

# Memoria practica 1 DGD

Daniel Vilardell

Igor Yuziv

17-10-2020

Aquesta primera practica es divideix en dues parts, la part a i la part b. La primera part es basa a aprendre com funciona el programa Quartus II a partir de fer un multiplexor 2:1 per a bussos de 4 bits i verificar el seu funcionament sobre la placa DE2. En la segona part, aplicant el coneixement adquirit en la primera, construïm una calculadora que permet multiplicar dos nombres de 4 bits en CA2.

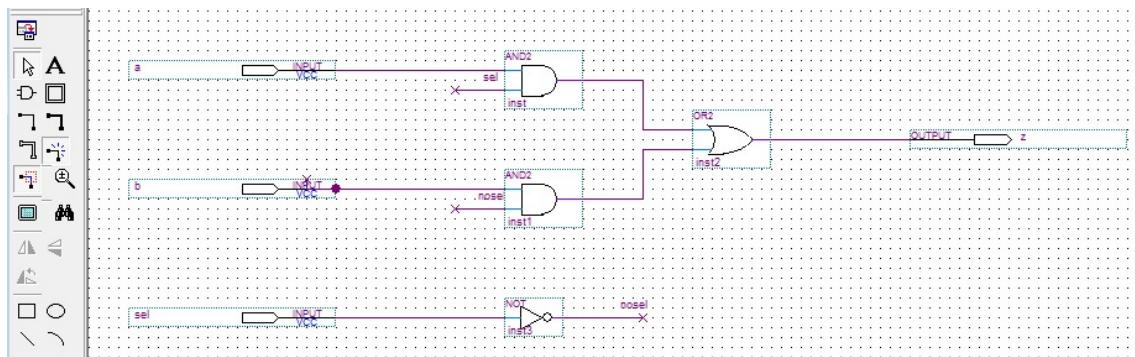
En aquesta memòria explicarem detalladament la construcció de cada component i com s'usa per a assolir els objectius proposats.

# 1 Primera part

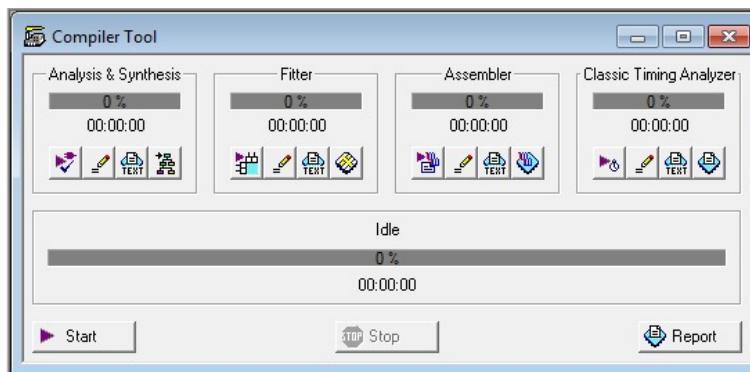
La primera part té l'objectiu de acomodarnos a l'ús del programa quartus a partir de fer un component tan bàsic com seria un multiplexor 4:1. En aquesta memòria s'explicarà detalladament els passos que s'han seguit per a arribar a veure els resultats a la mateixa placa.

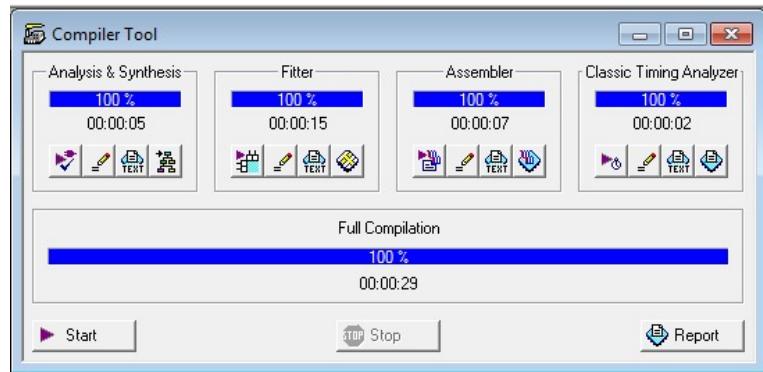
## Multiplexor 2:1

Crearem un multiplexor 2:1 amb el diagrama de blocs:

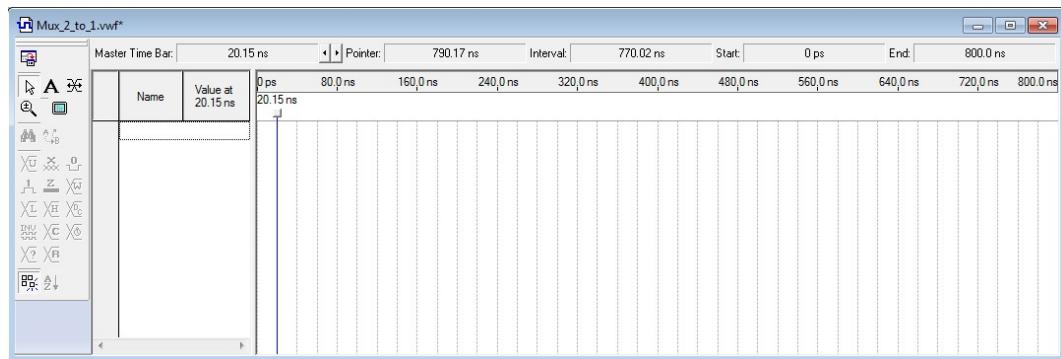


Una vegada descrit el circuit anirem a sintetitzar-lo de forma que pugui ser simulat, validat i gravat en la FPGA. Primer hem de seleccionar la nostra descripció com a entitat de nivell més alt, seguidament fer servir el Compilador.





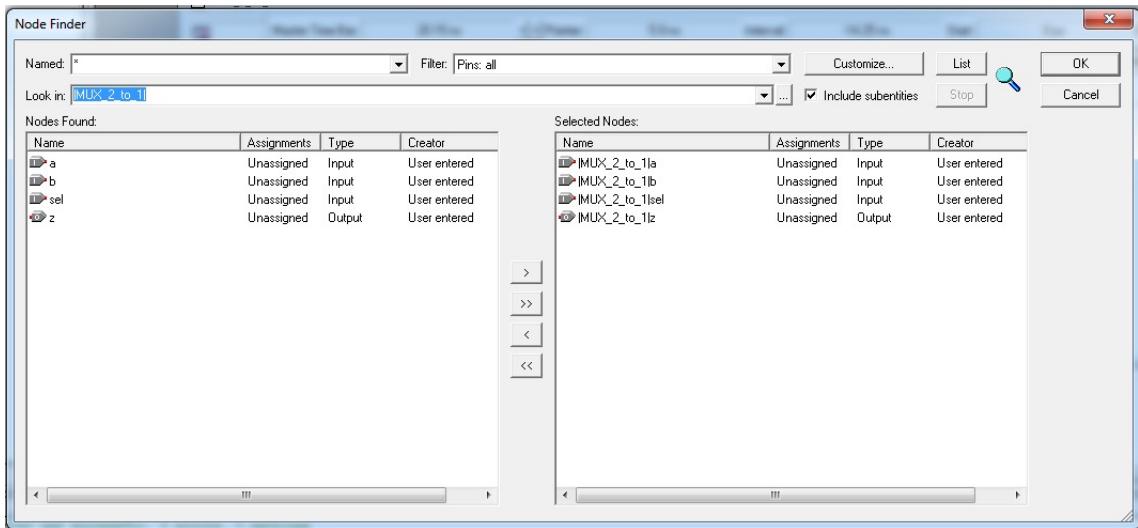
Un cop hem sintetitzat el circuit anirem a simular-lo amb el "Debugging File" que serà "Vector Wave File"



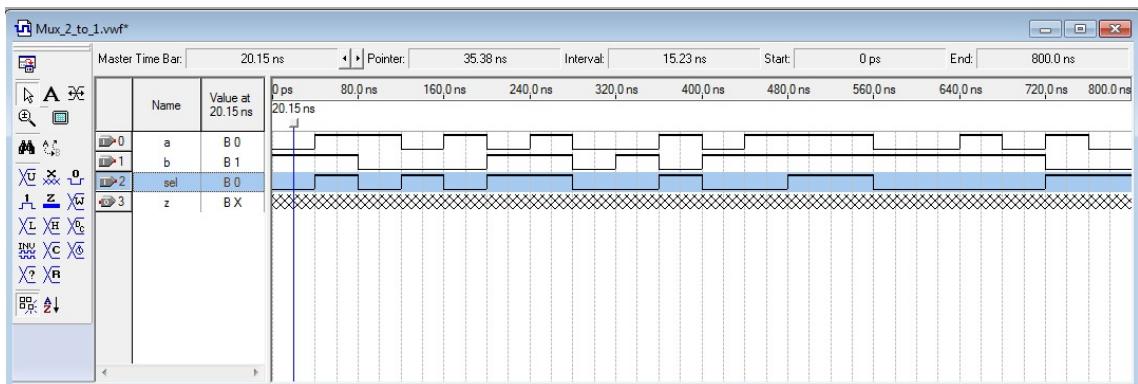
Un cop a qui haurem de seleccionar els nodes que volem que apareguin a la simulació.

