UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Generació de funcions lògiques mitjançant descodificadors binaris amb eixides actives a nivell baix

Cognoms, nom	Martí Campoy, Antonio (amarti@disca.upv.es)
Departament	Informàtica de Sistemes i Computadors
Centre	Universitat Politècnica de València

UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

1 Resum de les idees clau

En aquest article es presentarà la utilització de descodificadors binaris amb eixides actives a nivell baix per a la generació de funcions lògiques. Són moltes les formes de dissenyar una funció lògica, i una de les més senzilles és la utilització del bloc combinacional conegut com a descodificador binari. Per a poder adquirir els coneixements i habilitats presentades en aquest article, has de comptar amb uns coneixements previs, llistats en la taula 1. Però tranquil, durant el text s'inclouran breus descripcions d'aquestes idees prèvies.

Taula 1 Coneixements previs

Coneixements previs
1. Què és una funció lògica i la seua aritat
2. Tipus i taules de veritat de portes lògiques
3. Formes de representar una funció lògica: taula de veritat, formes canòniques i expressions algebraiques
4. Funcionament d'un descodificador binari
5. Circuit intern d'un descodificador binari

2 Objectius

Una vegada llegit aquest article docent i reproduïts els exemples presentats, hauràs de ser capaç **d'implementar** una funció lògica mitjançant l'ús de descodificadors binaris amb eixides actives a nivell **baix**.

A més a més, la implementació de la funció lògica podrà prendre com a punt de partida diferents representacions de la mateixa, com la taula de veritat o una forma canònica, per la qual cosa seràs capaç de **traduir** des d'una representació a una altra.

Finalment, i atenent criteris de simplificació de circuits, seràs capaç d'**escollir** el tipus de porta lògica més adequada.

3 Introducció

En primer lloc, una breu descripció dels conceptes previs més importants per a poder assolir els objectius proposats en aquest article. Aquestes descripcions poden ampliar-se consultant la bibliografia proposada al final del document.

- Funció lògica: expressió formal del comportament d'un circuit digital. L' aritat d'una funció lògica és el nombre de variables d'entrada.

- Taula de veritat: representació única en forma de taula d'una funció lògica.
- Formes canòniques: representació única com a suma de productes o com a producte de sumes d'una funció lògica.
- Expressió algebraica: combinació de variables i operadors lògics par expressar una funció lògica.
- Porta lògica: circuit digital que implementa una funció lògica bàsica.
- Circuit o funció combinacional: circuit en el que les eixides en un instant de temps depenen exclusivament de les entrades en eixe mateix instant de temps.
- Descodificador binari: circuit combinacional, amb m entrades binàries i $n=2^m$ eixides binàries. Les eixides s'activen de forma exclusiva, és a dir, tan sols s'activa una d'elles en un instant concret.
- La funció realitzada per un descodificador binari consisteix en activar l'eixida d'ordre i que correspon amb la codificació binària de les seues entrades. La figura 1 presenta el símbol lògic d'un descodificador binari d'm m a $n=2^m$ amb eixides actives a nivell **baix**.
- Una eixida activa a nivell baix vol dir que prendrà valor zero quan estiga activada, i valor un quan no estiga activada.

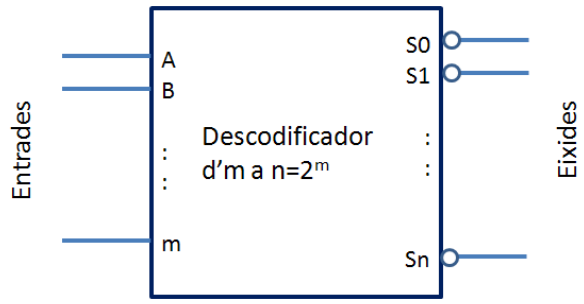

Figura 1 Símbol lògic d'un descodificador d'm a n amb eixides actives a nivell baix.

