Tutorial Modelsim

Simulazioni di circuiti logici utilizzando Modelsim

Descrizione: Si utilizzi il tool Modelsim per realizzare simulazioni di un circuito sequenziale asincrono affetto da Alea

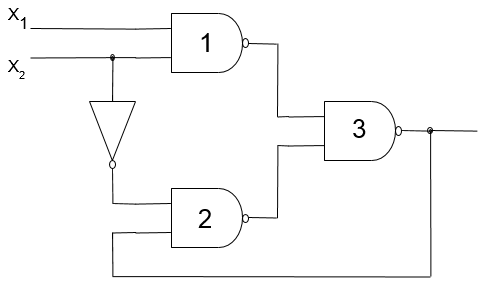
Scopo: familiarizzare col tool di sviluppo Modelsim ed apprendere i meccanismi che possono generare alee.

Apprendimento previsto:

- Descrizione di un semplice circuito logico utilizzando Verilog HDL
- Simulazione del circuito in maniera interattiva (step by step)
- Realizzazione di un file di stimoli e descrizione del medesimo attraverso il linguaggio HDL
- Utilizzo del file di stimoli per una simulazione batch

Introduzione al problema

Si voglia simulare il circuito qui di seguito riportato



.

Questo è un circuito sequenziale asincrono che può risultare affetto da un'alea statica [6]. Questo comporta un'interessante criticità: in pratica il funzionamento è fortemente legato dai ritardi di propagazione dei segnali attraverso i vari percorsi e si potrà notare che alterando questi, si viene a modificare il funzionamento del circuito stesso.

Il funzionamento del circuito è descritto da questa tavola di flusso:

st\x1x2	00	01	11	10
0	**0**	**0**	1	**0**
1	**1**	0	**1**	**1**

Si supponga che all'accensione entrambi i segnali d'ingresso x1 e x2 siano a 0. L'uscita può assumere in tal caso indifferentemente sia il valore 0 che il valore 1 ed autosostenersi attraverso il loop di reazione (in altre parole al momento di accensione del circuito non c'è modo di stabilire se l'uscita sia a 0 oppure a 1). Se però si alza il segnale x2 l'uscita assume il valore 0 e questo viene mantenuto anche se successivamente x2 si abbassa riportandosi alle condizioni iniziali.

st\x₁x₂	00	01	11	10
0	**0**	**0**	1	**0**
1	**1**	0	**1**	**1**

Viceversa se dopo l'accensione con entrambi gli ingressi a 0 si va ad alzare il segnale x1 l'uscita assume il valore 1 e lo mantiene anche quando x1 dovesse tornare ad abbassarsi.

st\x₁x₂	00	01	11	10
0	**0**	**0**	1	**0**
1	**1**	0	**1**	**1**

Se entrambi i segnali sono a livello alto l'uscita assume il valore 1, valore che dovrebbe essere mantenuto anche quando x2 si abbassa. Si può però notare che a seconda dei tempi di propagazione potrebbe anche avvenire la transizione sbagliata e l'uscita potrebbe portarsi allo stato basso. In pratica se $\Delta i + \Delta 2 - \Delta 1 > \Delta 3$ l'uscita passerà al valore 0, altrimenti la transizione sarà corretta e l'uscita rimarrà allo stato alto.

st\x₁x₂	00	01	11	10
0	**0**	**0**	1	**0**
1	**1**	0	**1**	**1**

st\x₁x₂	00	01	11	10
0	**0**	**0**	1	**0**
1	**1**	0	**1**	**1**

E' pertanto fondamentale poter simulare il circuito analizzando nel dettaglio i tempi di propagazione attraverso i vari elementi del circuito.

Procedimento

Attraverso un comune text editor si realizzi un file scritto in Verilog HDL che descriva (a livello comportamentale) il circuito sopra riportato con l'introduzione (manuale) di eventuali tempi di ritardo nei vari elementi e lo si salvi in un opportuno direttorio.

