Tutorial Modelsim

Simulazioni di circuiti logici utilizzando Modelsim

Descrizione: Si utilizzi il tool Modelsim per realizzare diverse simulazioni di un circuito sequenziale asincrono.

Scopo: familiarizzare col tool di sviluppo Modelsim e con le macchine sequenziali asincrone.

Apprendimento previsto:

- Descrizione di un semplice circuito logico utilizzando un linguaggio di descrizione Hardware
- Simulazione del circuito in maniera interattiva (step by step)
- Realizzazione di un file di stimoli
- Utilizzo del file di stimoli per una simulazione batch

Introduzione a ModelSim

Modelsim [1] è un tool di simulazione sviluppato da Mentor-Graphics [2] che è stato adottato come "state of the art" all'interno di diversi software EDA (Electronic Design Automation) dedicati al progetto di circuiti elettronici. In questo tutorial analizzeremo le basi per una simulazione di un circuito logico descritto in VerilogHDL utilizzando ModelSim.

Introduzione al problema

Si voglia realizzare una macchina sequenziale asincrona in grado di rilevare lo stato di libero/occupato di un Bus I2C[3]. All'atto pratico si vuole realizzare una macchina dotata di due ingressi (SDA ed SCL) ed un'uscita. Il sistema deve essere in grado in grado di alzare l'uscita quando in ingresso si presenta la sequenza 11-10 e viceversa di riportarla allo stato basso quando all'ingresso si presenta la sequenza 10 -11.

Si può facilmente verificare che una macchina che può realizzare tale sistema può essere descritta dalla seguente tavola di Huffman:

stati\ingressi	00	01	11	10
A	B/1	A/1	A/0	-/-
B	B/1	A/1	B/1	B/1

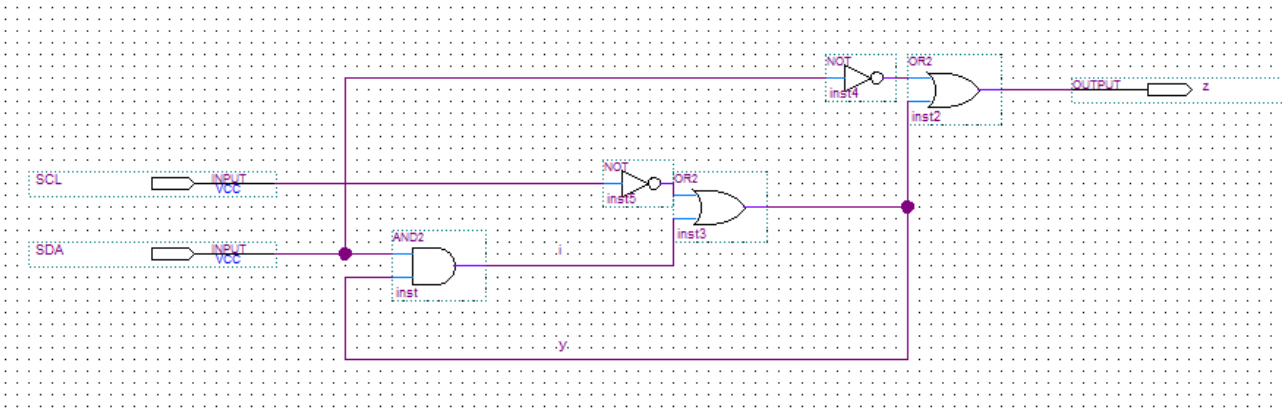
Se a questo punto si adotta la seguente codifica attraverso una sola variabile di stato:

- stato A : y=0
- Stato B: y=1

La nostra macchina può essere facilmente descritta dalle seguenti equazioni

$$y = \overline{SCL} + y'.SDA$$
$$z = y + \overline{SDA}$$

Che possono essere rappresentate schematicamente nel seguente circuito:



Il corretto funzionamento del circuito risiede essenzialmente nel tempo di ritardo intrinseco per l'aggiornamento della variabile di reazione 'Y', che di fatto, grazie ad esso, funge da 'memoria' dello stato passato. Risulta pertanto fondamentale poter disporre di un metodo in grado di descrivere non solo il comportamento del circuito, ma anche i vari ritardi legati alla propagazione del segnale attraverso le porte logiche. Per fare questo si può ricorrere ad un linguaggio di descrizione Hardware (HDL) quale ad esempio il Verilog HDL [4-5]. Peraltro oltre alla pura descrizione del circuito vi sarà la necessità, per poter comprendere appieno il funzionamento del circuito di disporre di un sistema di simulazione in grado di fornire dei segnali di ingresso al nostro sistema e di evidenziare l'evoluzione temporale di tutti i segnali del circuito in ogni istante di tempo.