Name:	<input type="text"/>	OK
Type:	<input type="button" value="INPUT"/>	Cancel
Value type:	<input type="button" value="9-Level"/>	<input type="button" value="Node Finder..."/>
Radix:	<input type="button" value="Binary"/>	
Bus width:	<input type="text" value="1"/>	
Start index:	<input type="text" value="0"/>	
<input type="checkbox"/> Display gray code count as binary count		

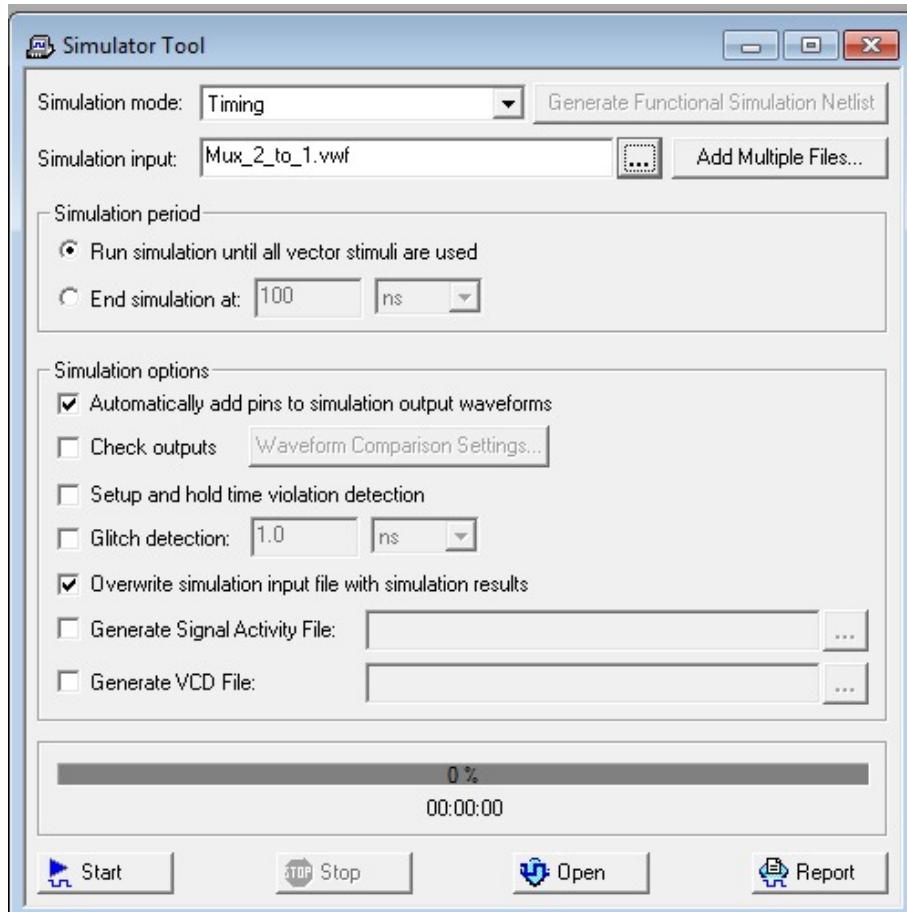
Donarem el botó de Node Finder i seleccionarem els nodes que ens interessa per simular.



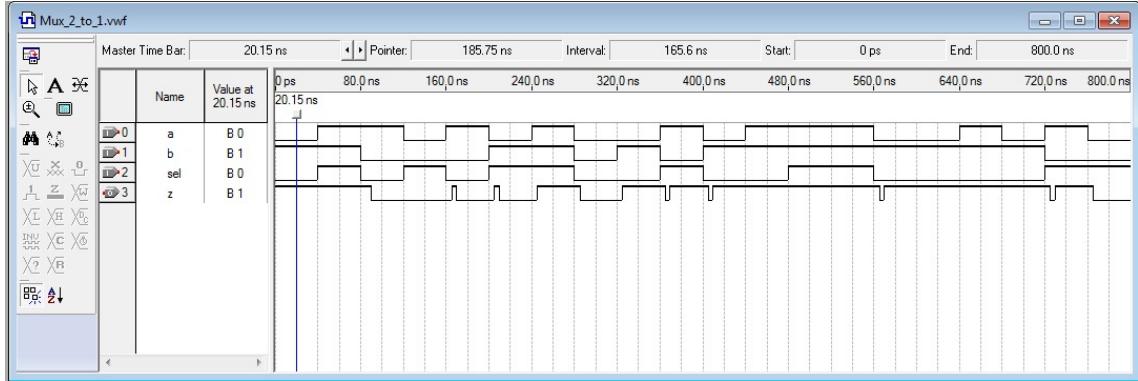
Una vegada que tenim els nodes seleccionats procedirem a canviar els valors dels nodes per poder fer la simulació.



Finalment farem la simulació i mirarem si ens dona el resultat esperat. Per fer la simulació farem servir el "Simulator Tool".

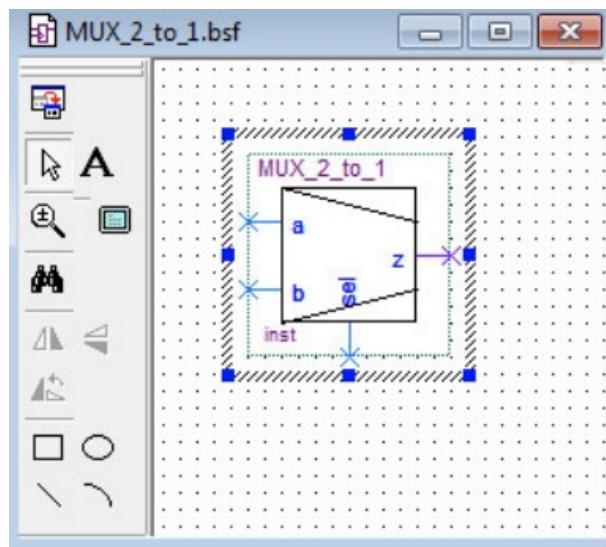


Com es pot veure aquí la simulació ja esta feta i el circuit es comporta com esperàvem.