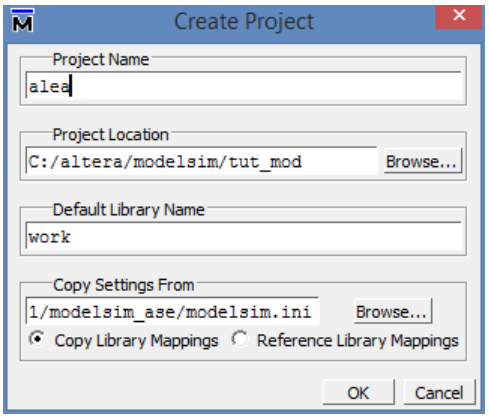
Il file in questione potrebbe essere il seguente:

```
`timescale 1ps / 1ps

module alea (x1,x2,y);
input x1,x2;
output y;

wire i1,i2,i3;

  assign #20 i1=~(x1 &  x2);
  assign #20 i2=~(i3 & y);
  assign #30 i3=~x2;

  assign #20 y=~(i1 & i2);
endmodule
```

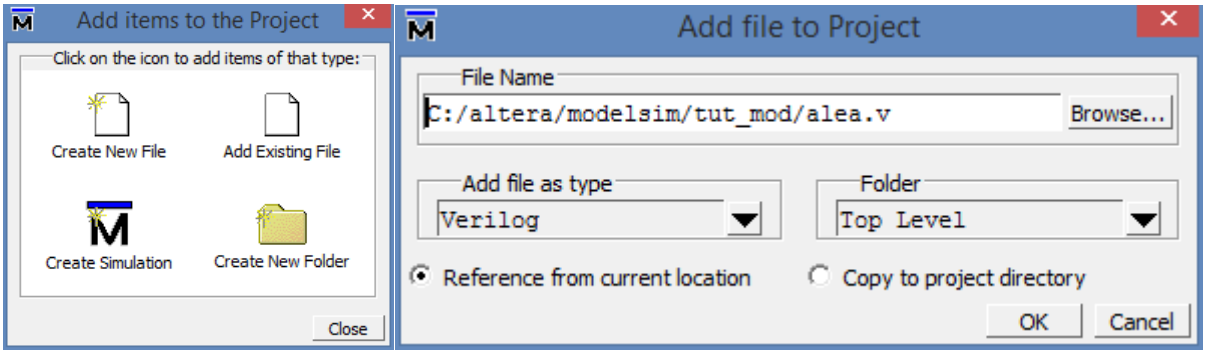
1. *Compilazione del sorgente*

Si apra il tool Modelsim o tramite icona sul desktop o dall'elenco dei programmi istallati

Si crei un nuovo progetto specificandone nome e ubicazione del direttorio nel quale farlo risiedere. Si mantenga pure il nome di default "work" della libreria alla quale appoggiarsi per farvi risiedere i files compilati. Si copi il file di configurazione di default o dal direttorio di installazione o dall'ultimo progetto svolto.



Si aggiunga al progetto il file sviluppato "add existing file"



> OK
> Close

Si compili il file in tal modo esso entra a far parte di una delle varie librerie disponibili. Per default è bene utilizzare allo scopo la libreria “work” (che conterrà tutti i moduli del progetto sul quale si sta lavorando).

Compile > Compile All

Eventuali errori di sintassi verranno evidenziati in questa fase. Se non vi sono errori nella libreria “Work” apparirà il modulo “alea”.

Eventualmente per far comparire la scheda relativa verificare che la scheda relativa alla gestione delle librerie sia attiva

View > Library (alt-vu)

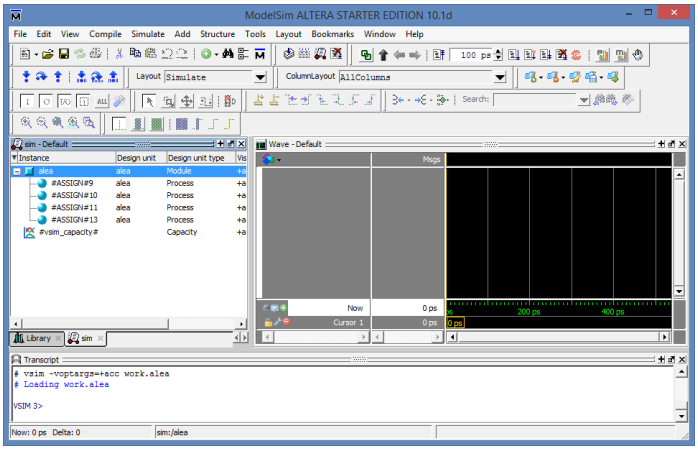
2. *Simulazione step by step*

All’interno della finestra Library cliccare col tasto destro sopra il modulo alea (all’interno della libreria work) e selezionare **Simulate.**

Compare una nuova scheda orientata alla gestione dei segnali in fase di simulazione.

