

Previ DGD practica 2

Daniel Vilardell

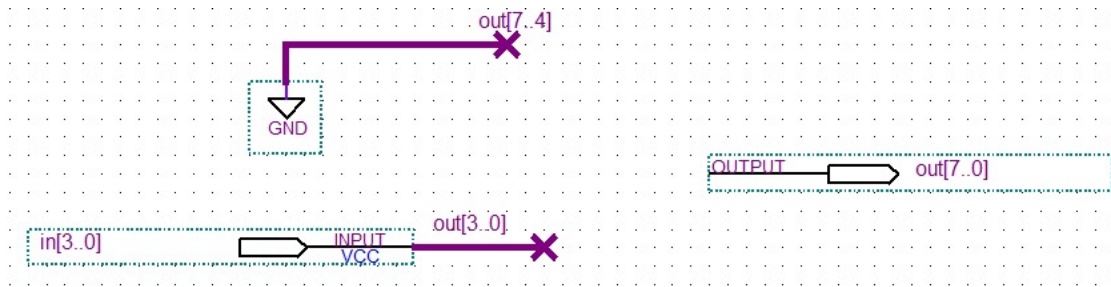
Igor Yuziv

Pregunta 1: Per dissenyar el bloc combinacional AxB aprofitau part del disseny fet en la pràctica anterior, on justament es feia un multiplicador. Expliqueu com era aquest disseny i quines modificacions cal fer-li per poder utilitzar-lo a ppal.

Per tal de fer el bloc AxB , usarem el bloc multiplicador realitzat a la practica anterior, el que com que l'entrada es de dos nombres de 4 bits mentres que el component multiplicador se li entren dos entrades de 8 bits, haurem de fer un conversor de nombres en binari de 4 a 8 bits. La sortida ens la donara en binari, per a passarla a BCD usarem un component programat amb vhdl que considerarà tots els possibles nombres solucio i assignarà al resultat el que li toqui.

Primer de tot farem un breu resum de com funcionava el multiplicador. Aquest feia la multiplicació escolar, es a dir, colocava un nombre a dalt i un a baix i multiplicava xifra per xifra. A mesura que anava multiplicant extreia el nombre de menys pes de cada producte individual i el situava a la sortida. Els altres digits els sumava amb el següent. Això fins als ultims digits que tots eren destinats al output.

El component conversor de 4 a 8 bits serà el següent



El component que ens convertira la sortida del multiplicador de binari a BCD es el següent.

```

LIBRARY ieee; USE ieee.std_logic_1164.ALL;

ENTITY BIN_BCD_8B IS PORT (
    BIN : IN STD_LOGIC_VECTOR(7 downto 0);
    BCD : OUT STD_LOGIC_VECTOR(7 downto 0));
END BIN_BCD_8B;

ARCHITECTURE taula_veritat OF BIN_BCD_8B IS
    BEGIN
        with BIN SELECT BCD <=
            "10011001" WHEN "00111000",  — 81
            "01110010" WHEN "01001000",  — 72
            "01100100" WHEN "01000000",  — 64
            "01010110" WHEN "00111000",  — 56
            "01010100" WHEN "00111000",  — 54
            "00101000" WHEN "00011100",  — 28
            "01001001" WHEN "00110001",  — 49
            "01001000" WHEN "00110000",  — 48
            "01000101" WHEN "00101101",  — 45
            "01000010" WHEN "00101010",  — 42
            "01000000" WHEN "00101000",  — 40
            "00110110" WHEN "00100100",  — 36
            "00110101" WHEN "00100011",  — 35
            "00110010" WHEN "00100000",  — 32
            "00110000" WHEN "00011110",  — 30
            "00101000" WHEN "00011100",  — 28
            "00100110" WHEN "00011011",  — 27
            "00100101" WHEN "00011001",  — 25