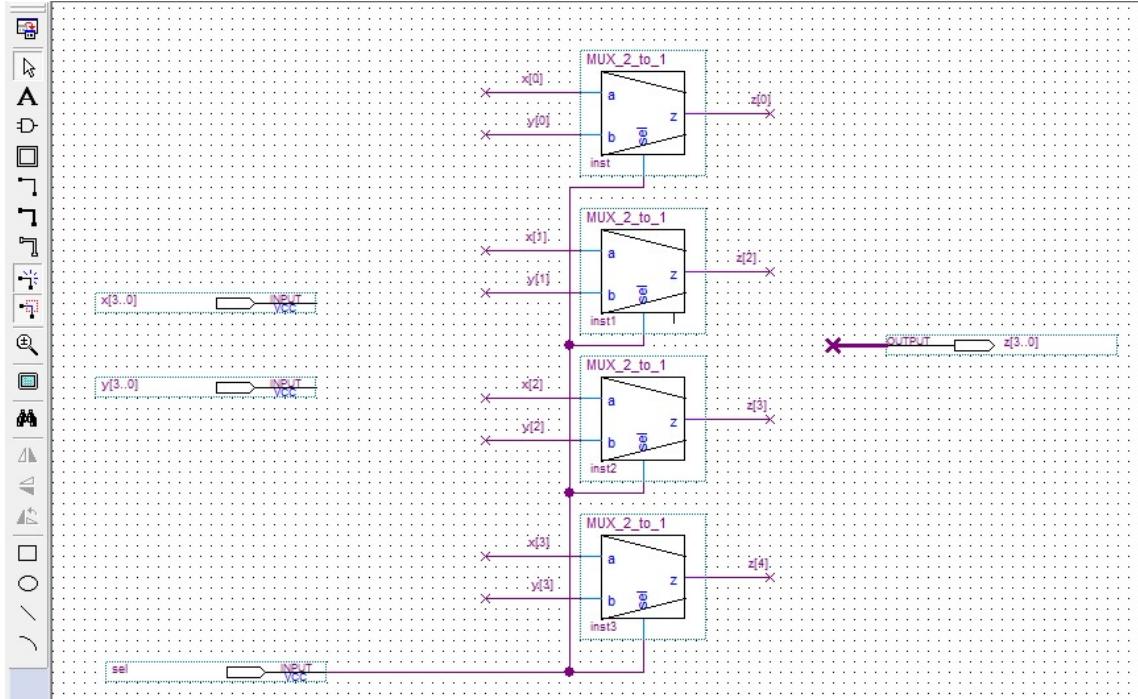


## Multiplexor 2:1 per busos de 4 bits

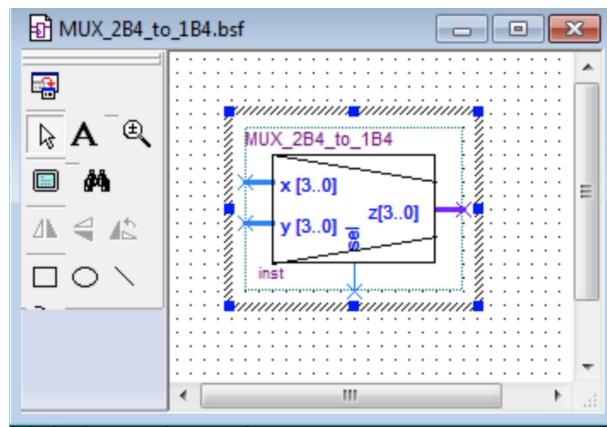
Per fer el Multiplexor primer haurem de crear un nou bloc amb el Multiplexor 2:1.



Un cop tenim el bloc del multiplexor, procedirem a fer el circuit del Multiplexor 2:1 per busos de 4 bits

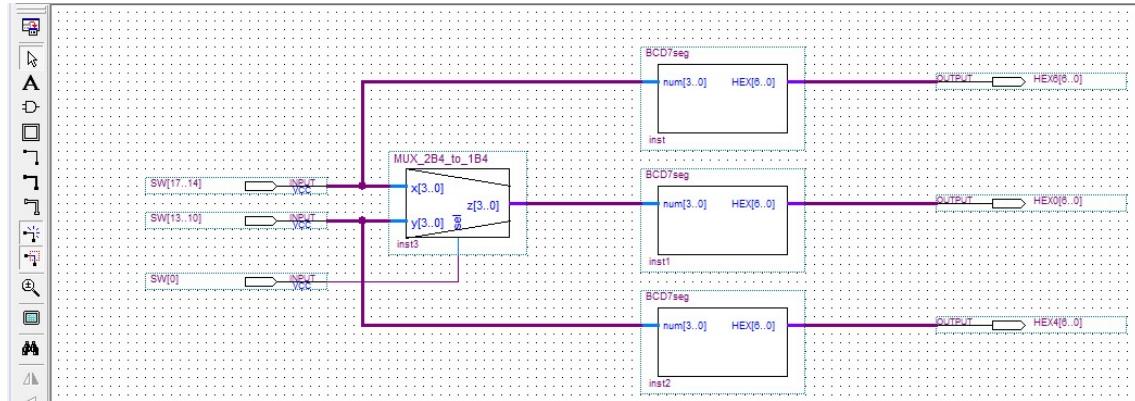


Un cop creat el bloc compilarem i simularem per comprovar el correcte funcionament i crearem un nou bloc per poder utilitzar-lo quan el necessitem.



## Convertidores de BCD a set segments

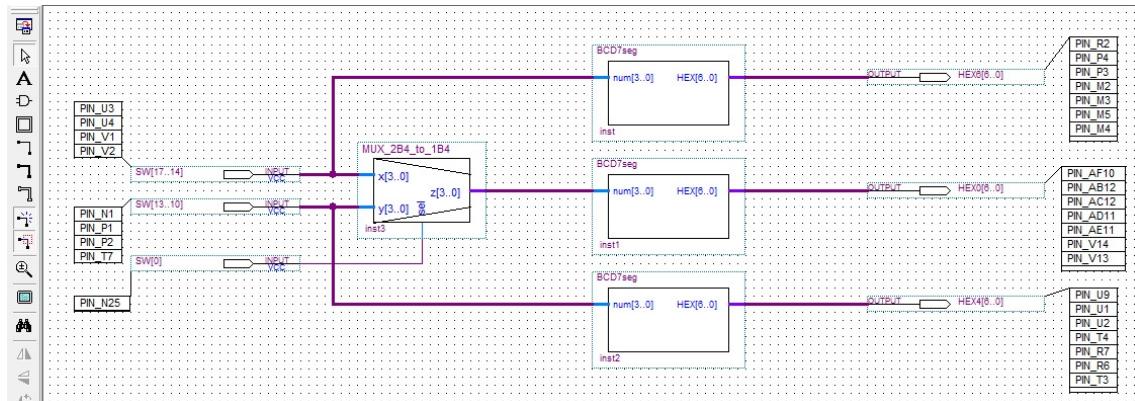
Per poder utilitzar els convertidors de BCD els importarem al projecte i muntarem el circuit, com es mostra a la següent figura.



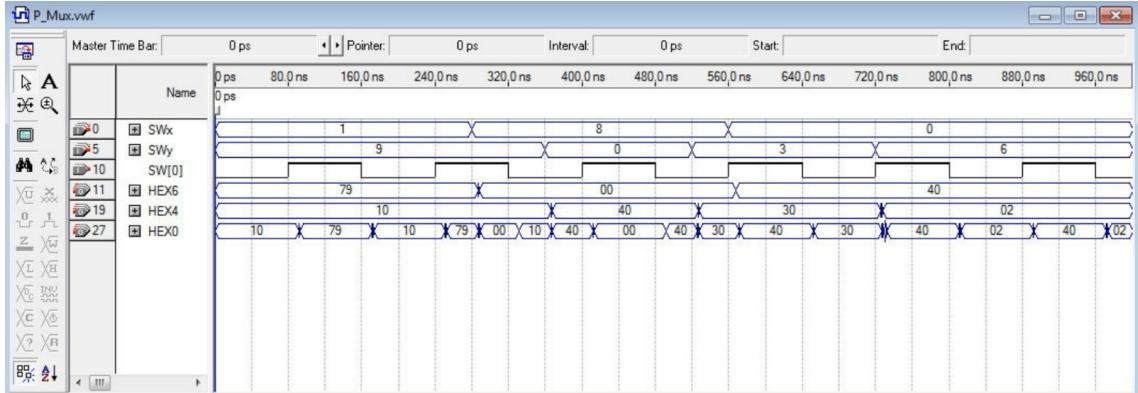
Un cop tenim el circuit complet dissenyat, hem de convertir-lo en el nivell de jerarquia més alt amb "Set as Top-Level Entity".

### Asignació de pins

Per poder simular el nostre circuit a la placa FPGA haurem d'importar l'assignació de pins des del document Excel que ens proporcionen, associa cada pin de la FPGA amb el nom del senyal del component de la placa on està connectat.

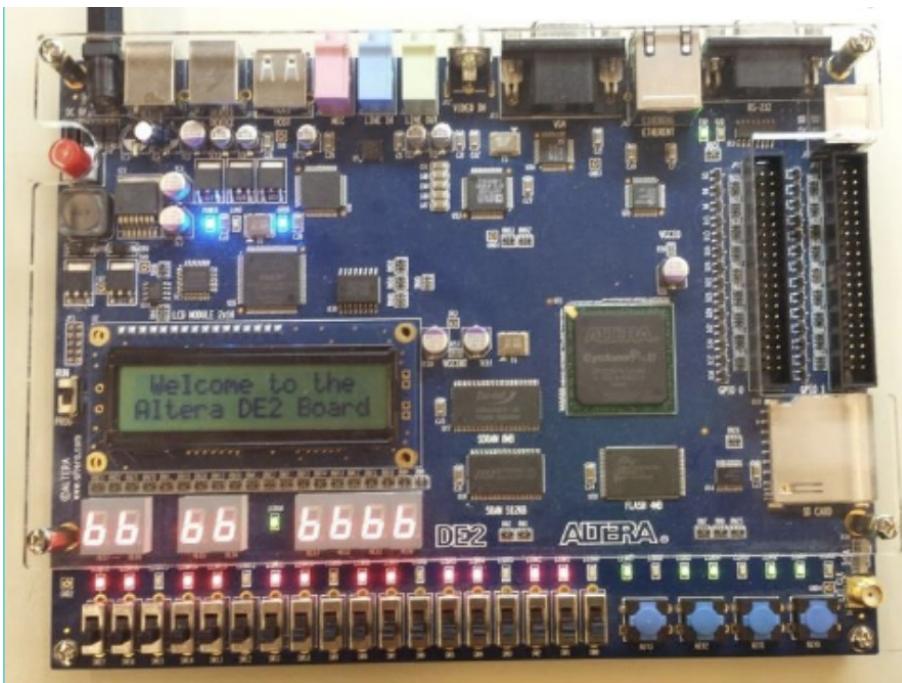


Finalment farem la simulació del disseny complet per comprovar el seu funcionament i poder enviar-lo a la placa FPGA.