Accertarsi che sia visibile la schermata per la visualizzazione grafica dei segnali (Wave)

View > Wave (alt-vv)

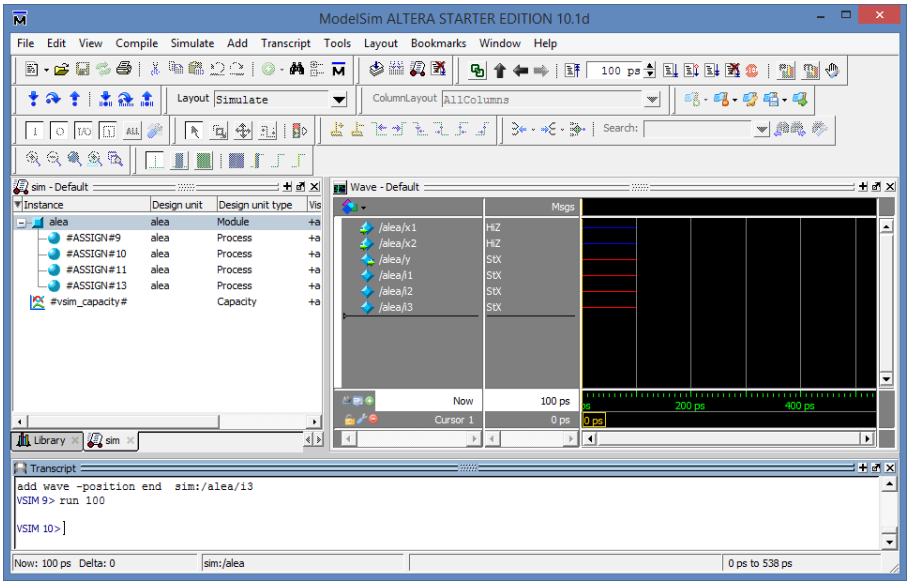


Trascinare l'istanza denominata "alea" dalla finestra sim alla finestra wave. Si noti che appaiono tutti i segnali inerenti tale blocco, compresi i segnali interni.

Nella finestra di Console digitare

```
> run 100
```

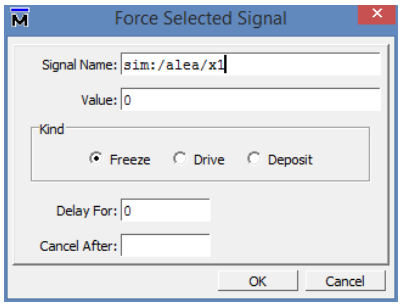
Il sistema esegue i primi 100 passi di simulazione (equivalenti nel nostro caso a 100 ps)



Si noti che inizialmente i segnali di ingresso non sono definiti (highZ) e pertanto i segnali d'uscita nonché i segnali interni risultano inteterminati (StX).

Si forzino ora i segnali di ingresso al livello logico 0.

- Right click sul segnale `/alea/x1` nella finestra Wave
- Force …
- Nella finestra che appare si scriva all'interno del campo Value il valore 0
- OK

