- Conclusions
- Miscel·lània

- També es basa en la descomposició en suma de *minterms*
- Compacta encara més la síntesi del circuit
- Utilitzarem un CLC que anomenarem ROM
 - *Read Only Memory*
 - Integra un Descodificador, matriu de connexions i portes OR

- Dispositiu que consta de:
 - Un Dec- $n-2^n$ per generar tots els *minterms* de n entrades
 - Una matriu de de $2^n \times m$ punts de connexió
 - m portes OR amb 2^n entrades
 - Per valors grans de n , es fa servir una tecnologia que permet crear portes OR amb una única entrada on s'hi connecten els 2^n senyals
- Exemple: ROM amb 2 entrades i 3 sortides
 - a) Encapsulat de la ROM
 - $A = (a_1, a_0)$ és l'adreça (*address*)
 - $D = (d_2, d_1, d_0)$ és la dada (*data*)
 - b) Esquema lògic
 - c) Esquema tecnològic



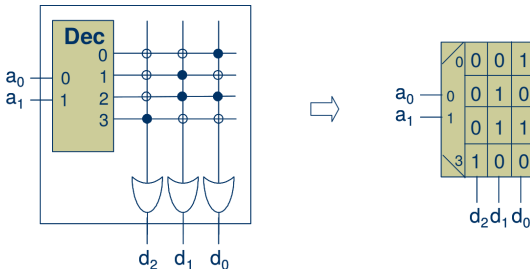
- Per implementar un CLC amb n entrades i m sortides cal una ROM amb n entrades i m sortides
 - Les entrades del CLC són directament les entrades de la ROM
 - Les sortides del CLC són directament les sortides de la ROM
- Cal determinar quines connexions de la matriu cal activar
 - Les dels *minterms* que contribueixen a cada sortida
 - Un cop fabricada la ROM, no es podran modificar
 - Alternatives:
 - PROM (Programmable ROM): el connexionat el fa el comprador però no es podrà modificar
 - EPROM (Erasable PROM): el connexionat es pot sobreescriure
- Cal respectar el pes de les entrades i de les sortides de la ROM

Exemple 1 ($n=2$, $m=3$)

- Sintetitzar un CLC amb 2 entrades i 3 sortides amb la següent TV

a_1	a_0	d_2	d_1	d_0
0	0	0	0	1
0	1	0	1	0
1	0	0	1	1
1	1	1	0	0

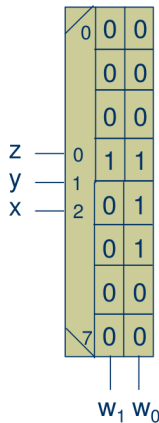
- Mostrem el connexionat així com la representació compacta
 - Vigileu amb els pesos dels senyals i l'ordre de les files!
 - A la representació compacta incloem la taula de veritat del CLC



Exemple 2 ($n=3$, $m=2$)

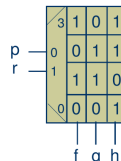
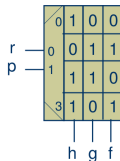
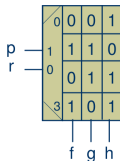
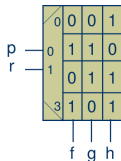
- Sintetitzar un CLC amb 3 entrades i 2 sortides amb la següent TV

x	y	z	w_1	w_0
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	0
1	1	1	0	0



- Què fem si hi ha alguna sortida a X (*Don't care*)?

- El CLC tindrà una funcionalitat diferent depenent de l'ordenació



r	p	f	g	h
0	0	0	0	1
0	1	1	1	0
1	0	0	1	1
1	1	1	0	1

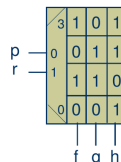
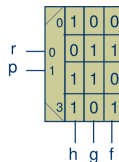
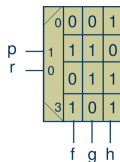
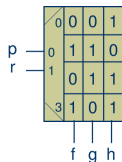
p	r	f	g	h
0	0	0	0	1
0	1	1	1	0
1	0	0	1	1
1	1	1	0	1

p	r	h	g	f
0	0	1	0	0
0	1	0	1	1
1	0	1	1	0
1	1	1	0	1

r	p	f	g	h
0	0	0	0	1
0	1	1	1	0
1	0	0	1	1
1	1	1	0	1

- Quins tenen la mateixa funcionalitat?