Per a veure si el funcionament del descodificador és clar, ens farem un parell de preguntes. Suposem un descodificador binari de 2 a 4 amb les eixides actives a nivell baix. Les entrades s'anomenen B i A, sent A la de menor pes, i les eixides reben el nom d'S0, S1, S2 i S3. Si els valors de les entrades són B=1 i A=0, el valor de les eixides S0, S1, S2 i S3 és:

Per favor, pensa quina és la resposta abans de mirar la solució[1]

Provem-ho una altra vegada. Si els valors de les entrades són B=0 i A=1, el valor de les eixides S0, S1, S2 i S3 és:

Per favor, pensa quina és la resposta abans de mirar la solució[2]

[1] El valor de les eixides es S0=1, S1=1, S2=0, S3=1, perquè BA=10 es correspon amb el valor binari 2, per la qual cossa s'activa l'eixida S2 amb valor 0 (baix) i la resta de les eixides amb valor 1.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

4 Generació de funcions

En aquest apartat veurem, en primer lloc, el significat algebraic de les eixides del descodificador binari amb eixides actives a nivell baix. En segon lloc, recordarem breument que una funció lògica pot crear-se a partir de la forma canònica conjuntiva coneguda com el producte dels seus maxitermes. Finalment, unint les dues idees prèvies, utilitzarem portes lògiques per a generar una funció emprant descodificadors binaris amb eixides actives a nivell baix.

4.1 Significat de les eixides del descodificador binari

La implementació interna d'un descodificador binari amb eixides actives a nivell baix és molt senzilla. Per a cadascuna de les eixides es realitza un circuit que activa l'eixida (posant un zero) si les entrades prenen el valor corresponent. Per exemple, per a un descodificador de 2 a 4, les eixides es corresponen amb les expressions algebraiques mostrades en l'equació 1, que coincideixen, a més a més, amb les expressions dels maxitermes[3].

Ara que coneixem que les eixides d'un descodificador binari corresponen amb la implementació de cadascun dels maxitermes, podem incloure aquesta informació en el símbol lògic, que es mostra en la figura 2. Açò és important per a, posteriorment, comprendre com generar una funció emprant descodificadors binaris amb eixides actives a nivell baix.

Equació 1 Expressions algebraiques per a les eixides d'un descodificador binari de 2 a 4 amb eixides actives a nivell baix

$$S0 = B + A = \prod_{B,A}(0)$$

S0 prendrà valor 0 si B = A = 0

$$S1 = B + \bar{A} = \prod_{B,A}(1)$$

S1 prendrà valor 0 si B = 0 i A = 1

$$S2 = \bar{B} + A = \prod_{B,A}(2)$$

S2 prendrà valor 0 si B = 1 i A = 0

$$S3 = \bar{B} + \bar{A} = \prod_{B,A}(3)$$

S3 prendrà valor 0 si B = A = 1

4.2 Forma Canònica Conjuntiva

Una forma senzilla de construir una funció lògica és a partir de la forma canònica conjuntiva, coneguda també com a producte de sumes o producte dels maxitermes de la funció.

---

[2] El valor de les eixides és S0=1, S1=0, S2=1, S3=1, perquè que BA=01es correspon amb el valor binari 1, per la qual cossa s'activa l'eixida S1 amb valor 0 (baix)

[3] Un maxiterme és la suma de totes les variables d'entrada, que apareixen en forma directa si el seu valor es 0 i en forma negada si el seu valor es 1

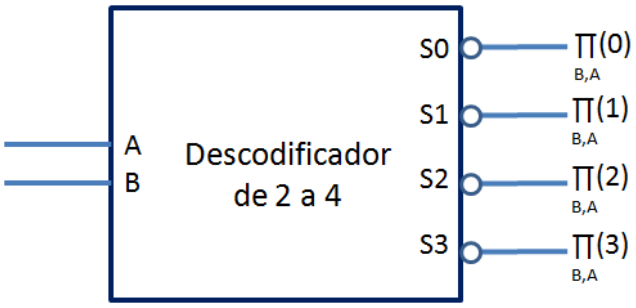

Figura 2 Descodificador 2 a 4 amb identificació dels maxitermes

Quins són els maxitermes d'una funció? Són aquells per als que la funció pren valor 0. I la forma canònica conjuntiva diu que una funció és el producte dels seus maxitermes. Però millor vegem un exemple. La taula 2 mostra una funció de nom H i aritat 3, i l'equació 2 mostra la forma canònica conjuntiva d'aquesta funció.