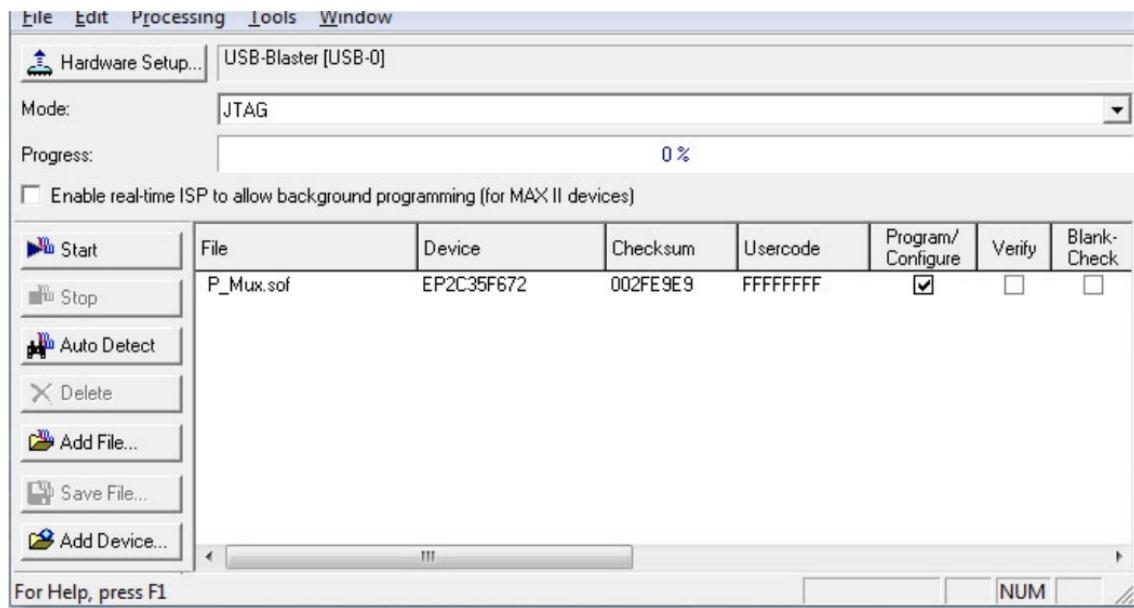


## Validació del disseny sobre la placa DE2

Engegarem la font d'alimentació i posarem la tensió de sortida a uns 8.5V. Connectarem el cable USB a l'entrada USB-Blaster de la placa DE2 i a un port USB del PC. Clicarem l'interruptor vermell de la placa. Un cop encesa procedirem a programar la placa.



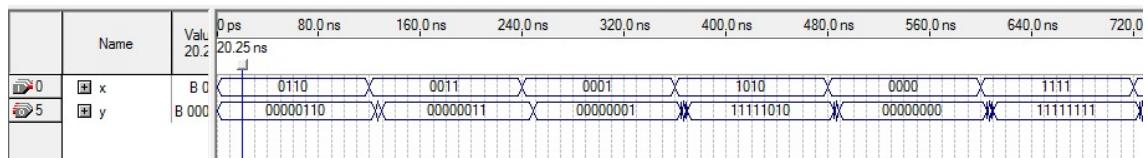
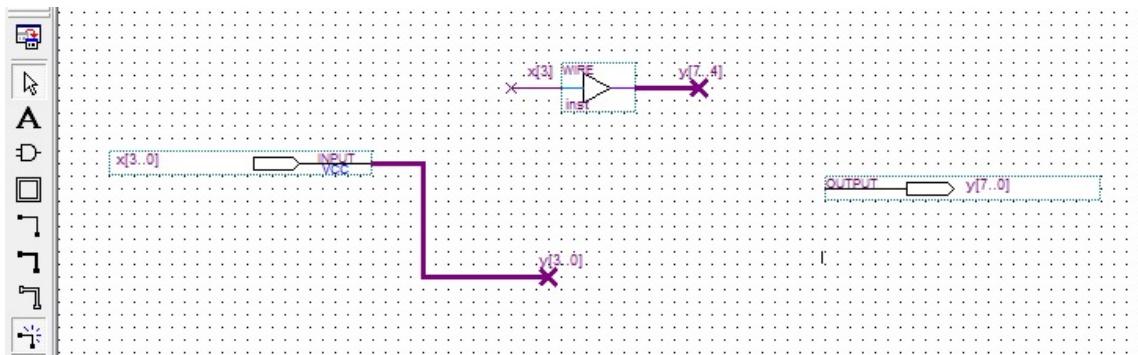
Per poder programar la placa anirem a "tools - Programmer".



## 2 Segona part

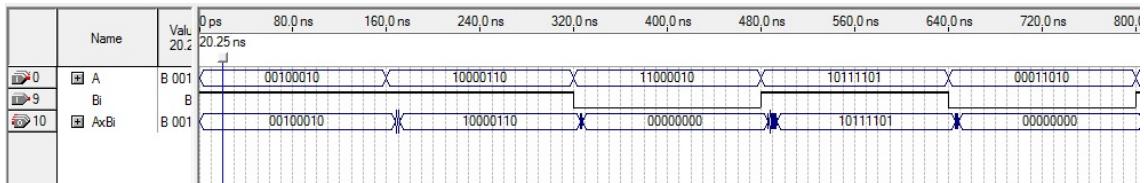
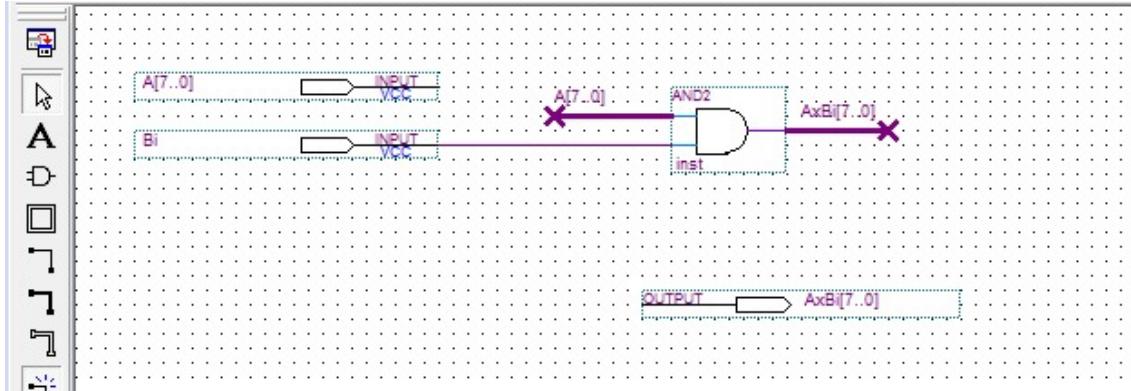
### Convertidor de CA2 de 4 a 8 bits

Aquest component és ben simple i s'usarà per a poder fer el producte de dos nombres en CA2, ja que el resultat de la multiplicació és sempre un nombre de 8 bits. El que fa és agafar el bit de més pes de l'entrada i el repeteix als bits de més pes següents. La raó per la que funciona aquesta idea ja està vista al previ



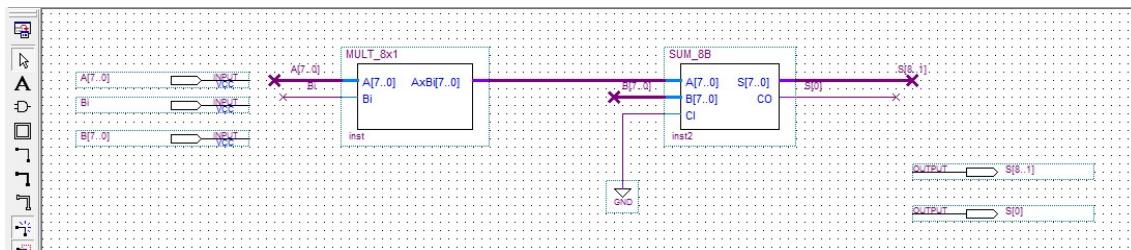
### Multiplicador 8x1

Aquest component multiplica un nombre de 8 bits per un nombre d'1 bit. Per tant la sortida serà el nombre de 8 bits entrat si l'altre nombre és 1, i 0 altrament.



