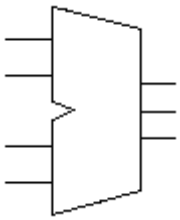
Esercitazione n° 3

Lo scopo di questa esercitazione è di introdurre alla creazione di un disegno di progetto di tipo gerarchico, nel quale cioè si creino vari elementi per comporre via via oggetti di complessità crescente. Anche in questo caso verrà mostrato come effettuare la simulazione con il tool ModelSim. Nei successivi progetti sarà cura dello studente eseguire autonomamente la simulazione.

Problema

Realizzare mediante un file sorgente di tipo schematico un sommatore a 2 bit del tipo seguente



Ciascun bit in ingresso sia controllato da uno switch (dipsw1, dipsw2, dipsw3, dipsw4) e il risultato della somma venga visualizzato sul display a 7 segmenti mediante una codifica di tipo decimale.
Gli switch dipsw1 e dipsw2 siano riservati al primo numero (con dipsw1 al bit meno significativo) mentre dipsw3 e dipsw4 al secondo (con dipsw3 al bit meno significativo).
Ad esempio la somma: 01 + 01 avrà dipsw1 e dipsw3 allo stato ON e gli altri switch allo stato OFF; sul display dovrà comparire il risultato 2.

Suggerimenti

- Realizzare prima il circuito che opera la somma, ad esempio utilizzando il full adder visto nella precedente esercitazione. A tale scopo, per ogni file 'source' è possibile creare con l'opzione 'Create Schematic Symbol' in 'Process View' un simbolo che sintetizza l'oggetto programmato. In questo modo l'oggetto potrà essere utilizzato all'interno di uno schematico semplicemente selezionando nel menu 'Symbols', 'Categories' la categoria con il nome della directory del progetto (es. c:/Xilinx/bin/som2bit+dec).

- In secondo luogo si realizzi il decoder che comanda il display a 7 segmenti. È consigliabile programmare il decoder in linguaggio VHDL. A tal proposito si provino a consultare le librerie accessibili in 'Edit'→ 'Language Templates'.

- Infine si realizzi uno schematico che colleghi vari oggetti.

Soluzione

Si inizi come sempre creando un nuovo progetto dal nome ‘som2bit+dec’ per esempio.
Seguendo quanto suggerito, procediamo con la realizzazione del sommatore a 2 bit.

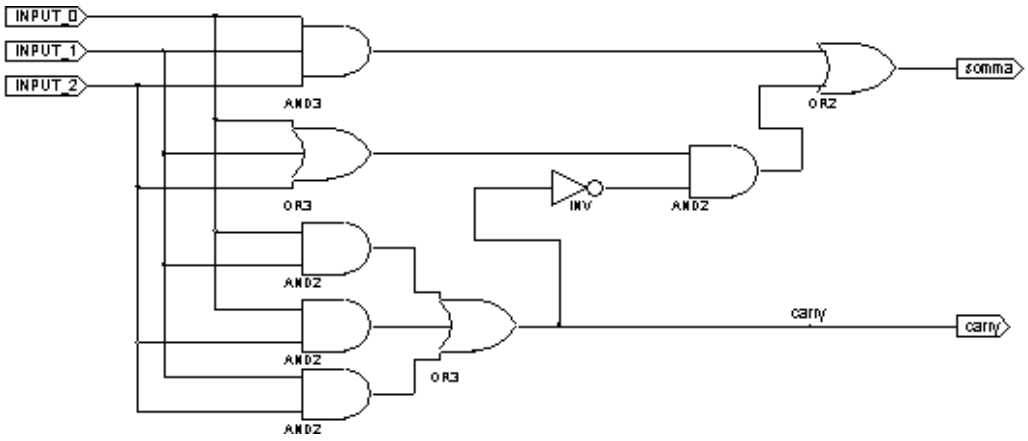
Realizzazione del sommatore 2 bit

Come già detto, utilizzeremo quanto fatto nella precedente esercitazione, eventualmente con qualche piccola modifica.