Procedimento

Attraverso un comune text editor si vada a descrivere un possibile funzionamento del circuito attraverso il linguaggio Verilog HDL. Una possibile descrizione dello stesso potrebbe essere la seguente:

```
`timescale 1ps / 1ps

module i2c_busy(sda,scl,z);
input sda,scl;
output z;

wire y,i;

assign #20 i=( sda & y);
assign #20 y=(~scl | i);
assign #20 z=(~sda | y);

endmodule
```

- la specifica ‘timescale definisce l’unità di misura temporale.
- Module descrive il sistema (modulo) da un punto di vista esterno: gli fornisce un nome (i2c_busy) e dà un nome ai segnali di input/output. Nelle due righe seguenti vengono definiti come segnali d’ingresso rispettivamente ‘sda’ ed ‘scl’, mentre come segnale d’uscita il segnale ‘z’
- Si definisce successivamente l’esistenza di due segnali interni al circuito stesso che sono rispettivamente ‘y’ (variabile di stato) ed ‘i’ (segnale intermedio di collegamento tra le diverse porte logiche. Volendo si sarebbero potuti descrivere anche i segnali interni che collegavano gli invertitori alle rispettive porte logiche, ma per semplicità si è preferito ipotizzare che l’invertitore sia integrato nella porta OR che segue
- Successivamente si trovano le equazioni che descrivono il funzionamento del circuito ipotizzando che ciascuna di esse preveda un ritardo pari a 20 unità di tempo (che per la timescale definita all’inizio equivarranno a 20ps. Si noti che
    - Il simbolo ~ : rappresenta l’operazione NOT
    - Il simbolo | : rappresenta l’operazione OR
    - Il simbolo & : rappresenta l’operazione AND
- Endmodule conclude la descrizione.

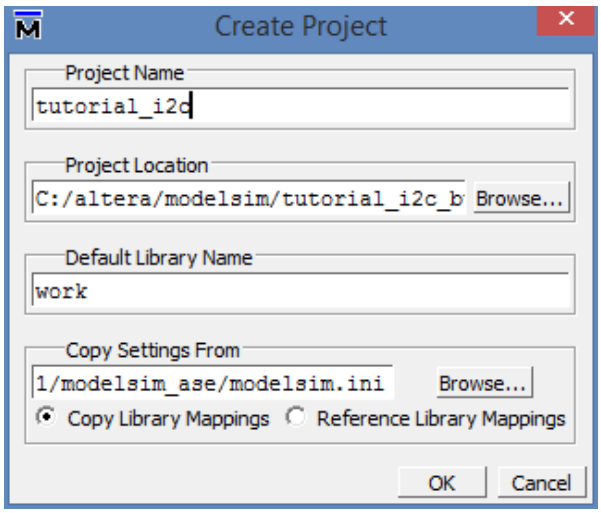
1. *Compilazione del sorgente*

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

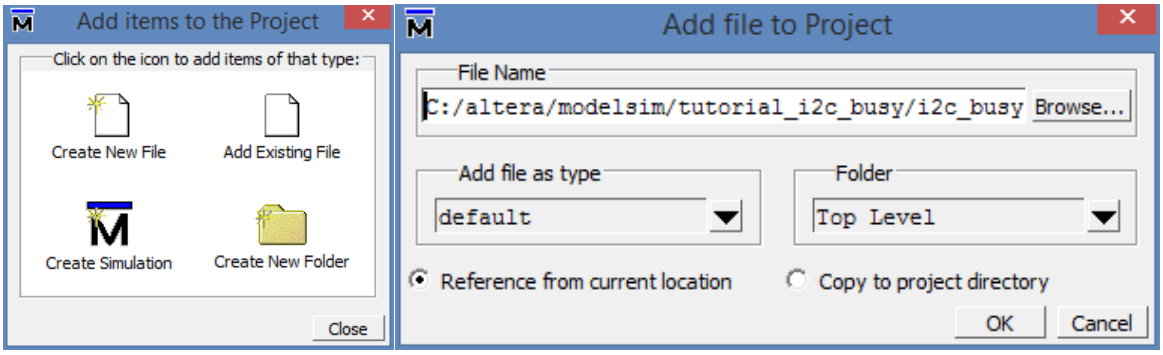
Si crei un nuovo progetto

```
File > New > Project
```