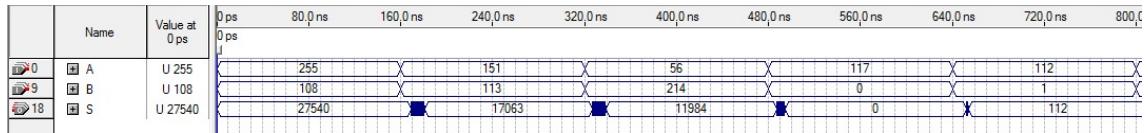
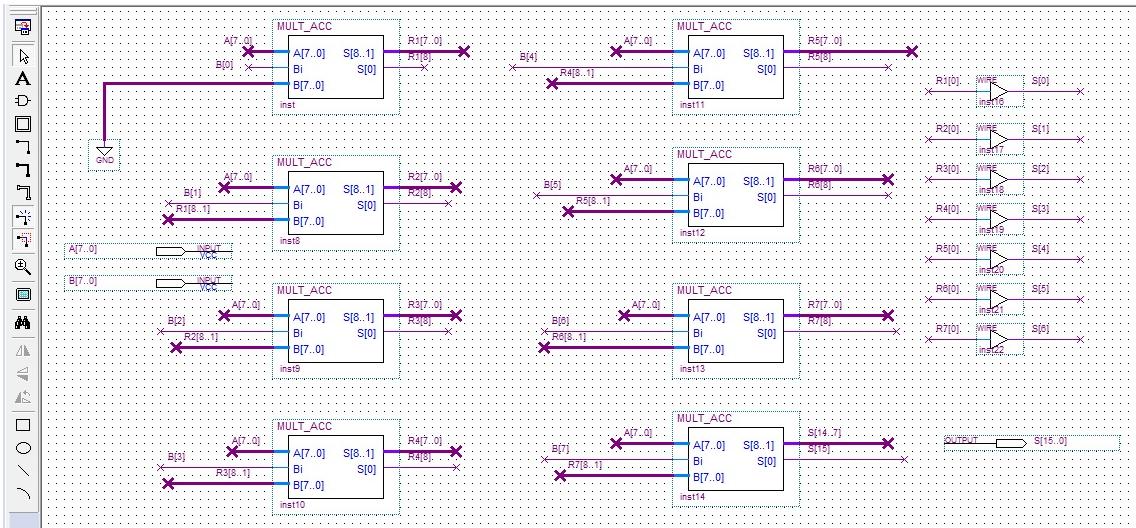
## Multiplicador i sumador

Aquest component té la funció de multiplicar un nombre de 8 bits per un d'1 usant el component mencionat anteriorment per després sumar el resultat amb un altre nombre de 8 bits que ve a l'entrada. Això és farà servir per a fer la multiplicació clàssica que es basa a multiplicar un per un els bits del nombre de baix i sumar amb els anteriors sempre traient el bit de menys pes. És per això que les sortides del component són per una banda el bit de menys pes que s'usarà per a posar al resultat, i per altra els 7 bits de més pes resultants de la suma.



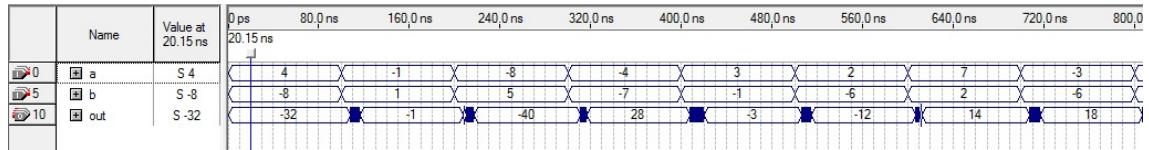
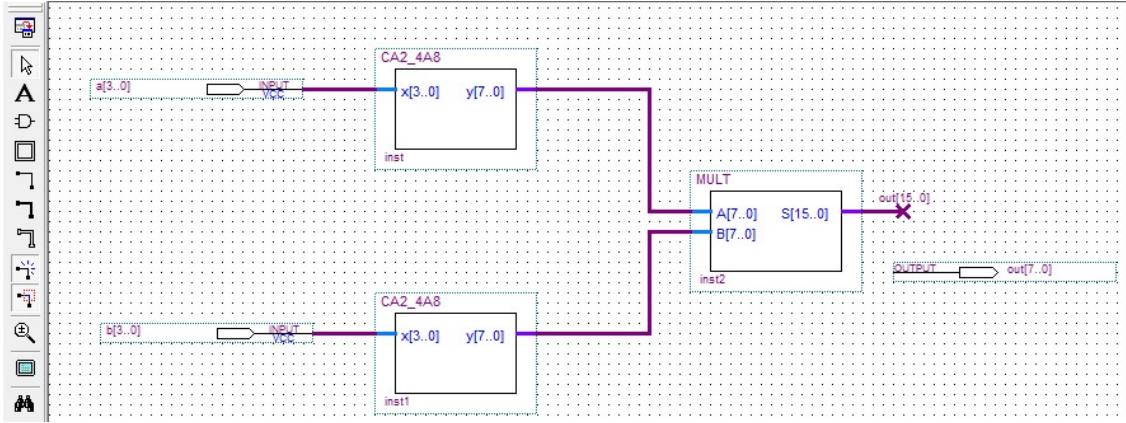
## Multiplicador binari de 8 bits

Aquest component multiplica dos nombres de 8 bits a partir de posar en serie els multiplicadors i sumadors comentats abans. L'entrada són dos nombres de 8 bits, i a cada entrada dels 8 components en serie s'hi posarà el nombre de dalt, és a dir  $A[7..0]$ , el bit que toqui del nombre de baix,  $B[i]$  i finalment se li entrerà la suma de les anteriors multiplicacions extraient el bit de menys pes. Aquest bit de menys pes anirà al nombre de 16 bits que hi ha a la sortida. Finalment quan s'arriba a l'últim component s'introduceix tota la sortida (la suma i el bit de menys pes) a la sortida general del component.



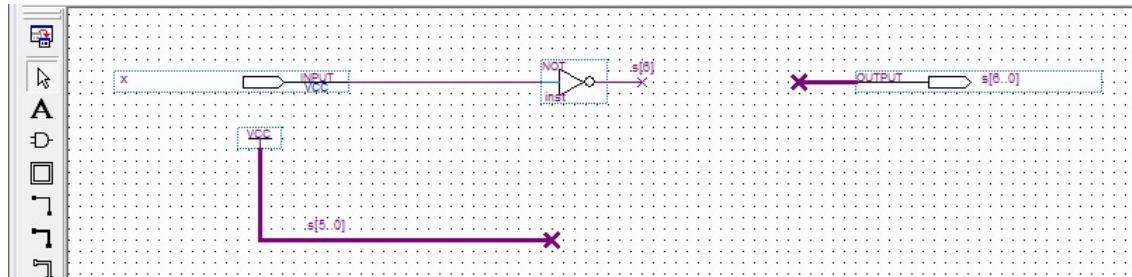
## Multiplicador CA2 de 4 bits

Aquest component multiplica dos nombres en CA2 de 4 bits. Per a fer això primer els transforma a CA2 de 8 bits amb el primer component mencionat, per a després introduir-lo al multiplicador binari de 8 bits. Tot i que la sortida és de 16 bits, només ens interessen els 8 bits de menys pes, ja que en multiplicar dos nombres de 4 bits, el resultat màxim tindrà 8 bits.



## Adaptador de signe

Aquest component té la finalitat de veure si el valor és negatiu o positiu i en el cas que sigui negatiu, encén el segment de la placa corresponent perquè pugui ser visualitzat.