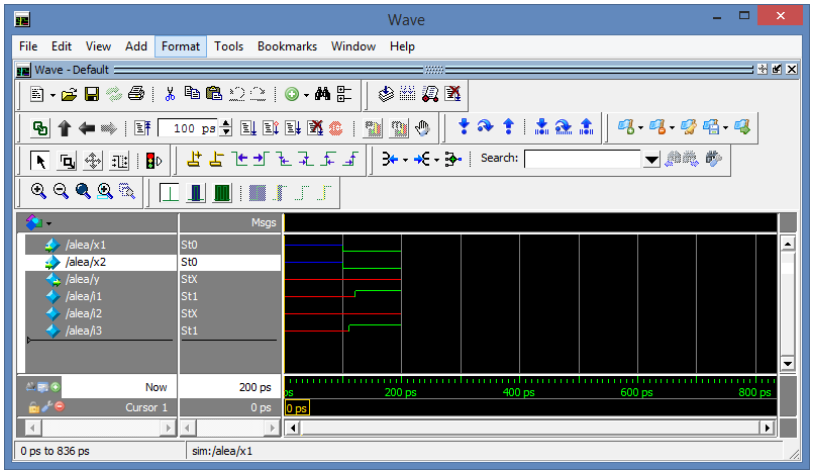


Si replichi la procedura anche per il segnale x2
All'interno della finestra di Console si digiti

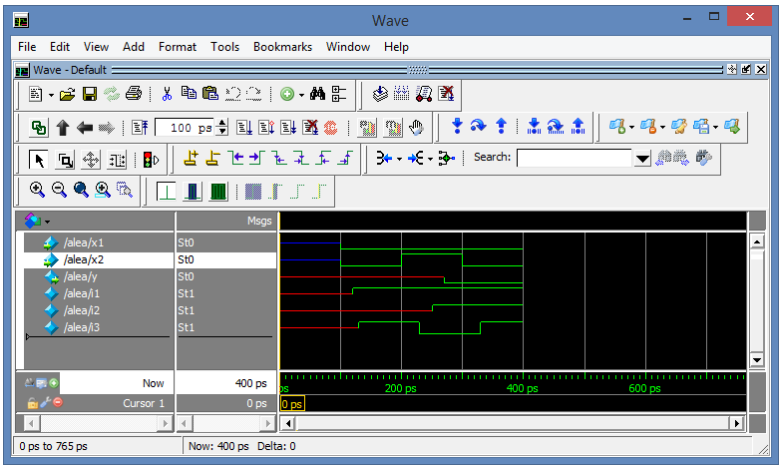
```
> run 100
```

Ora diversi segnali assumono il corretto valore logico dopo un certo intervallo temporale compatibile con i tempi di ritardo introdotti in fase di descrizione del circuito, ma altri (y ed i2) permangono allo stato "StX" (indefinito). Questo è dovuto alla natura stessa del circuito, infatti entrambi i valori logici (0 e 1) sarebbero

compatibili col funzionamento e non vi è modo come detto nell'introduzione, di discriminare in quale dei due stati si trovi il sistema.



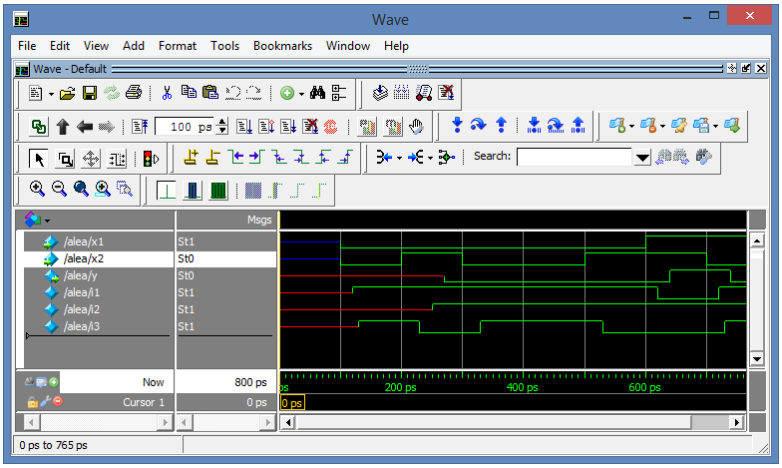
Ripetendo la procedura illustrata sopra si provi ad alzare e ad abbassare il segnale x2, facendo seguire ad ogni modifica un periodo di funzionamento di 100ps.



Si noti che con questa procedura il sistema ha portato l'uscita in uno stato ben definito (lo stato 0) e che in tale stato esso permane anche quando i segnali in ingresso sono ritornati entrambi allo stato 0.