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

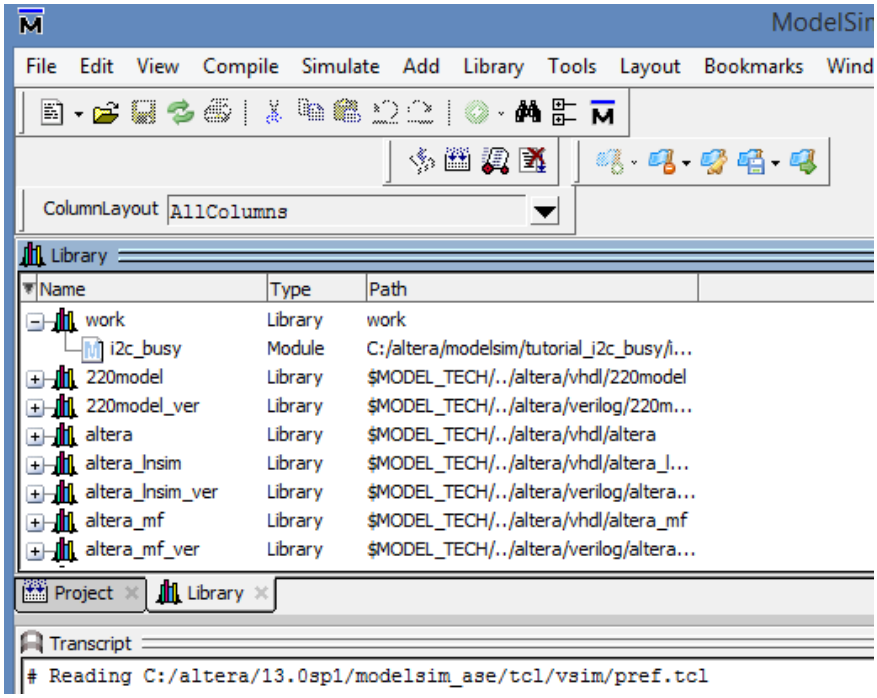
Si compili il file, in tal modo esso entra a far parte di una delle varie librerie disponibili. Per default è bene utilizzare allo scopo la libreria “work” (che conterrà eventualmente tutti i moduli del progetto sul quale si sta lavorando).

Compile > Compile All

Eventuali errori di sintassi verranno evidenziati in questa fase. Se non vi sono errori nella libreria “Work” apparirà il modulo “`i2c_busy`”.