## Convertidor CA2 de 4 bits al seu modular

Aquest component és el que ens permetrà extreure el mòdul d'un nombre en CA2 de 4 bits per tal de representar-lo a la placa.

```

1 -- Exercici 9 practica 1B
2
3 LIBRARY ieee;
4 USE ieee.std_logic_1164.ALL;
5 ENTITY MOD_CA2_4B IS PORT (
6     CA2 : IN STD_LOGIC_VECTOR(3 downto 0 );
7     Modulo : OUT STD_LOGIC_VECTOR(3 downto 0));
8 END MOD_CA2_4B;
9 ARCHITECTURE truth_table OF MOD_CA2_4B IS
10 BEGIN -- enter des de -8 fins a 7
11     with CA2 SELECT Modulo <=
12         "1000" WHEN "1000",
13         "0111" WHEN "1001",
14         "0110" WHEN "1010",
15         "0101" WHEN "1011",
16         "0100" WHEN "1100",
17         "0011" WHEN "1101",
18         "0010" WHEN "1110",
19         "0001" WHEN "1111",
20         "0000" WHEN "0000",
21         "0001" WHEN "0001",
22         "0010" WHEN "0010",
23         "0011" WHEN "0011",
24         "0100" WHEN "0100",
25         "0101" WHEN "0101",
26         "0110" WHEN "0110",
27         "0111" WHEN "0111";
28     END truth_table;
29

```

## Multiplicador final per a visualització

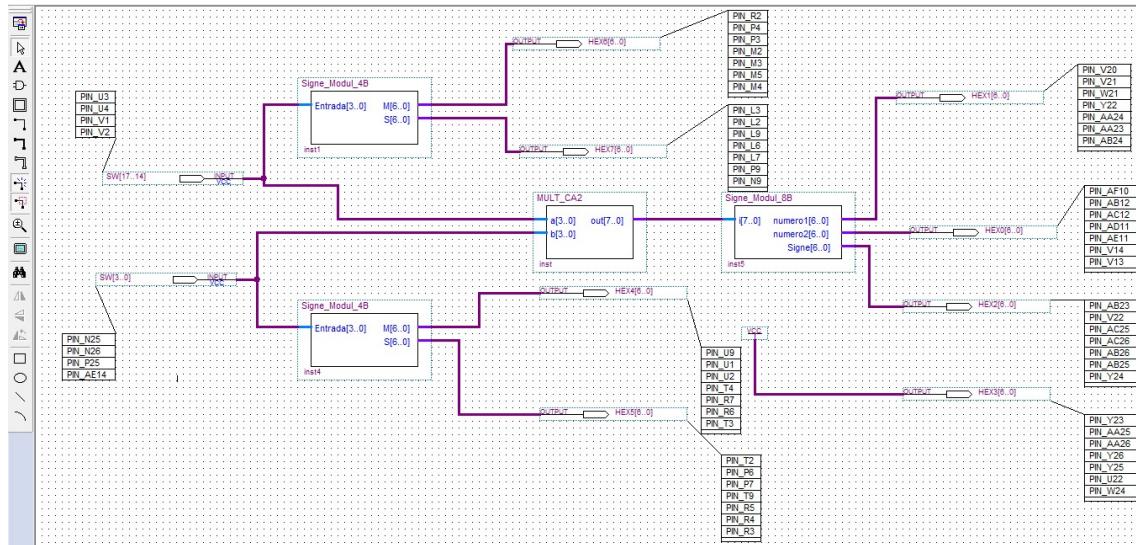
Aquesta multiplicació final té tres branques, una per a la visualització del primer nombre entrat, una per a la visualització del segon i finalment una per a visualitzar el resultat. L'entrada serà de dos nombres de 4 bits als quals assignarem pins de la placa.

La primera branca i la segona funcionen igual així que només explicarem una d'elles.

- Quan s'entra un nombre en CA2 de quatre bits, primer determinem el signe l'adaptador de signe. Aquesta sortida ja va directe a la placa i mostra el signe.
- Després de determinar el signe es converteix el nombre en CA2 al seu mòdul.
- Quan ja tenim el mòdul l'introduïm a un component facilitat per l'assignatura que converteix nombres binaris de 4 bits a BCD de 7 segments. Aquesta sortida també va directe a la placa i mostra el nombre.

La tercera i última branca funciona de la següent manera:

1. Primer s'introdueixen els dos nombres al multiplicador CA2 de 4 bits, el qual dóna el seu producte en CA2.
2. S'extreu el signe a partir del bit de més pes i es mostra a la placa usant l'adaptador de signe.
3. La sortida del producte s'introdueix dins del component facilitat també per l'assignatura que et dóna el BCD d'un nombre de 8 bits.
4. Els primers 4 bits seran el primer nombre de la sortida, així doncs aquest nombre es passa a BCD de 7 segments i es mostra a la placa.
5. Finalment es fa el mateix pels 4 últims bits, els de menys pes, i es mostra a la placa.



### 3 Tercera part(extra)

Un cop implementada la calculadora que permet fer el producte de dos nombres en CA2, muntarem un component que a partir de tres entrades, A i B i selop tregui la sortida següent:

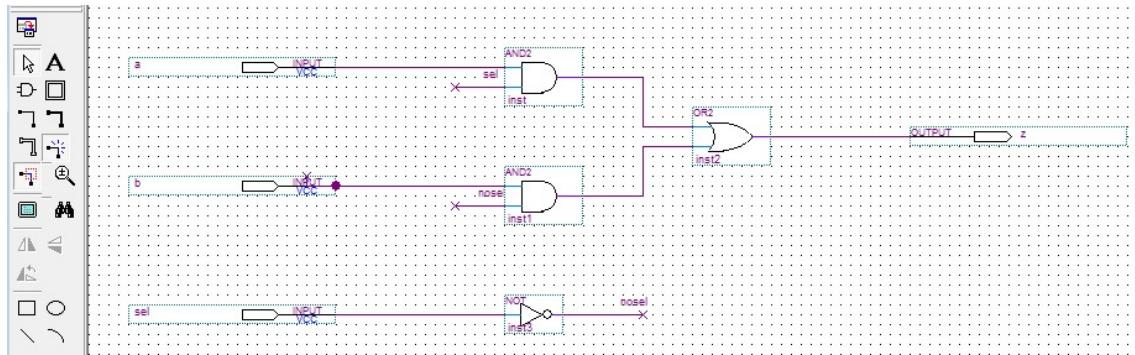
- $A \cdot B$  si selop = 00
- $A^2$  si selop = 01
- $B^2$  si selop = 10
- $2^A \cdot 2^B$  si selop = 11

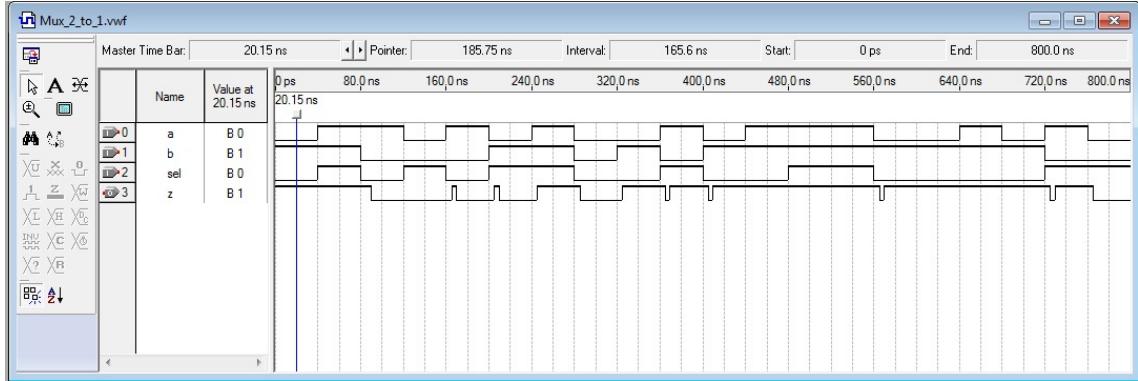
Cal tenir en compte en primer lloc que la sortida és de 8 bits, i per tant la sortida  $2^A \cdot 2^B$  no podrà mostrar totes les sortides possibles. Tenint en compte que com la sortida és en CA2, només disposem de 7 bits i per tant només podrem representar totes aquelles sortides tals que  $A + B \leq 7$ . A més a més, com que només podem representar nombres enters, la suma de  $A + B \geq 0$ . Vist això ara cal pensar com farem el disseny del circuit. En el nostre cas primer escollirem quins dos nombres multiplicar, és a dir, si selop és 00, escollirem per una banda A i per altra banda B. Si selop és 01 escollirem A i A, si selop és 10 escollirem B i B. Finalment si selop és 11, com que no ens importa el resultat del producte ja que la sortida vindrà d'un altre component que farà el càlcul  $2^A \cdot 2^B$ , multiplicarem  $0 \cdot 0$ .

Per a fer la selecció primer fem un multiplexor 2x1, que usarem per a construir altres components més grans.