Si provi ora ad assegnare seguendo la procedura sopra suggerita la sequenza di stimoli (relativa ai segnali x1 ed x2) 00 – 01 – 11 – 10, facendo seguire a ogni stimolo un sufficiente periodo di simulazione (ad es. 100ps) tale da garantire che tutti i ritardi si siano estinti.

st\x1x2	00	01	11	10
0	**0**	**0**	1	**0**
1	**1**	0	**1**	**1**



Si può notare che correttamente l'uscita si porta al valore alto in corrispondenza al valore 11 per gli ingressi, ma al passaggio a 10 invece di mantenersi al valore alto come previsto nella tavola di flusso, per la presenza di un'alea si riporta al valore 0.

st\x1x2	00	01	11	10
0	**0**	**0**	1	**0**
1	**1**	0	**1**	**1**

Si può infatti notare che se il ritardo dell'invertitore viene ridotto dagli attuali 30ps a 10ps l'alea non si presenta ed il funzionamento sarà diverso. Per effettuare questa simulazione adotteremo un metodo diverso è più intuitivo.

3. *Simulazione in batch attraverso definizione delle f.d.o.*

- All'interno della finestra Library cliccare col tasto destro sull'istanza "alea" e scegliere Edit.
- Nella finestra di testo che compare modificare il ritardo dell'invertitore da #30 a #10

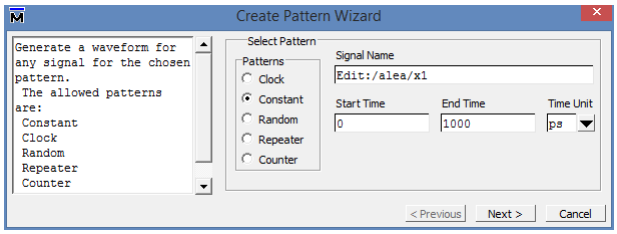
```
. . .
assign #20 i1=~(x1 &  x2);
assign #20 i2=~(i3 & y);
assign #10 i3=~x2;

assign #20 y=~(i1 & i2);
. . .
```

- Salvare
- Sempre attraverso il click destro scegliere "`Recompile`"
- e successivamente "`Simulate`"
- Sempre nella finestra Library (presente eventualmente sotto alla finestra "sim") con un click destro sull'istanza "alea" scegliere "`Create Wave`".

Nella finestra Wave appaiono ora 3 segnali (x1,x2 e y)

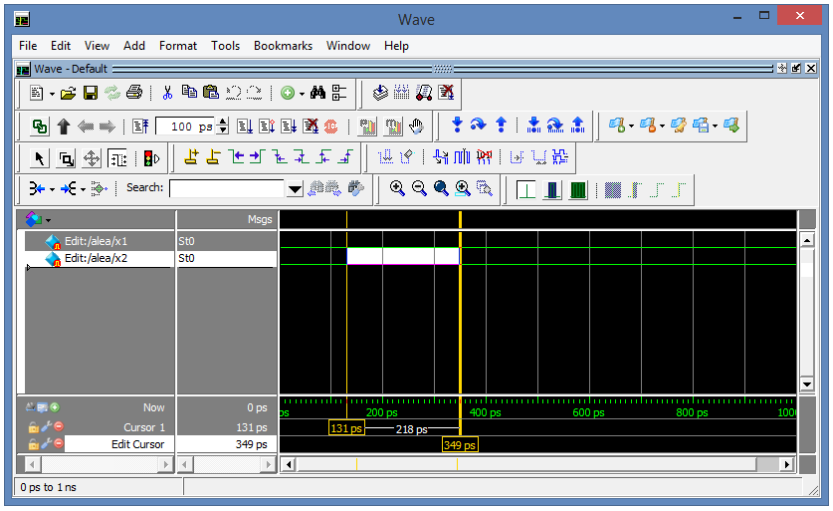
- Cancellare il segnale y (`right clik > Edit > Delete`)
- Editare i segnali x1 e x2 perché assumano tra 0ps e 1000ps il valore costante 0:
    - `Right click > Edit > Create/Modify Waveform`
    - Nella finestra che appare scegliere
        - Costant,