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

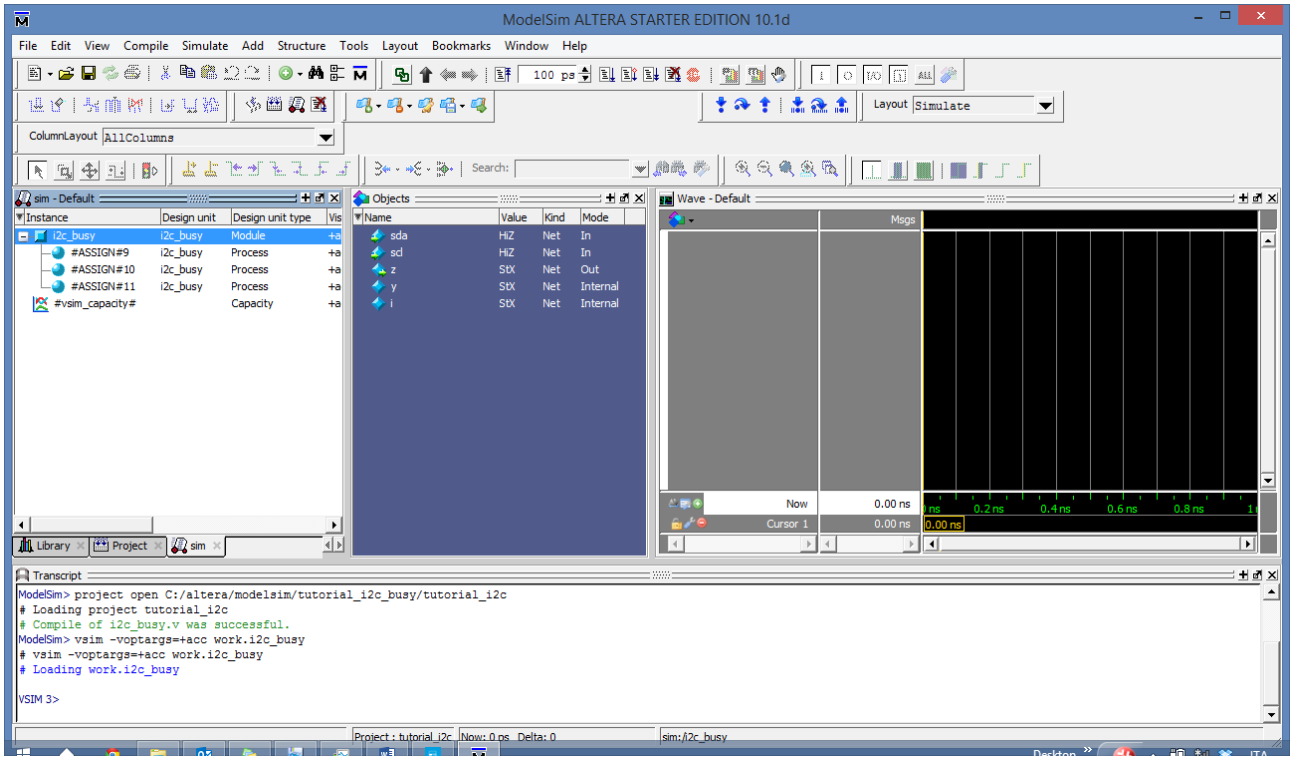
`View > Library (alt-vu)`



2. Simulazione step by step

All'interno della finestra Library cliccare col tasto destro sopra il modulo i2c_busy (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) e quella per la visualizzazione degli "Objects"

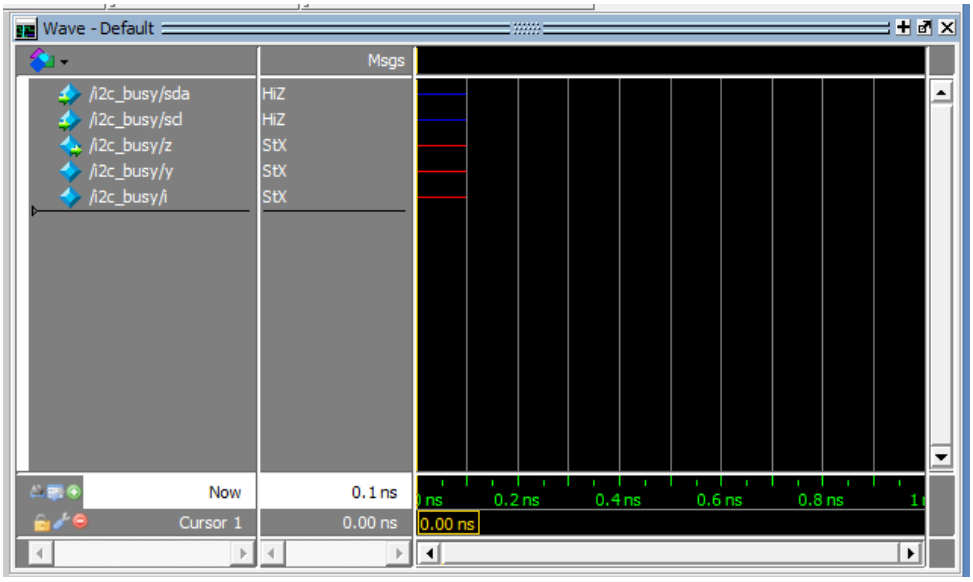
```
View > Wave  (alt-vv)

View > Objects
```