Taula 2 Taula de veritat de la funció H

Maxiterme	C B A	H
0	0 0 0	0
1	0 0 1	1
2	0 1 0	0
3	0 1 1	0
4	1 0 0	1
5	1 0 1	0
6	1 1 0	1
7	1 1 1	0

Equació 2 Forma Canònica Conjuntiva per a la funció H

$$H = \prod_{C,B,A}(0,2,3,5,7)$$

Per construir el circuit que implementa la funció H es pot utilitzar portes lògiques, implementat els maxitermes de la funció amb portes OR, i utilitzant una porta AND per a realitzar el producte lògic dels maxitermes. En total, comptant les portes NOT necessàries per a construir els maxitermes, necessitem_____[4] portes.

4.3 Generació de funcions amb descodificadors

Tal com hem vist abans, podem crear una funció seguint els passos mostrats en la figura 3.

L'últim pas correspon a la implementació dels maxitermes de la funció mitjançant portes NOT i portes OR, i al producte utilitzant una porta AND. Però, com hem vist anteriorment, un descodificador binari amb eixides actives a nivell baix implementa, en cadascuna de les seues eixides, un maxiterme, per la qual cosa la primera part de la construcció del circuit pot ser substituïda per un descodificador binari, amb

[4] Se necesitan una puerta AND de 5 entradas, 5 puertas OR de 3 entradas, y 3 puertas NOT. En total, 9 puertas.

tantes entrades com a variables d'entrada tinga la funció lògica que es vol implementar.

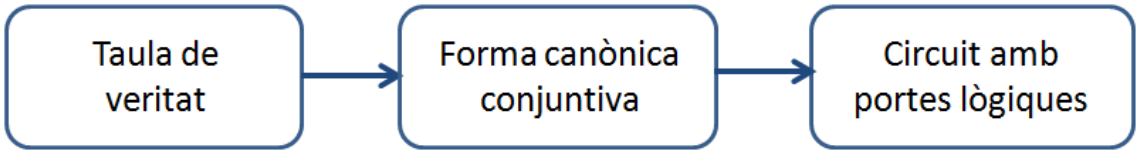

Figura 3 Passos per implementar una funció lògica

4.3.1 Implementació amb una porta AND

Podem aplicar la propietat associativa per al producte lògic, mostrada en l'equació 3, al producte de maxitermes de la funció d'exemple H, mostrada en l'equació 4.

Equació 3 Propietat associativa per al producte

$$(a \cdot b \cdot \cdots \cdot n) = a \cdot b \cdot \cdots \cdot n$$

Equació 4 Aplicació de la propietat associativa a la forma canònica conjuntiva de la funció G de l'exemple de la taula 1

$$H = \prod_{C,B,A}(0,2,3,5,7) = \prod_{C,B,A}(0) \cdot \prod_{C,B,A}(2) \cdot \prod_{C,B,A}(3) \cdot \prod_{C,B,A}(5) \cdot \prod_{C,B,A}(7)$$

Bé, el que ens diu la propietat associativa en aquest cas és que podem agafar els maxitermes d'una funció, és a dir, les eixides corresponents del descodificador, i realitzar el producte lògic per mitjà d'una porta AND. I l'eixida d'aquesta porta AND es correspon amb la implementació de la funció. La figura 4 mostra la implementació de la funció H utilitzant un descodificador de 3 a 8 amb eixides actives a nivell baix i una porta AND.

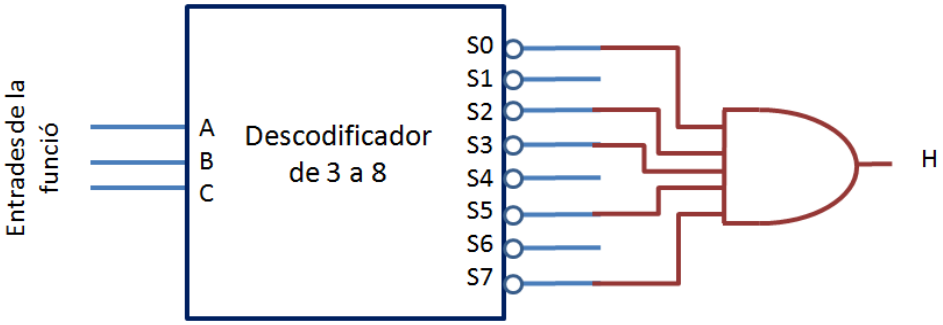

Figura 4 Implementació de la funció H mitjançant un descodificador binari i una porta AND

D'aquesta manera, per implementar una funció lògica tan sols necessite un descodificador binari amb tantes entrades com tinga la funció i una porta AND.