        - Start Time=0
        - End Time=1000
        - Time Unit ps
        - Next



    - Nella successive finestra
        - fissare il valore a 0
        - Finish

- Ripetere la procedura anche per il segnale x2
- Modificare la modalità di funzionamento del mouse
    - `Wave > Mouse Mode > Edit Mouse`
- Utilizzare il mouse per evidenziare una porzione del segnale x2



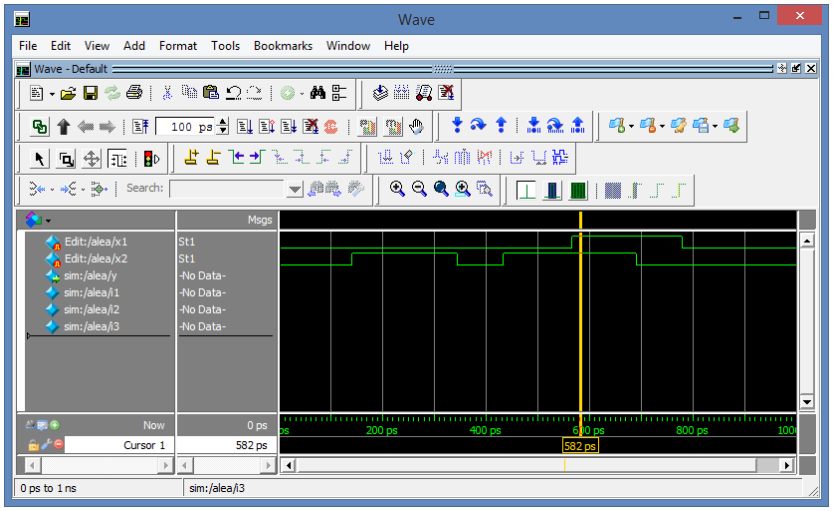
- Modificare il valore di questa porzione

    `Wave > Wave Editor > Invert`

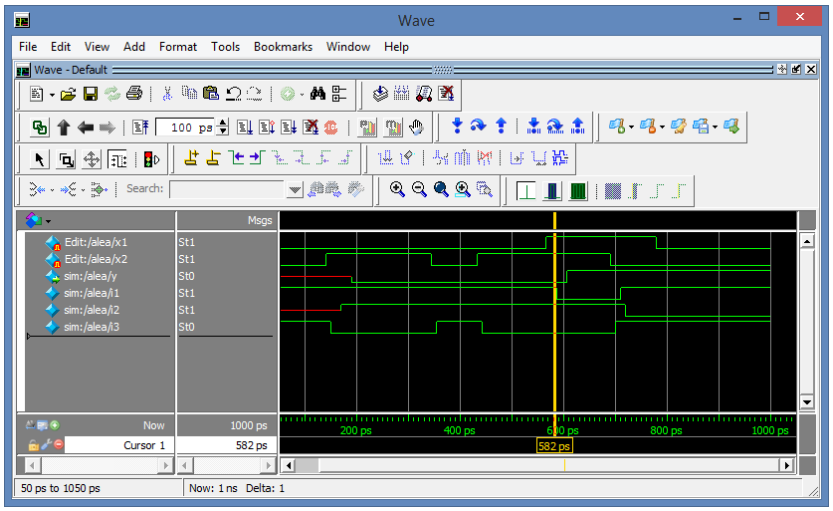
Eventualmente si può sfruttare la "`Edit Toolbar`" per avere a disposizione pulsanti atti editare più rapidamente i segnali

`Window > Toolbars > Wave Edit`

- Utilizzare questa tecnica per definire completamente le f.d.o con cui stimolare il circuito
- Trascinare dalla finestra "Sim" alla finestra "Wave" tutti i segnali che si intendono monitorare



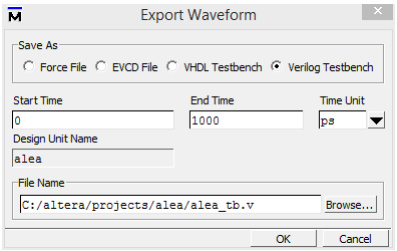
- Nella finestra console digitare
    - `Run -all`



Si può notare che con un diverso ritardo dell'invertitore il fenomeno dell'alea non si verifica più ed il circuito funziona in modo corretto.