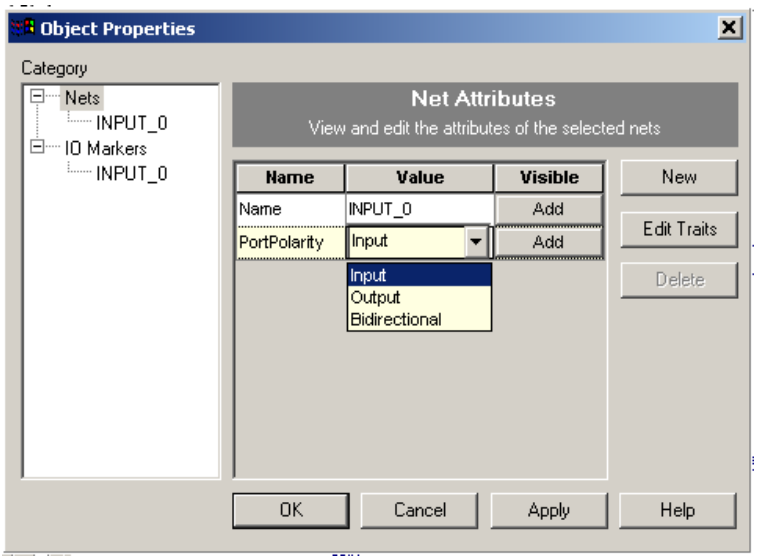
- Si crei un nuovo file sorgente: Project → New Source, Schematic dal nome ‘somma3bit_schem_no_inv’. Si aprirà la finestra dell’ECS.

- Dall’ECS si apra lo schematico che nella precedente esercitazione abbiamo chiamato ‘somma3bit_schem’ (il file si trova nella directory relativa al progetto della precedente esercitazione). Lo si copi nel file corrente ‘somma3bit_schem_no_inv’ nel seguente modo: Edit → Select All → Edit → Copy, poi si selezioni dal menu a tendina il file ‘somma3bit_schem_no_inv’ , infine Edit → Paste.

- Dal momento che avremo bisogno di iterare la struttura del full adder più volte per costruire il sommatore a 2 bit, leveremo, a partire dal disegno appena copiato, gli invertitori posti subito dopo gli Input Marker ottenendo il seguente schematico. Salviamo il tutto.

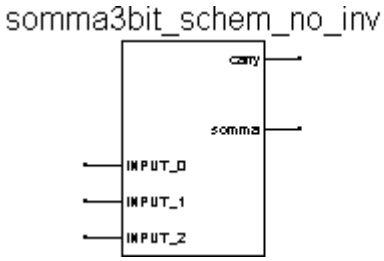


Nel rimuovere gli invertitori dal vecchio schematico, si faccia attenzione alla polarità degli Input Marker: deve essere ‘Input’; per verificare è sufficiente fare doppio click sul simbolo dell’ I/O Marker e controllare la ‘Port Polarity’:



Si ricordi che gli invertitori verranno reintrodotti nello schematico finale, subito dopo gli Input Marker collegati ai dipsw.

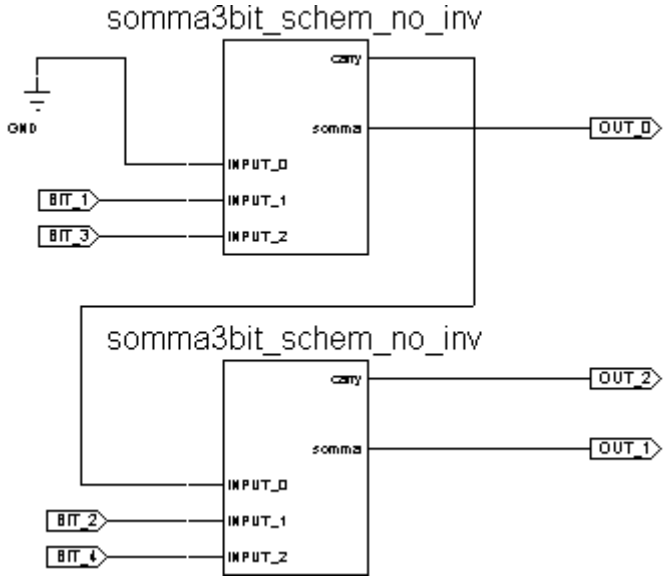
- È ora necessario creare il simbolo relativo all'oggetto 'somma3bit_schem_no_inv'. Per farlo basta operare come segue dal Project Navigator: selezionare con un click il file source schematico 'somma3bit_schem_no_inv' dal Module View (in alto a sinistra), poi doppio click su 'Create Schematic Symbol'. A questo punto un tick verde si accenderà in corrispondenza della scritta 'Create Schematic Symbol'; ciò significa che il tool ha creato un simbolo che descrive l'oggetto 'somma3bit_schem_no_inv', con lo stesso numero d'ingressi e di uscite; qualcosa di simile a:



Per vederlo basta andare nell'ECS, menu 'Symbols', nella 'Category' con la directory dell'attuale progetto (ad es. c:/Xilinx/bin/som2bit+dec).