#### Multiplexor 2x1

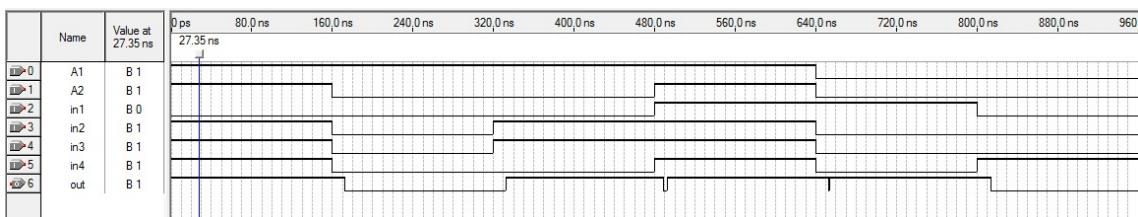
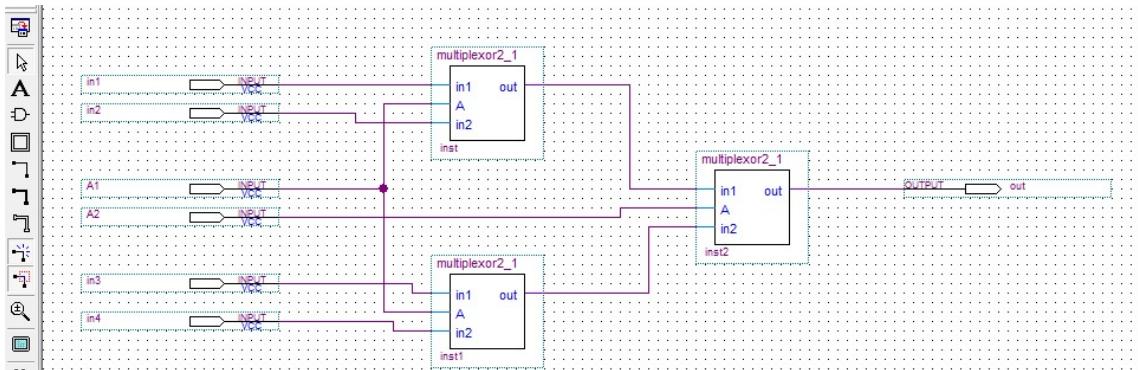
Aquest component és ben senzill, té tres entrades de 1 bit: A, B, i X. Si  $X = 0$  la sortida serà A, si  $X = 1$  la sortida serà B.





## Multiplexor 4x1

Aquest és igual de senzill que l'anterior. Se li donen sis entrades, quatre d'informació i dos de selecció. En funció de les de selecció s'extreu una o un altre de les de informació per la sortida.



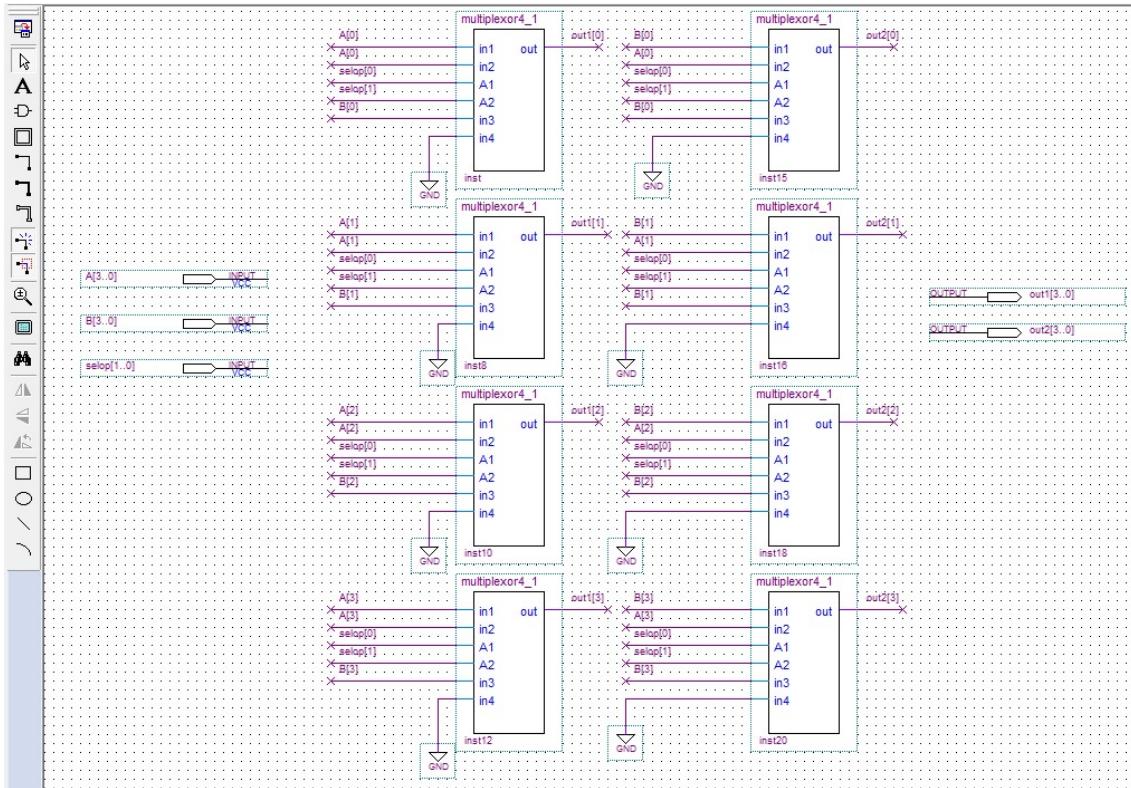
## Selector mult

Amb el multiplexor 4x1 ja en vam tenir prou per a dissenyar el component que s'encarregaria de seleccionar i retornar els dos noms que s'estan

multiplicant. L'entrada del component són dos nombres de 4 bits A i B, i selop, de dos bits.

Per a construir el primer nombre de la sortida usarem els multiplexors, a les entrades els hi posarem  $A[i]$ ,  $A[i]$ ,  $B[i]$  i 0 on i es cada xifre. La sortida la dirigirem al dígit  $i$  del vector de 4 bits de la sortida. A l'entrada de selecció hi posarem selop. Per tant si selop és 00 el primer component que traurà serà  $A$ , si és 01 traurà  $A$ , si és 10 treurà  $B$  i 0 si és 11, per la raó comentada abans.

La segona sortida serà construïda de la mateixa forma però a l'entrada del multiplexor se li posarà  $B[i]$ ,  $A[i]$ ,  $B[i]$  i 0.

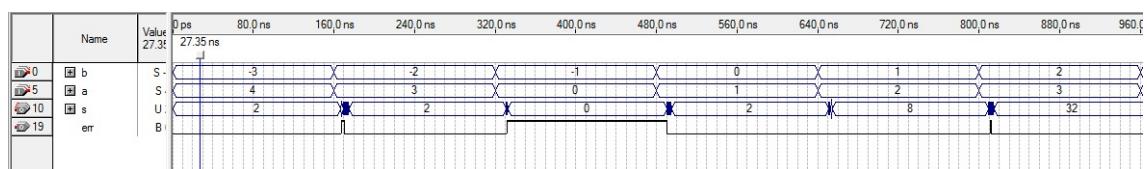
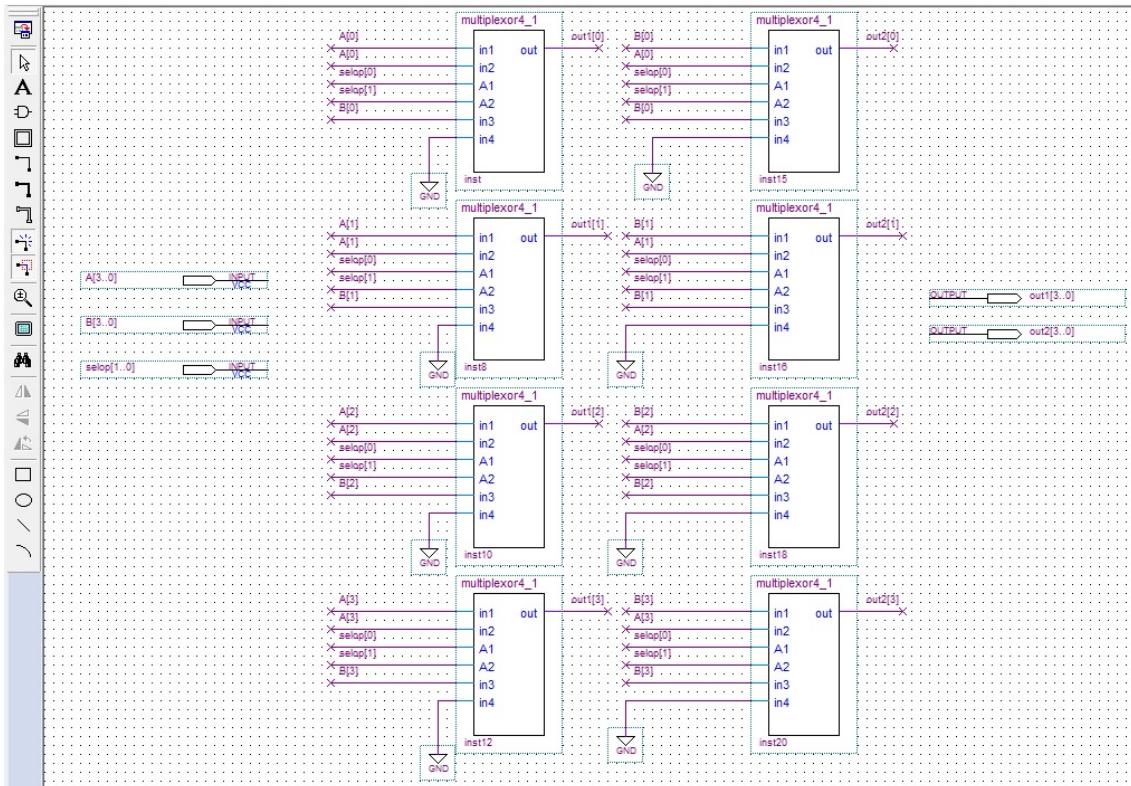