```

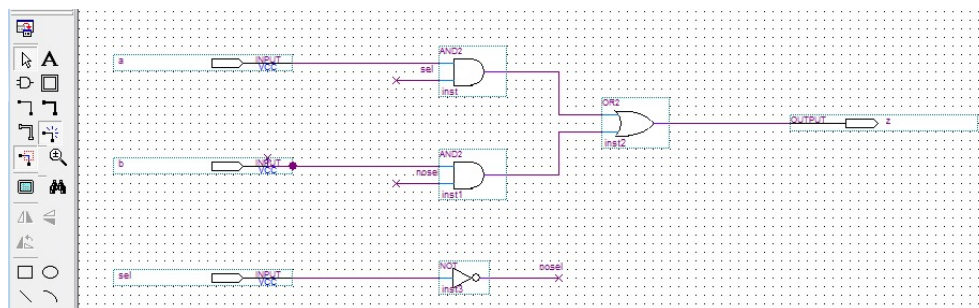
```

"00100100" WHEN "00011000", — 24
"00100001" WHEN "00010101", — 21
"00100000" WHEN "00010100", — 20
"00011000" WHEN "00010010", — 18
"00010110" WHEN "00010000", — 16
"00010101" WHEN "00001111", — 15
"00010100" WHEN "00001110", — 14
"00010010" WHEN "00001100", — 12
"00010000" WHEN "00001010", — 10
"00001001" WHEN "00001001", — 9
"00001000" WHEN "00001000", — 8
"00000111" WHEN "00000111", — 7
"00000110" WHEN "00000110", — 6
"00000101" WHEN "00000101", — 5
"00000100" WHEN "00000100", — 4
"00000011" WHEN "00000011", — 3
"00000010" WHEN "00000010", — 2
"00000001" WHEN "00000001", — 1
"00000000" WHEN "00000000", — 0
"_____ " WHEN OTHERS; END taula_veritat;

```

Pregunta 2: Dissenyeu el mòdul combinacional sel en forma de logigrama amb un nombre mínim de portes lògiques estàndard.

Per a fer la seleccio usarem un multiplexor 2:1 per a busos de 8 bits. Aquest estarà compost per 8 multiplexors 2:1 classics de un bit de la forma



Per tant a una de les entrades de informació se li posara AxB i a l'altre un vector "1111111", ja que si el bit es 1 el segment de la placa esta apagat. I a la entrada de seleccio li posarem show.

Pregunta 3: Dissenyeu el mòdul combinacional keygroup en forma de logigrama amb un nombre mínim de portes lògiques estàndard.
En primer lloc construirem una taula de veritat

Tecla	kc_3	kc_2	kc_1	kc_0	BCD	AST	COI
0	0	0	0	0	1	0	0
1	0	0	0	1	1	0	0
2	0	0	1	0	1	0	0
3	0	0	1	1	1	0	0
4	0	1	0	0	1	0	0
5	0	1	0	1	1	0	0
6	0	1	1	0	1	0	0
7	0	1	1	1	1	0	0
8	1	0	0	0	1	0	0
9	1	0	0	1	1	0	0
A	1	0	1	0	0	0	0
B	1	0	1	1	0	0	0
C	1	1	0	0	0	0	0
D	1	1	0	1	0	0	0
*	1	1	1	0	0	0	1
#	1	1	1	1	0	1	0

D'aquí podem veure rapidament les formules que tindran les funcions AST i COI

$$AST = kc_3 \cdot kc_2 \cdot kc_1 \cdot \overline{kc_0}$$

$$COI = kc_3 \cdot kc_2 \cdot kc_1 \cdot kc_0$$

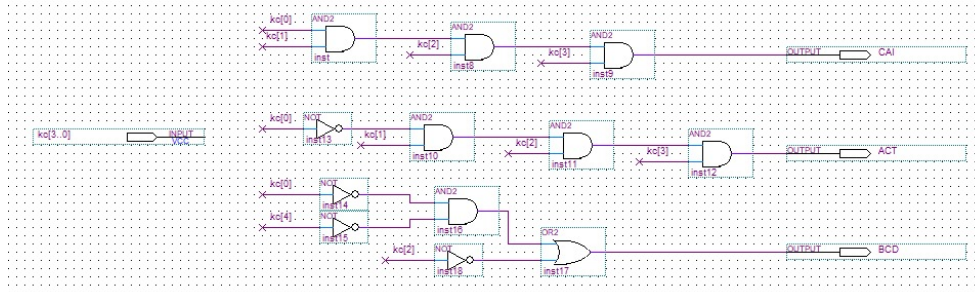
Per a la última sortida, la BCD construirem una taula de Karnaugh i agruparem uns per a trobar suma de minterms.

BCD	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	0	0	0	0
10	1	1	0	0

Veiem que podem fer un grup de 8 uns i un altre grup de quatre, per tant el resultat final es el següent

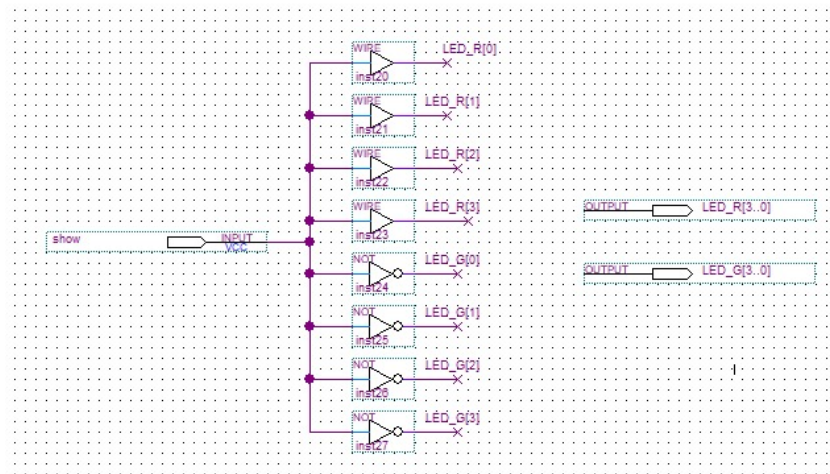
$$BCD = \bar{k}c_3 + \bar{k}c_1 \cdot \bar{k}c_4$$