Si proceda quindi con la creazione del sommatore a 2 bit vero e proprio.

- Creare un nuovo file sorgente di tipo schematico (Project → New Source, Schematic) dal nome 'sommatore2bit'. Nell'ECS si realizzi il seguente schematico:



Si tratta dello schema finale del sommatore a 2 bit. Anche in questo caso si crei il simbolo relativo all'oggetto in questione con un doppio click su 'Create Schematic Symbol'.

Ora bisogna progettare il decoder che comandi il display a 7 segmenti.

Realizzazione del decoder

- Il metodo più facile per realizzarlo è quello di utilizzare il linguaggio VHDL, pertanto si aggiunga al progetto un file sorgente di tipo VHDL nel seguente modo: Project → New

Source → VHDL Module, a cui si assegni ad esempio il nome ‘dec7segm3bit’. Poi ‘Avanti’.

In accordo con le regole del linguaggio VHDL compare una finestra relativa alla ‘VHDL Source’ in cui è possibile specificare ingressi e uscite del dispositivo, in particolare è possibile: assegnare il nome delle porte (Port Name); la direzione del flusso di dati (Direction) che può essere in, out o inout; in relazione alla porta in questione, assegnare all’interno del flusso di dati (che è sostanzialmente un vettore), la posizione del bit più significativo (MSB – Most Significant Bit) e del bit meno significativo (LSB – Least Significant Bit). Una volta specificati i valori desiderati, il tool scrive automaticamente l’entity nel listato VHDL.

Nel nostro caso avremo bisogno di una porta input CLK, di una porta input D_IN con MSB pari a 2 e LSB pari a 0, e di una porta output D_OUT con MSB pari a 6 e LSB pari a 0. L’ingresso CLK sarà riservato al clock, il vettore D_IN conterrà i dati provenienti dal sommatore (contenuti in 3 bit), mentre il vettore D_OUT conterrà i 7 bit che comanderanno l’accensione dei 7 segmenti.

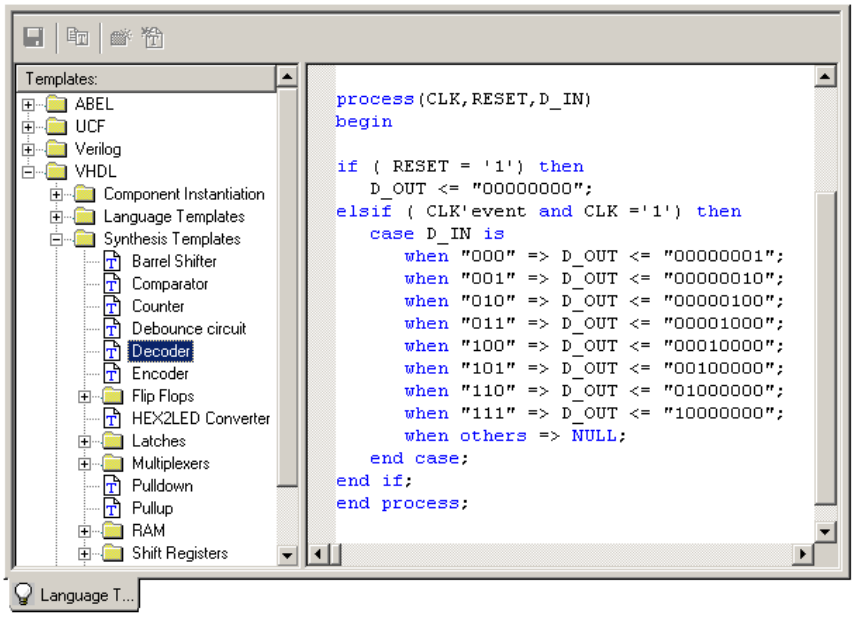
In verità non vi è alcun motivo per realizzare un decoder di tipo sincrono. Si può piu' semplicemente realizzarne uno asincrono che non abbisogna di CLK