Trascinare l'istanza denominata "busy_i2c" dalla finestra sim alla finestra wave. Si noti che appaiono tutti i segnali inerenti tale blocco, compresi i segnali interni. Oppure eventualmente si possono trascinare solo i segnali di interesse dalla finestra "Objects" alla finestra "Wave"

Nella finestra di Console digitare

```
> run 100
```

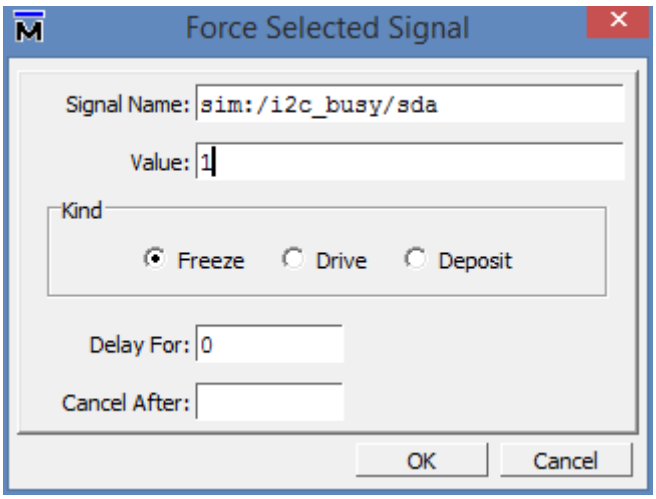
Il sistema esegue i primi 100 passi di simulazione (equivalenti nel nostro caso a 100 ps)



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

Si forzino ora i entrambi i segnali di ingresso al livello logico 1.

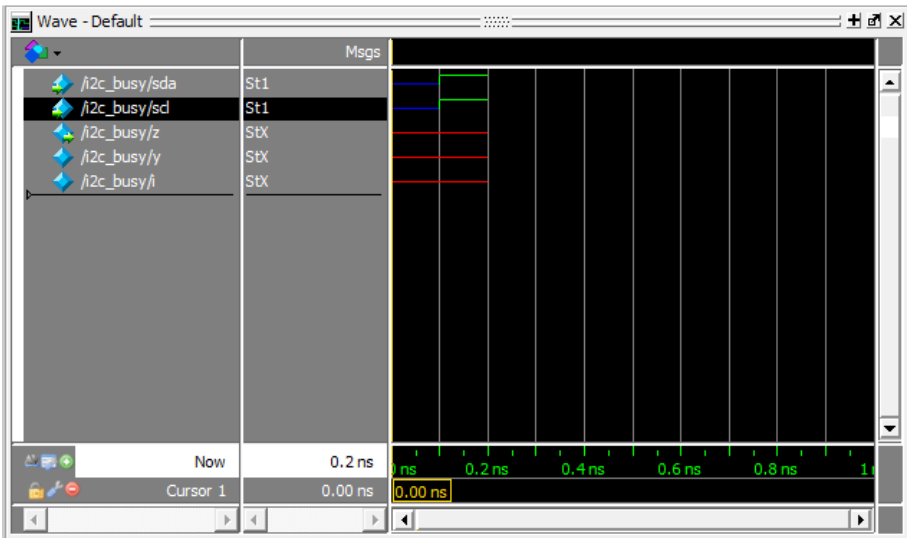
- Right click sul segnale `/i2c_busy/sda` nella finestra Wave
- Force …
- Nella finestra che appare si scriva all'interno del campo Value il valore 1
- OK