Se si vogliono simulare più soluzioni circuitali attraverso i medesimi stimoli è possibile salvare sia gli stimoli che l'unità da testare all'interno di un file Verilog HDL.

- Assicurarsi che la finestra "Wave" risulti evidenziata (eventualmente cliccandoci sopra)
- `File > Export > Waveform`
    - Scegliere il formato Verilog Testbench
    - La durata della simulazione
    - Il nome del file ed il direttorio in cui salvarlo



    - OK

Si consiglia di aprire il file appena generato con un text editor e di analizzarne la sintassi che ne descrive il comportamento e di come gli stimoli vengano forniti alla UUT (Unit Under Test). Questo in esame è un file che impiega il linguaggio Verilog HDL per realizzare una descrizione "comportamentale" del sistema completo (UUT + generatore di stimoli).

4. Simulazione di un testbench completo descritto in Verilog HDL

Il file generato al passo precedente contiene sia la definizione degli stimoli, sia come questi vengono portati alla unità da testare (UUT). Pertanto esso, insieme alla descrizione Verilog HDL dell'unità da testare possono essere impiegati per rigenerare la simulazione o eventualmente possono essere modificati per generare diversi test.

Si apra Modelsim o se quest'ultimo è già aperto si ultimi la simulazione in corso:

- Simulate > End Simulation

Si importi nel corrente progetto il file in oggetto: facendo riferimento alla finestra Project (eventualmente aprendola se essa dovesse essere chiusa)

```
View > Project (Alt - vx)

Project > Add To Project > Existing File
```

Al momento il file (nella sua versione testuale) fa parte del progetto, ma per poterlo utilizzare esso deve venir compilato. Si evidenzi pertanto il file cliccandovi sopra

```
Compile > Compile Selected (Alt – ce)
```

Ora all'interno della libreria work vi sono due moduli: il file che realizza il testbench ed il file che descrive il circuito sotto test.

- Si clicchi col tato destro sul modulo relativo al test bench e si scelga `Simulate`
- Nella Finestra "Wave" si trascinino i segnali che si vogliono analizzare (ad esempio si può trascinare l'intera istanza DUT contenuta all'interno del testbench per visualizzare tutti i segnali che ad essa fanno riferimento)
- Nella finestra di Console di digiti
    - `Run –all`

Ora si può eventualmente modificare il sorgente Verilog HDL che descrive il circuito da analizzare modificandone ritardi e/o collegamenti e dopo una ricompilazione si può rilanciare la simulazione con i medesimi stimoli e verificare i risultati.

Ad esempio si potrebbe modificare il circuito introducendo un percorso ridondante per annullare gli effetti dell'alea:

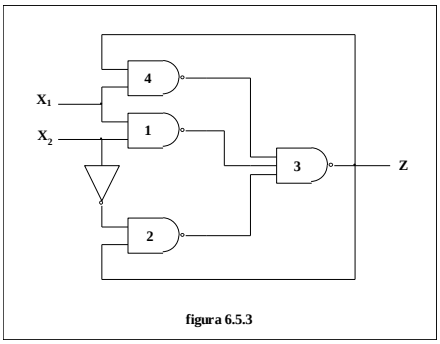


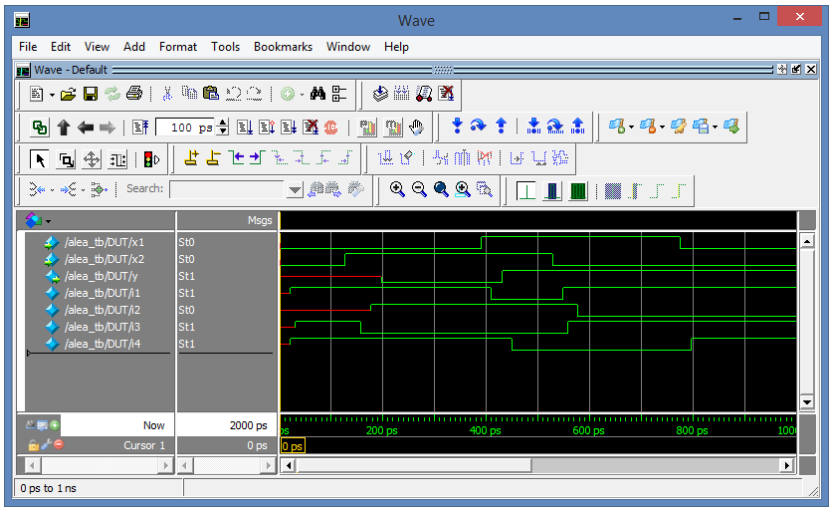
figura 6.5.3

Tale circuito potrebbe venir descritto in Verilog HDL col seguente sorgente:

```
`timescale 1ps / 1ps

module alea (x1,x2,y);
input x1,x2;
output y;

wire i1,i2,i3,i4;

assign #20 i1=~(x1 &  x2);
assign #20 i2=~(i3 & y);
assign #30 i3=~x2;
assign #20 i4=~(x1 & y);


assign #20 y=~(i1 & i2 & i4);
endmodule
```

Si può pertanto

- Fermare la simulazione attuale
- Rieditare il modulo “alea.v” originale
- Ricompilare il modulo alea così modificato
- Rilanciare la simulazione
- Definire i segnali da visualizzare
- Far procedere la simulazione per un certo tempo.



I risultati dimostrano che nonostante il tempo di ritardo dell’invertitore possa essere critico, l’introduzione del nuovo percorso garantisce l’assenza di alee ed un corretto funzionamento della reazione.

Bibliografia

[1] Modelsim Home page
http://www.mentor.com/products/fv/modelsim/

[2] Mentor-Graphics Home page
http://www.mentor.com/

[3] ENEE 359a: Digital VLSI Circuits by B. Jacob – "**Project 1: ModelSim Tutorial and Verilog Basics**"
http://www.ece.umd.edu/class/enee359a.S2008/p1.pdf

[4] Altera Corporation: "**Introduction to Simulation of Verilog Designs Using ModelSim Graphical Waveform Editor**"
ftp://ftp.altera.com/up/pub/Altera_Material/9.1/Tutorials/Verilog/ModelSim_GUI_Introduction.pdf

[5] Altera Corporation: "**Using ModelSim to Simulate Logic Circuits in Verilog Designs"**
ftp://ftp.altera.com/up/pub/Altera_Material/13.0/Tutorials/Verilog/Using_ModelSim.pdf

[6] Antonio D'Amore : "**Dispense del corso di Reti Logiche**"
http://www2.units.it/marsi/reti/dispense/capitolo%2006.pdf

[7] Altera Corporation: "**Introduction to Simulation of Verilog Designs Using ModelSim Graphical Waveform Editor**"
ftp://ftp.altera.com/up/pub/Altera_Material/13.0/Tutorials/Verilog/ModelSim_GUI_Introduction.pdf

[8] Mentor Graphics: **"ModelSim Reference Manual - Software Version 10.1a**"
http://moodle.units.it/moodle/file.php/670/Manuals/Modelsim/ModelSim_Reference_Manual_v10.1c.pdf

[9] Mentor Graphics: **"ModelSim User Manual - Software Version 10.1a**"
http://moodle.units.it/moodle/file.php/670/Manuals/Modelsim/ModelSim_Users_Manual_v10.1c.pdf

[10] Mentor Graphics "**ModelSim Tutorial -Software Version 10.1a**"
http://www.fatih.edu.tr/~onur/download/EEE546/modelsimTutorial.PDF

[11] Larbi Boughaleb "**ModelSim Tutorial**"
http://www.ece.northwestern.edu/~ismail/courses/c92/ModelSim_Tutorial.pdf

[12] "**ModelSim Tutorial**"
http://ee.hawaii.edu/~sasaki/EE361/Fall06/Lab/Lab4.1/ModelSim.pdf