Ne uscirà un listato così fatto:

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_ARITH.ALL;
4  use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6  --  Uncomment the following lines to use the declarations that are
7  --  provided for instantiating Xilinx primitive components.
8  --library UNISIM;
9  --use UNISIM.VComponents.all;
10
11 entity dec7segm3bit is
12     Port ( CLK : in std_logic;
13            D_IN : in std_logic_vector(2 downto 0);
14            D_OUT : out std_logic_vector(6 downto 0));
15 end dec7segm3bit;
16
17 architecture Behavioral of dec7segm3bit is
18
19 begin
20
21
22 end Behavioral;
23
```

Il tool di sviluppo è dotato di alcune librerie di listati di dispositivi. Si sfrutterà pertanto il componente decoder che risulta essere presente in tali librerie. Per accedere al listato VHDL del decoder: Edit → Languages Templates nella directory VHDL\Synthesis Templates\ fare click su decoder:

Cercando meglio si può trovare già pronto il "template" per la realizzazione del decoder a sette segmenti. Attenzione pero' a verificare che la numerazione dei segmenti coincida con quella riportata nel file di vincoli.



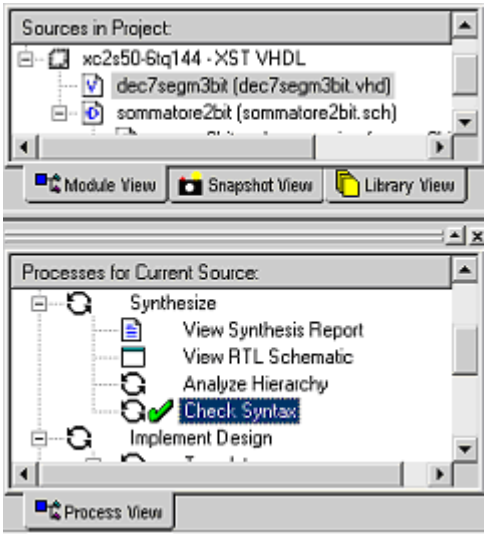
Innanzitutto elimineremo il reset. Inoltre la codifica relativa non è quella giusta per una codifica decimale su decoder a 7 segmenti; è necessario pertanto modificare il listato, che alla fine dovrà essere:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

--  Uncomment the following lines to use the declarations that are
--  provided for instantiating Xilinx primitive components.
--library UNISIM;
--use UNISIM.VComponents.all;

entity dec7segm3bit is
    Port ( CLK : in std_logic;
           D_IN : in std_logic_vector(2 downto 0);
           D_OUT : out std_logic_vector(6 downto 0));
end dec7segm3bit;

architecture Behavioral of dec7segm3bit is

begin
process (CLK,D_IN)
begin

if ( CLK'event and CLK ='1') then
   case D_IN is
      when "000" => D_OUT <= "1110111"; --0
      when "001" => D_OUT <= "0010010"; --1
      when "010" => D_OUT <= "1011101"; --2
      when "011" => D_OUT <= "1011011"; --3
      when "100" => D_OUT <= "0111010"; --4
      when "101" => D_OUT <= "1101011"; --5
      when "110" => D_OUT <= "1101111"; --6
           when others => NULL;
   end case;
end if;
```

end process;

end Behavioral;

Ora per assicurarci che il listato sia corretto, lo si salvi e si verifichi la sintassi facendo doppio click su 'Check Syntax':