4.3.2 Implementació amb una porta NAND

Però, què passa si no puc aconseguir la porta AND que necessite?

Fem una altra pregunta. Què succeeix si en compte d'agafar els maxitermes que **SÍ** que són de la funció, agafem els que **NO** són de la funció?

Açò seria el mateix que canviar els zeros per uns i els uns per zeros en l'eixida de la funció. Per exemple, amb la funció H, tindríem la taula de veritat mostrada en la taula 3, amb la funció $\overline{H}$, la negació de la funció H. L'equació 5 mostra la forma canònica disjuntiva per a $\overline{H}$.

Taula 3 Taula de veritat de la funció H i $\overline{H}$

Minitérme	C B A	H	$\overline{H}$
0	0 0 0	0	1
1	0 0 1	1	0
2	0 1 0	0	1
3	0 1 1	0	1
4	1 0 0	1	0
5	1 0 1	0	1
6	1 1 0	1	0
7	1 1 1	0	1

Equació 2 Forma Canònica Conjuntiva per a la funció H

$$H = \prod_{C,B,A}(0,2,3,5,7)$$

Equació 5 Forma Canònica Disjuntiva per a la funció $\overline{H}$

$$\overline{H} = \prod_{C,B,A}(1,4,6)$$

Però nosaltres no volem implementat la funció $\overline{H}$, sinó que volem implementar la funció H. I açò ho podem aconseguir agafant els maxitermes que **NO** són de la funció i utilitzar, en compte d'una porta **AND**, una porta **NAND**. D'aquesta manera aprofitem la propietat anomenada involució[5] i al negar $\overline{H}$, obtenim H, que és el que buscàvem.

La figura 5 mostra la implementació de la funció H mitjançant l'ús d'un descodificador de 3 a 8 amb eixides actives a nivell baix i una porta NAND.

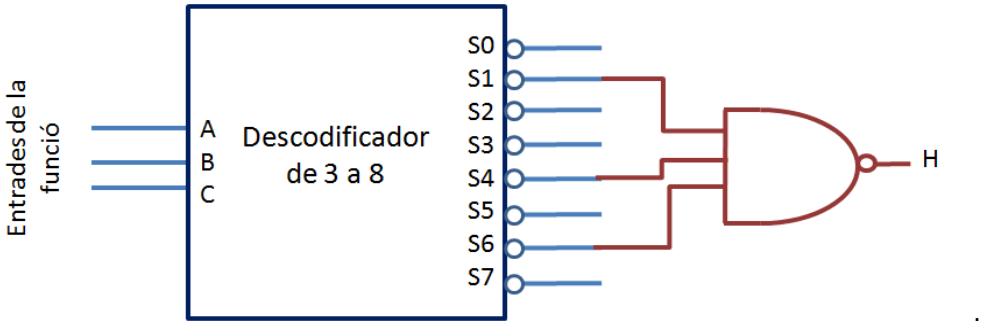

Figura 5 Implementació de la funció H mitjançant un descodificador binari i una porta NOR

[5] La propietat anomenada involució diu que $\overline{\overline{a}} = a$, és a dir, que si neguem una funció dues vegades, obtenim la funció original.

4.3.3 Resum

El resultat d'utilitzar una porta AND triant els maxitermes que SÍ que són de la funció és el mateix que el d'utilitzar una porta NAND triant els maxitermes que NO són de la funció. Així, doncs, a nivell funcional les dues opcions són equivalents. Tanmateix això, algunes funcions tenen un nombre més gran de zeros que d'uns en les seues eixides, o viceversa. En aquests casos, la utilització d'una porta AND o d'una porta NAND pot tenir importància en la complexitat i cost del circuit final, ja que és possible triar la porta amb menor nombre d'entrades.