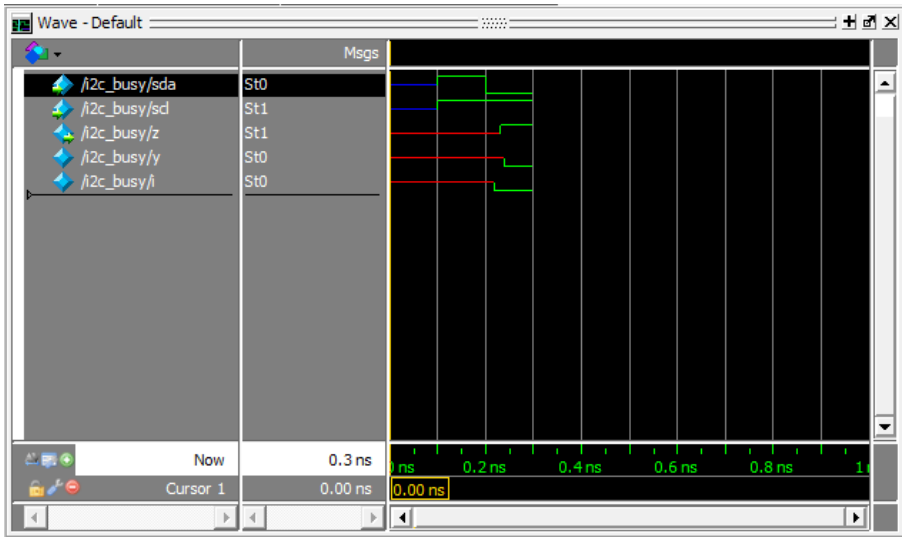
Si replichi la procedura anche per il segnale SCL
All'interno della finestra di Console si digiti di nuovo

```
> run 100
```

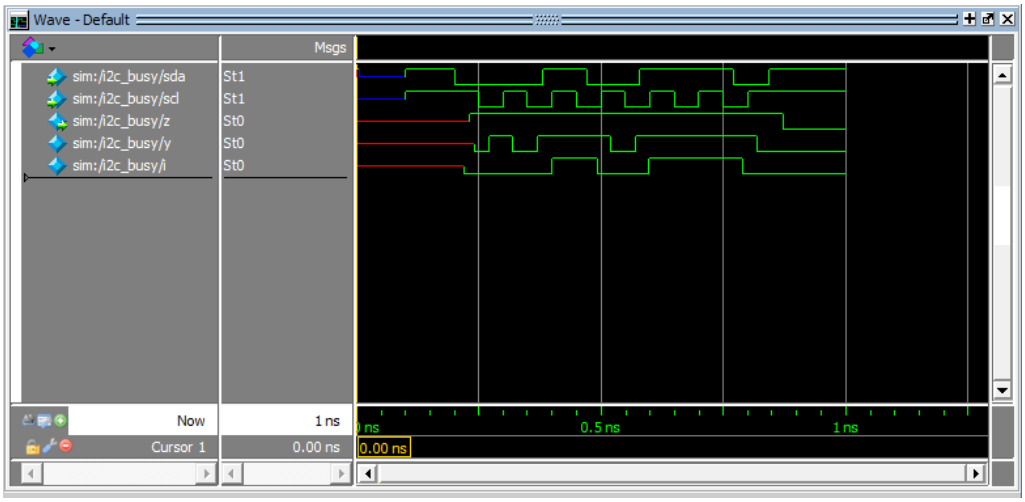
Ora i segnali in ingresso assumono il corretto valore logico, ma gli altri (y, i e z) permangono allo stato "StX" (indefinito). Questo è dovuto alla natura stessa del circuito, infatti entrambi i valori logici (0 e 1) sarebbero compatibili col funzionamento e non vi è modo per il momento, di discriminare in quale dei due stati si trovi il sistema.



Ripetendo la procedura illustrata sopra si provi ora ad abbassare il segnale SDA mantenendo il segnale SCL a livello alto e si faccia seguire a tale operazione un periodo di funzionamento di altri 100ps. Si noterà ora come dopo un certo ritardo, coerente con i ritardi definiti nella descrizione del circuito, tutti i segnali assumano un livello ben definito e la macchina si porta nello stato 'A' caratterizzato dalla variabile y allo stato basso.



Continuando tale procedura si può continuare a far evolvere la macchina a piacere e si possono analizzare tutti i valori dei segnali in essa presenti. Inoltre si può verificare come l'uscita si abbasserà solo quando agli ingressi verrà applicata la sequenza 01-11



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

Evidentemente il metodo appena descritto risulta alquanto lento, specie ove di dovesse ripetere l'intera procedura magari a seguito di una modifica nel circuito. Può pertanto risultare più agevole descrivere prima la sequenza degli stimoli da applicare, eseguire la simulazione ed eventualmente salvare i medesimi per future ri-simulazioni.