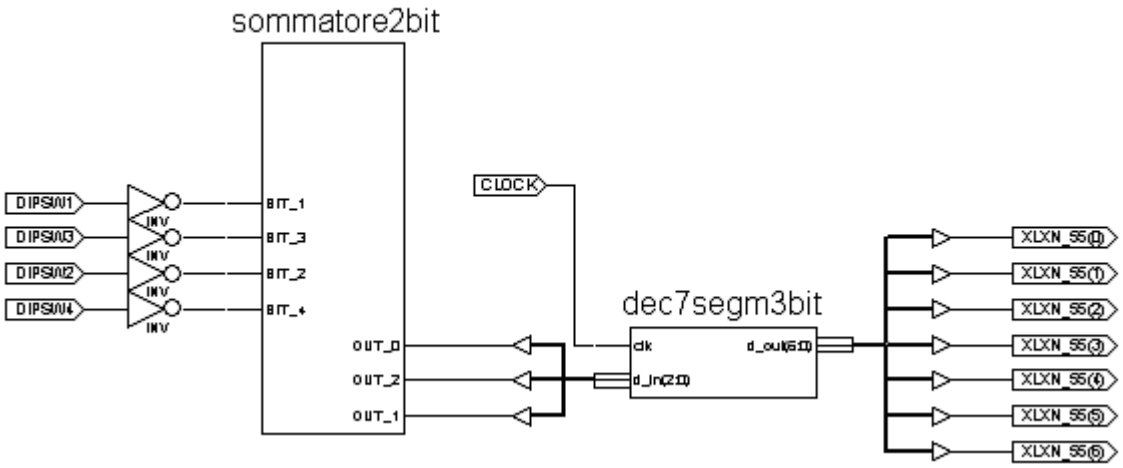


- Anche al decoder associamo un simbolo che adopereremo nello schematico finale: doppio click su 'Create Schematic Symbol' che si trova nel 'Process View', nelle 'Design Entry Utilities'.

A questo punto è necessario creare il disegno finale che sarà uno schematico.

Realizzazione dello schematico finale

- La procedura prevede di aggiungere un nuovo file sorgente al progetto che chiameremo 'som2bit_fin': Project → New Source → Schematic.
  Si realizzi il seguente schema:



Da un punto di vista grafico questo schematico risulta poco chiaro:

- sarebbe meglio dare un nome esplicito tanto ai bus quanto ai singoli collegamenti che ad esso fanno riferimento
- l'uso dei "TAP" può e deve essere fatto intendendo questi ultimi come punti di contatto tra i singoli "wires" ed il bus stesso e senza attaccarli sul punto estremo del bus stesso. Graficamente infatti questa rappresentazione facilmente crea confusione si potrebbe infatti facilmente confondere i TAP con dei BUFFER (che in alcuni casi verrebbero ad avare una direzione errata).

Si noti che in ingresso e in uscita del dec7segm3bit il filo è più grosso. Si tratta di un bus. L'unico problema nella realizzazione dello schema può riguardare il collegamento di un filo a un bus. Tale collegamento avviene unicamente attraverso un bus tap  che si può introdurre nello schematico con un click su  . Bisogna quindi orientare il Bus Tap. A questo scopo alla sinistra della finestra dell'ECS si trovano le 'Bus Tap Options'. Il Bus tap va orientato con l'estremità più larga verso il bus e l'estremità più stretta (filo) verso il filo. È ora necessario assegnare al filo la giusta corrispondenza con il filo del bus. Esaminiamo per esempio il bus che esce dal decoder nello schema sopra: il nome del bus è XLXN_55(6:0), lo si vede facendo doppio click sul bus o semplicemente puntando il cursore sul bus in questione. Per assegnare a ciascun filo alla destra di uno dei Bus Tap il filo corrispondente nel bus bisogna assegnare al filo alla destra del Bus Tap il nome del filo del bus che ad esso si vuole collegare. Ad esempio nel primo filo alla destra del Bus Tap si è assegnato il nome XLXN_55(0) semplicemente facendo doppio click sul filo e cambiando il 'Value' relativo al 'Name' in XLXN_55(0).
Lo stesso vale per i fili e il bus all'ingresso del decoder.

Si salvi lo schema e si ritorni al Project Navigator per verificare tutti i disegni, operare la sintesi e l'implementazione. Prima di creare il file che programmerà la board, può essere opportuno simulare il comportamento del progetto mediante l'utilizzo del ModelSim.

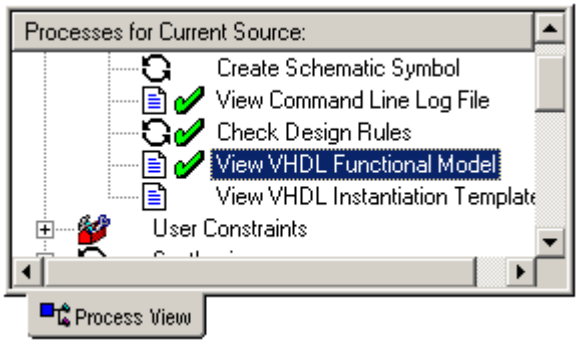
Simulazione con ModelSim

Per la simulazione del dispositivo appena creato sarà necessario utilizzare ModelSim. Come visto nella precedente esercitazione:

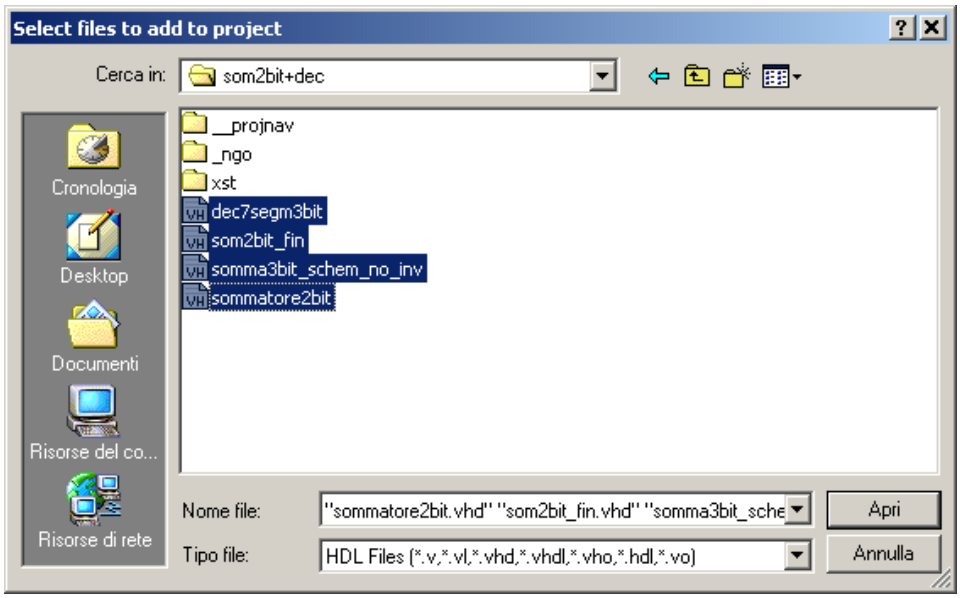
- Avviare ModelSim. Qualora compaia la finestra con le 'New ModelSim Features' premere 'Jumpstart', poi click sull'ipertesto 'Create a Project'. Tutta questa procedura può essere fatta ugualmente dalla finestra ove compare il prompt di ModelSim:

- Si assegni al progetto un nome, ad esempio 'som2bit+dec', e la directory di lavoro desiderata nella 'Project Location'.

- Ora bisogna aggiungere i file che costituiscono il progetto da simulare, nel nostro caso tutti i file sorgente del progetto 'som2bit+dec'.
  Attenzione: si ricordi che ModelSim accetta solo i file in formato HDL pertanto sarà necessario trasformare tutti i file sorgente schematici in file HDL come fatto nell'esercitazione precedente. Ribadiamo ulteriormente la procedura.
  Relativamente a ciascun file sorgente, si osservi la finestra 'Process View' dal Project Navigator; si può notare che nel momento in cui si è fatto il 'Chek Design Rules' per ciascuno degli schematici, è comparso un tick verde anche alla voce 'View VHDL Functional Model':



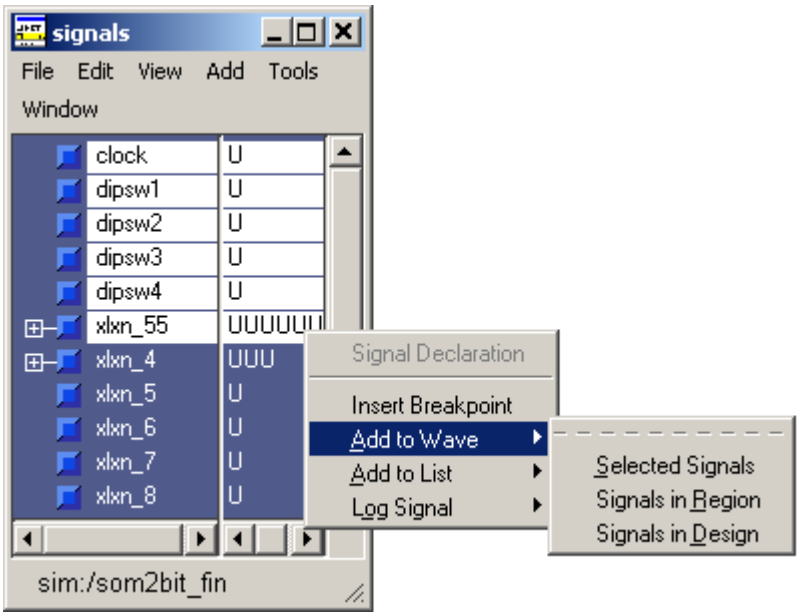
Con un doppio click su 'View VHDL Functional Model' si apre nella finestra a lato il listato del codice VHDL che descrive lo schematico. Il file generato è del tipo *.vhf pertanto l'estensione va modificata in *.vhd semplicemente risalvando il file come 'NomeFile.vhd' (File → Save As → NomeFile.vhd).
Questa procedura va fatta per ciascun file sorgente che non sia in formato VHDL. Nel nostro caso l'unico file pronto per la simulazione è quello relativo al decoder, che è già di per sé scritto in codice VHDL.
A questo punto ritorniamo al ModelSim e con un doppio click su 'Add Existing Files' aggiungiamo tutti i file sorgente al progetto ModelSim:



Attenzione: come intuibile, non è possibile aggiungere e quindi simulare solo il top file del progetto (nel nostro caso 'som2bit_fin') in quanto mancano i riferimenti ai simboli usati (quelli da noi creati, come 'dec7segm3bit'). Si verificheranno degli errori nei passi successivi della simulazione.
Nella finestra successiva 'Add file to Project' esiste la possibilità di copiare i file selezionati nella directory di lavoro del progetto ModelSim selezionando 'Copy to project directory' oppure di prelevare i file dalla loro posizione corrente selezionando 'Reference from current location'.

- Il passo successivo consiste nel compilare i file aggiunti.
  A tale scopo premere il tasto destro del mouse all'interno del menu a tendina 'Project' a sinistra, poi selezionare 'Compile' → 'Compile All'.
  A questo punto i file compilati sono stati copiati nella cartella 'work' del menu a tendina 'Library'. È ora possibile scegliere quale dei file si vuole simulare. Nel nostro caso analizzeremo solo il comportamento del file finale, ma è possibile studiare i segnali per ciascun file con la medesima procedura. Passiamo alla fase di simulazione vera e propria.

- Per avviare la simulazione di un determinato file, è sufficiente fare doppio click sul file desiderato nella cartella 'work' del menu 'Library'. Nel nostro caso 'som2bit_fin'. Vengono quindi caricati tutti i dati relativi al file in esame e si apre il menu 'sim'.
  Per avere un quadro completo della situazione è ora consigliabile aprire tutte le finestre relative alla simulazione, anche se poi ne utilizzeremo solo alcune: dalla finestra principale di ModelSim 'View'→'All Windows'.
  Ci serviranno, oltre a quella principale dalla quale scriveremo i comandi per generare i segnali, saranno le finestre 'signals' e 'wave', pertanto per ora chiuderemo le altre.
  Si tratta ora di scegliere i segnali che si vogliono analizzare, cioè l'insieme dei segnali che si vogliono controllare e osservare. Ad esempio si selezionino (ctrl + tasto sinistro del mouse): 'clock', 'dipsw1', 'dipsw2', 'dipsw3', 'dipsw4' e tutti gli 'xlxn_55' che controllano l'accensione dei led del display a 7 segmenti. Poi col tasto destro del mouse di selezioni 'Add to Wave'→ 'Selected Signals':



  I segnali selezionati verranno copiati nella finestra 'wave'. Dalla finestra wave è possibile comandare la simulazione mediante i tasti posti in alto.
  Prima di far partire la simulazione è necessario generare gli stimoli i segnali.

- Per generare gli stimoli la cosa più semplice è, come visto nella precedente esercitazione, ritornare al prompt dei comandi.
  Il comando è sempre 'force'. Ricordiamo che, digitando 'help force' dal prompt, compariranno le istruzioni relative alla sintassi:

  force [-freeze | -drive | -deposit] [-cancel <time>] [-repeat <time>] <item_name> <value> [<time>] [, <value> <time> ...]