4.3.4 Exercicis

Seguidament uns exercicis senzills. Per a la funció lògica H expressada per la seua forma canònica conjuntiva, respon a les preguntes següents:

$$H = \prod_{D,C,B,A} (0,2,3,6,8,12,13,15)$$

a. Quina grandària de descodificador binari necessitem per implementar-la?
b. Si utilitzem una porta AND, quantes entrades cal que tinga la porta?
c. Si utilitzem una porta AND, quines seran les eixides del descodificador que connectarem a la porta AND?
d. Si utilitzem una porta NAND, quantes entrades cal que tinga la porta?
e. Si utilitzem una porta NAND, quines seran les eixides del descodificador que connectarem a la porta NAND?
f. És millor utilitzar una porta AND o una porta NAND?
g. Prova de respondre les preguntes abans de veure les solucions, per favor [6].

5 Conclusions

En aquest article has pogut conèixer una forma ràpida i senzilla d'implementar una funció lògica. La figura 6 mostra els passos a seguir per a, partint de la taula de veritat d'una funció lògica, arribar a la implementació de la funció utilitzant un descodificador amb eixides actives a nivell baix i una porta AND o una porta NAND.

Algunes idees importants que cal recordar:

- El descodificador ha de tenir tantes entrades com a entrades tinga la funció. És a dir, ha de tenir la mateixa aritat que la funció.
- L'entrada de menor pes de la funció ha de connectar-se a l'entrada de menor pes del descodificador. I així successivament amb les entrades següents fins a la de major pes.
- Les eixides del descodificador que no s'utilitzen es deixen a l'aire.
- No és millor utilitzar una porta AND o una porta NAND, però depenent de la funció, és possible que resulte més senzill una opció enfront de l'altra. Però funcionalment les dues opcions són idèntiques.

[6] a: De 4 entrades a 16 eixides; b: De 8 entrades; c: S0, S2, S3, S6, S8, S12, S13, S15; d: De 16-8=8 entrades; e: S1, S4, S5, S7, S9, S10, S11, S14 f: La porta NAND necessita el mateix nombre d'entrades, per la qual cossa és indiferent gastar la AND o la NAND.

- Si el descodificador té entrada d'habilitació, aquesta ha d'estar sempre activada, ja que en cas contrari no es genera cap funció.

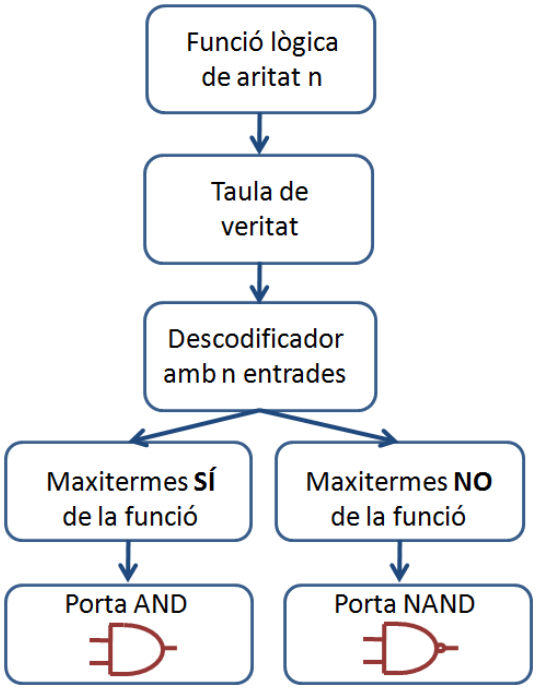

Figura 6 Passos per a la implementació d'una funció lògica utilitzant descodificadors binaris amb eixides actives a nivell baix

6 Bibliografia

6.1 Llibres:

[1] John F. Wakerly *Digital design : principles and practices*, Upper Saddle River : Pearson Prentice Hall. 2006

[2] Antonio Lloris Ruiz; Alberto Prieto Espinosa; Luis Parrilla Roure *Sistemas digitales*, Aravaca, Madrid : McGraw-Hill/Interamericana de España. 2003

6.2 Referències de fons electròniques:

[3] Martí Campoy, Antonio *"Circuitos combinacionales: decodificadores"* Universitat Politècnica de València. Escola Tècnica Superior d'Enginyeria Informàtica. 2011. http://politube.upv.es/play.php?vid=46040