## Elevador

Aquest component té la funció de donades dues entrades  $A$  i  $B$  si  $A + B < 7$  o  $A + B < 0$  treure dues sortides, una que sigui 0 i l'altra serà 1, aquesta segona serà la que permetrà que el led s'il·lumini. Si la suma està dins dels

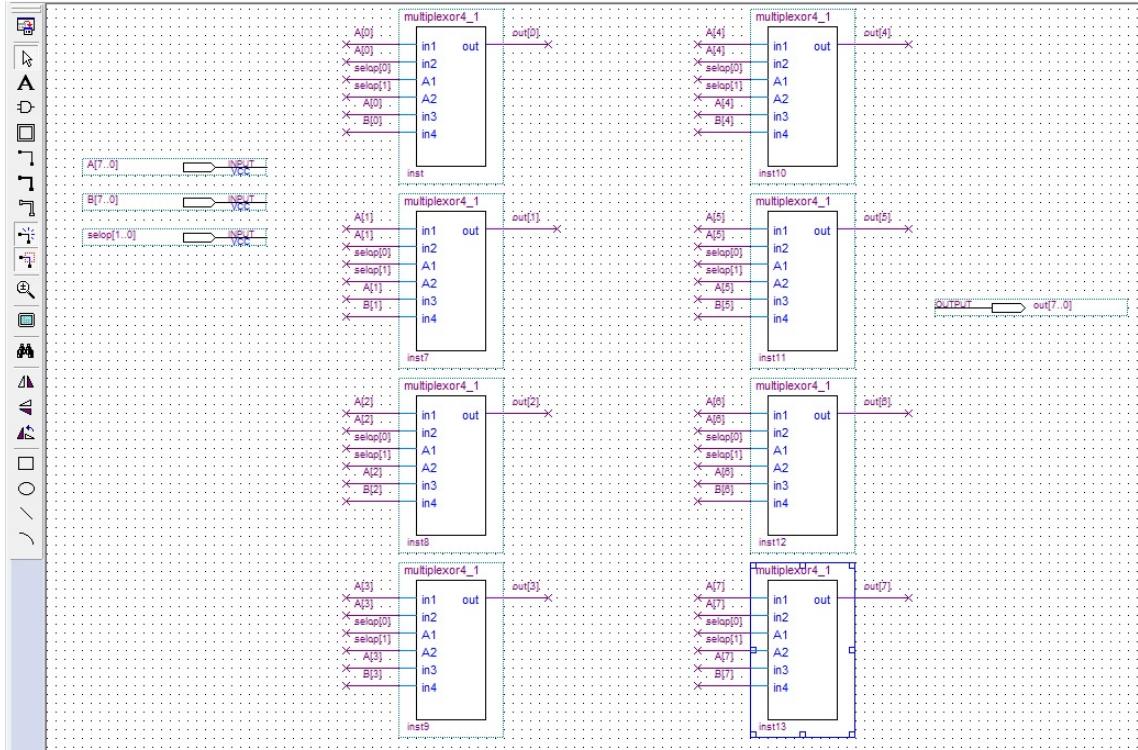
valors permesos, la primera sortida serà  $2^A \cdot 2^B$  i la segona sortida serà 0, ja que en aquest cas el bit no s'hi haurà d'il·luminar.

Aquest component ha estat fet amb vhdl per comoditat.



## Multiplexor 2x1 per entrades de 4 bits

Aquest component ens servirà per a simplificar el disseny final de la calculadora. Fa el següent: Donats dos nombres de 4 bits  $A$  i  $B$  i un nombre de selecció de 2 bits selop, si selop és 00, 01 o 10 la sortida serà  $A$ , si selop és 11 la sortida és  $B$ .



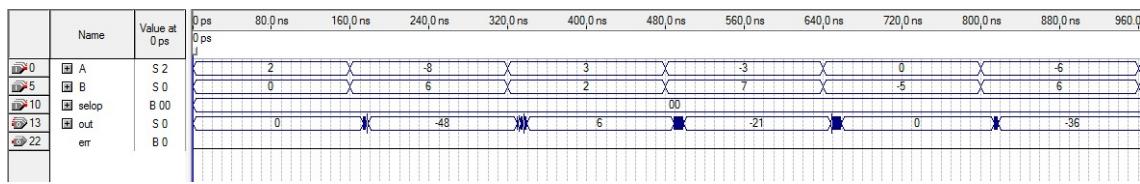
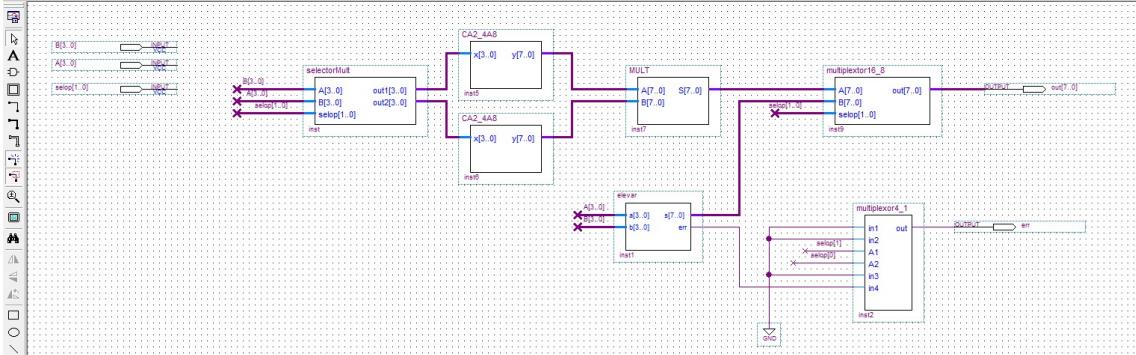
## Calculadora

Aquest component serà el que ens farà el càcul. Donades 3 entrades,  $A$ ,  $B$  i selop primer es passaran les tres entrades pel component selector mult, que ens retornarà quins dos nombres multiplicar. Aquestes dues sortides amb l'ajuda de CA2 4 - 8 bits passarem les 2 sortides de 4 a 8 bits en CA2. Aquestes sortides s'introduiran al multiplicador que ens retornarà el seu producte en CA2.

Per altra banda, introduirem  $A$  i  $B$  a l'elevador del qual hi haurà dues sortides. La segona d'elles ens informarà si hi ha d'il·luminar el led en el cas que selop fos 11. Per a veure si és 11 o no usarem un multiplexor d'1 bit on les tres primeres entrades estaran connectades a terra i l'última entrada estarà connectada a la sortida del bit d'error de l'elevador. Les entrades de selecció seran selop. La sortida del multiplexor ens dirà si ha d'il·luminar el bit d'error o no.

Ara usarem un multiplexor 2x1 per entrades de 4 bits a on se li posarà la sortida del multiplicador a la primera entrada, la sortida de l'elevador a la segona entrada i selop a les entrades de selecció. La sortida d'aquest

component serà el valor buscat.



## Simulador calculadora

Aquest component és el que enviarem a la placa, per tant te l'objectiu de llegir les entrades dels pins i mostrar els resultats per pantalla.

Consta de tres entrades,  $A$   $B$  i selop. De la mateixa manera que en la segona part de la pràctica es mostraran les entrades  $A$  i  $B$  a la placa.

Les entrades s'introduiran dins del component calculadora. Aquest ens extraurà el càlcul per una banda, que després de passar-lo pel convertidor de nombre de 8 bits en CA2 a BCD de 7 segments l'enviarem a la sortida perquè sigui mostrat a la placa. Per altra banda ens extraurà si ca encendeix el bit d'error en el càlcul o no, que enviarem també a la placa directament.