Ara doncs ja podem construir el disseny amb el mínim de portes and, or i not i tindrà la següent forma



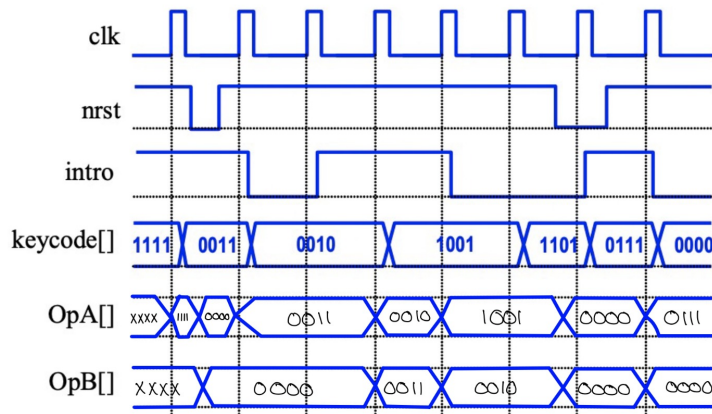
Pregunta 4: Dissenyeu el mòdul combinacional leds en forma de logigrama amb un nombre mínim de portes lògiques estàndard. Aquest mòdul es descriu en l'apartat 3.2.

Aquest modul es ben senzill, nomes caldra rebre la entrada show, i a les sortides de leds vermells treure show, i a les de led verd show negat. Per tant tindrà la següent forma.

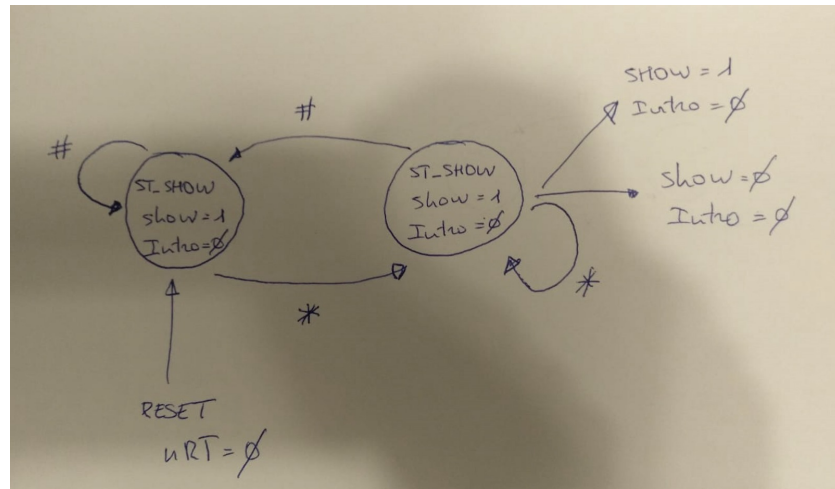


Pregunta 5: Expliqueu quina és la finalitat i com funciona el mòdul regs. Empleneu les línies OpA[] i OpB[] del cronograma-exemple donat a sota.

La finalitat del mòdul regs és guardar dos dígits introduïts per teclat. El funcionament del mòdul és a partir de 2 registres de 4 bits agafà i guarda en la memòria el valor de l'entrada de manera que cada vegada que **intro** està actiu, el valor prèviament guardat al primer registre passa al segon, canviant aquest primer el seu valor per l'introduït al teclat. També quan $\text{nrst} = 0$ els valors de OpA[] i OpB[] es posen a 0.



Pregunta 6: Dibuixeu el diagrama d'estats del mòdul control. Aquest diagrama ha d'especificar tan els canvis d'estat com les sortides.



Pregunta 7: Expliqueu el funcionament del mòdul control. Empleneu les línies state, show i intro del cronograma-exemple donat a sota.

El funcionament del mòdul control es gestionar el mòdul ppal per saber si te que mostrar la informació (*show*) i introduir (*intro*) i també ast i coi ens indica quan estem introduint valors i quan volem que és mostrin i es calculin. Bcd ens indica si hem introduït una lletra o un valor.