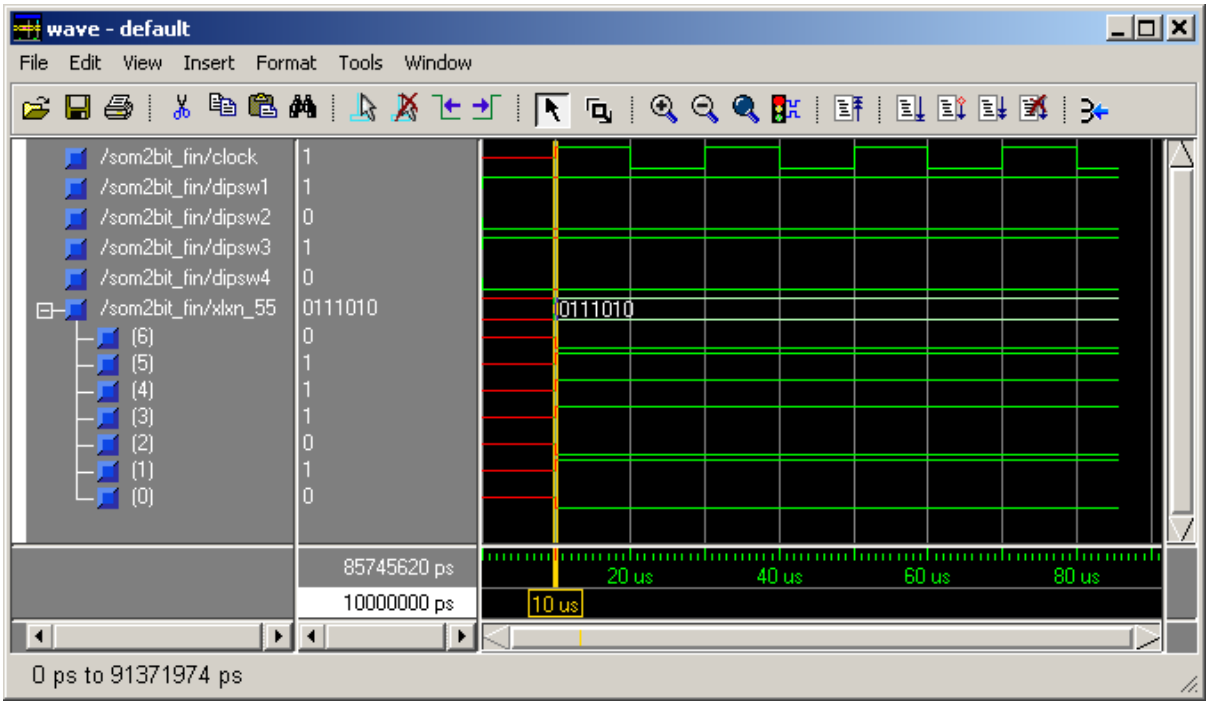
  Per informazioni sul comando 'force' si veda l'esercitazione precedente o si consulti lo 'User's Manual' ('Help' → 'PDF Documentation' → 'User's Manual').

  Poniamo di voler simulare la somma 10+10. Mettiamo un clock di periodo 20µs, che, grossomodo, corrisponde alla frequenza di 48.7 kHz precedentemente impostata. Attenzione: nel disegno si sono posti gli invertitori subito dopo i dipsw quindi bisogna creare gli stimoli al contrario; pertanto poniamo dipsw1 e dipsw3 al valore 1, gli altri a 0. Al prompt digitiamo:

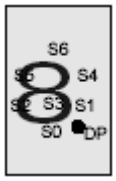
  force clock 1 10us, 0 20us -repeat 20us
  force dipsw1 1 10 -repeat 10
  force dipsw2 0 10 -repeat 10

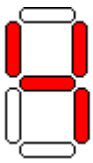
force dipsw3 1 10 -repeat 10
force dipsw4 0 10 -repeat 10

- A questo punto torniamo alla finestra ‘wave’ e premiamo per esempio il pulsante  che corrisponde al comando ‘run -all’. La simulazione parte ed è possibile controllarne l’avanzamento seguendo il contatore ‘Now: …’. Per bloccare la simulazione è sufficiente premere il tasto  . A questo punto si può osservare il risultato nella finestra ‘wave’. Ad esempio è possibile vedere il tempo di reazione del circuito dal momento in cui vengono applicati gli stimoli:



Com’è visibile il circuito, in questo caso, reagisce dopo 10µs dall’inizio della simulazione. Ponendo a 1 i segnali che accendono i led s1, s3, s2, s5. Controlliamo se il risultato di 10 + 10 = 4 è visualizzato correttamente.



che corrisponde a  .
La simulazione ha confermato quanto ci aspettavamo.