- El CLC tindrà una funcionalitat diferent depenent de l'ordenació



r	p	f	g	h
0	0	0	0	1
0	1	1	1	0
1	0	0	1	1
1	1	1	0	1

p	r	f	g	h
0	0	0	0	1
0	1	1	1	0
1	0	0	1	1
1	1	1	0	1

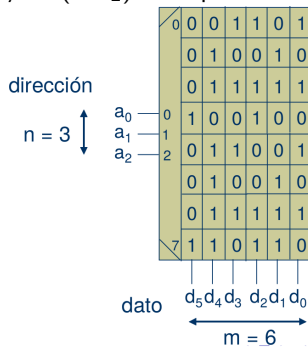
p	r	h	g	f
0	0	1	0	0
0	1	0	1	1
1	0	1	1	0
1	1	1	0	1

r	p	f	g	h
0	0	0	0	1
0	1	1	1	0
1	0	0	1	1
1	1	1	0	1

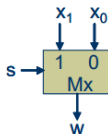
- Quins tenen la mateixa funcionalitat?

Primera i quarta, segona i tercera

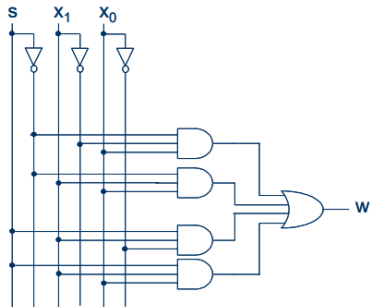
- Una ROM emmagatzema informació *read only* de forma no-volàtil
 - Una ROM de n entrades i m sortides emmagatzema $2^n \cdot m$ bits
- Terminologia:
 - Adreça/Direcció/@ (*address*): $0 \leq A_u \leq 2^n - 1$
 - Paraula (*word*): vector de m bits
- Exemple: ROM de 8 paraules de 6 bits cadascuna ($n = 3, m = 6$)
 - Mida: $2^3 \cdot 6 = 48$ bits
 - El contingut de l'adreça 3 (011_2) és la paraula 100100_2



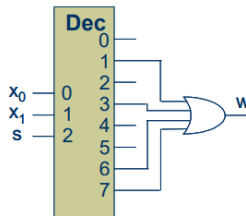
Tres síntesis del CLC Multiplexor 2-1



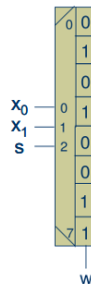
s	x ₁	x ₀	w
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1



Síntesi amb suma de minterms



Síntesi amb decodificador



Síntesi amb ROM

- Síntesi de CLC's
- Exercicis
- Conclusions
- Miscel·lània

a) ¿Cuántas puertas And y Or y de cuántas entradas cada una hacen falta para implementar directamente la expresión en suma de minterms de la función w de la siguiente tabla de verdad

a	b	c	w
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Número puertas AND = de entradas.

Número puertas OR = de entradas.

b) Especificar el tamaño mínimo de la ROM para sintetizar un circuito de 4 entradas y 3 salidas.

Número de palabras = Bits por palabra =

a) ¿Cuántas puertas And y Or y de cuántas entradas cada una hacen falta para implementar directamente la expresión en suma de minterms de la función w de la siguiente tabla de verdad

a	b	c	w
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Número puertas AND = de entradas.

Número puertas OR = de entradas.

b) Especificar el tamaño mínimo de la ROM para sintetizar un circuito de 4 entradas y 3 salidas.

Número de palabras =

Bits por palabra =

Dado el esquema del siguiente circuito (incluida la tabla de verdad del bloque C1) completad la tabla de verdad de la salida W y escribid la expresión lógica en suma de minterms.

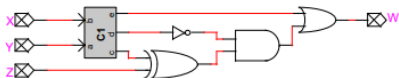


Tabla de verdad de C1

a	b	c	d	e
0	0	1	1	0
0	1	0	0	0
1	0	1	0	1
1	1	1	1	0

Tabla de verdad de W:

X	Y	Z	W
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Expresión en suma de minterms de W:

Dado el esquema del siguiente circuito (incluida la tabla de verdad del bloque C1) completad la tabla de verdad de la salida W y escribid la expresión lógica en suma de minterms.

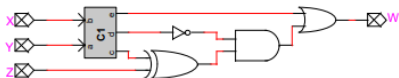


Tabla de verdad de C1

a	b	c	d	e
0	0	1	1	0
0	1	0	0	0
1	0	1	0	1
1	1	1	1	0

Tabla de verdad de W:

X	Y	Z	W
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

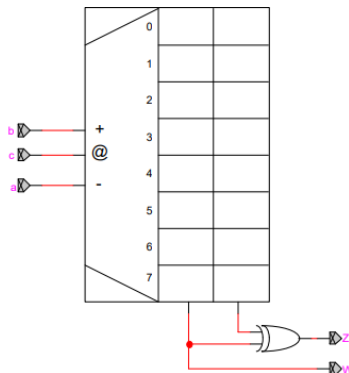
Expresión en suma de minterms de W:

$\neg X \cdot Y \cdot \neg Z + \neg X \cdot Y \cdot Z + X \cdot \neg Y \cdot Z$

- Queremos implementar un circuito combinacional cuya funció/comportament està descrit en la següent taula de veritat. Per implementar-lo ja tenim el circuit dissenyat. Rellena el contingut de la ROM del circuit per que faci la funció descrita en la taula de veritat

Tabla de verdad

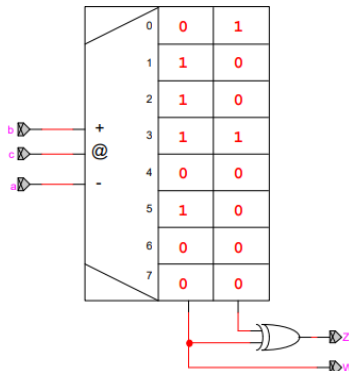
a	b	c	Z	W
0	0	0	1	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	1	1
1	1	1	0	0



- Queremos implementar un circuito combinacional cuya funció/comportament està descrit en la següent taula de veritat. Per implementar-lo ja tenim el circuit dissenyat. Rellena el contingut de la ROM del circuit per que faci la funció descrita en la taula de veritat

Tabla de verdad

a	b	c	Z	W
0	0	0	1	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	1	1
1	1	1	0	0



- Síntesi de CLC's
- Exercicis
- **Conclusions**
- Miscel·lània

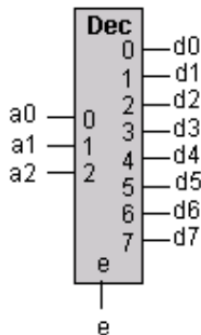
- Qualsevol CLC és sintetitzable amb portes AND-2, OR-2 i NOT
- Hem introduït les funcions lògiques *minterms*
 - Retornen "1" únicament per a una combinació de les entrades
 - Tota funció lògica es pot descompondre com a suma de *minterms*
 - També s'anomena descomposició en suma de productes
- Mètodes de síntesi vistos:
 - Mitjançant descomposició en suma de funcions *minterms*
 - Mitjançant descodificador i portes OR
 - Mitjançant ROM
- Els esquemes obtinguts no tenen perquè ser òptims
 - Es més, les síntesis amb descodificador i amb ROM utilitzen més hardware del que seria estrictament necessari
 - Veurem com simplificar circuits (mapes de Karnaugh)
- Terminologia ROM's: *address* (adreça) i *word* (paraula)
- No oblideu respondre l'ET3b a Atenea!

- Síntesi de CLC's
- Exercicis
- Conclusions
- **Miscel·lània**

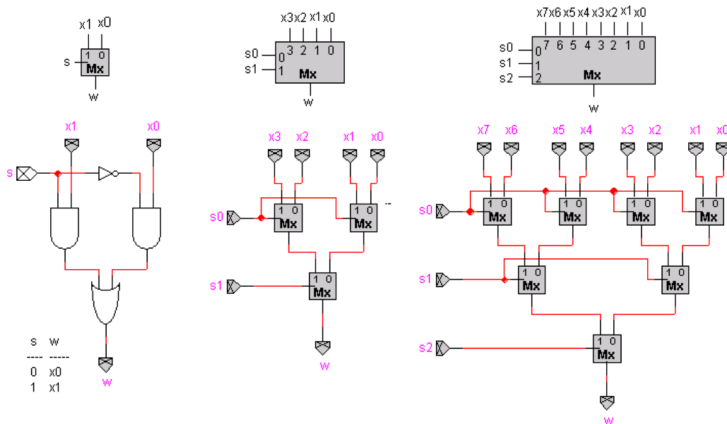
- Afegeix una entrada extra al Dec- $n-2^n$, la senyal *enable*
 - Si val 0, totes les sortides del Decoder valdran 0
 - Si val 1, el Decoder operarà amb normalitat
- Exemple: Dec-3-8 amb senyal *enable*

Truth Table (compressed):

e	a2	a1	a0	d7	d6	d5	d4	d3	d2	d1	d0
0	X	X	X	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	1	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0



- El multiplexor 2-1 es pot generalitzar a multiplexor 2^n-1
 - Senyals d'entrada: 2^n senyals de dades (x) i n senyals de selecció (s)
 - Senyal de sortida: 1 senyal de dades
 - $Mx-2^n-1$ posa a la sortida la dada corresponent a la selecció



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://upload.wikimedia.org/wikipedia/commons/c/c0/74181aluschematic.png>.

Introducció als Computadors

Tema 3: Circuits Lògics Combinacionals (CLC)

<http://personals.ac.upc.edu/enricm/Docencia/IC/IC3b.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.

Presentació publicada sota llicència Creative Commons 4.0