Supponiamo ad esempio di portare una leggera modifica al circuito, invertendo l'uscita che pertanto indicherà lo stato di "occupato" del bus I2C portando l'uscita a 0 che viceversa sarà alta quando il bus risulterà libero

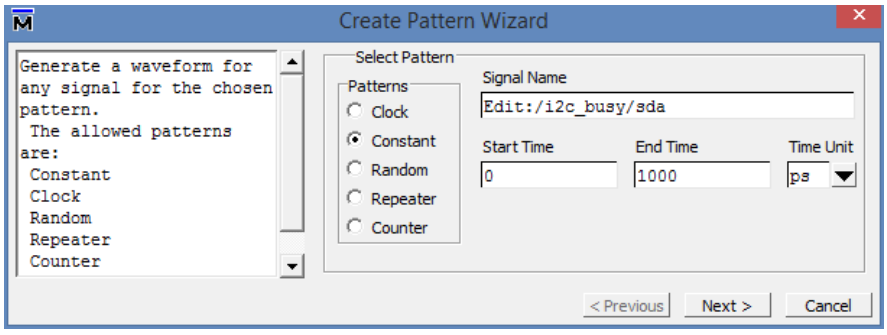
- All'interno della finestra Library cliccare col tasto destro sull'istanza "i2c_busy" e scegliere Edit.
- Nella finestra di testo che compare modificare il testo come segue:

```
...
assign #20 i=( sda & y);
assign #20 y=(~scl | i);
assign #20 z=~(~sda | y);
...
```

- 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 "busy_i2c" scegliere "`Create Wave`".

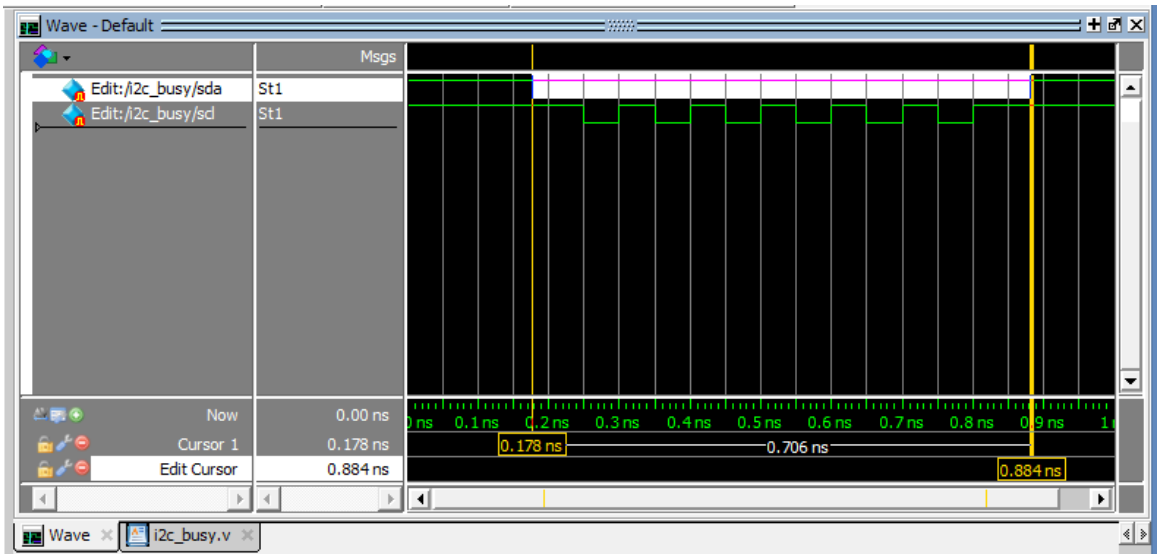
Nella finestra Wave appaiono ora 3 segnali (SDA,SCLe z)

- Cancellare il segnale z (`right clik > Edit > Delete`)
- Editare il segnale SDA perché assuma tra 0ps e 1000ps il valore costante 1:
    - `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 1
        - Finish
- Ripetere la medesima procedura anche per il segnale SCL

- Ripetendo ancora una volta la medesima procedura per il segnale SCL si può ridefinire definire il segnale come clock nel periodo compreso ad esempio tra 200 e 800 ns con periodo pari a 100ns e duty cyle 50
- Modificare ora la modalità di funzionamento del mouse
    - `Wave > Mouse Mode > Edit Mouse`
- Utilizzare il mouse per evidenziare una porzione del segnale SDA



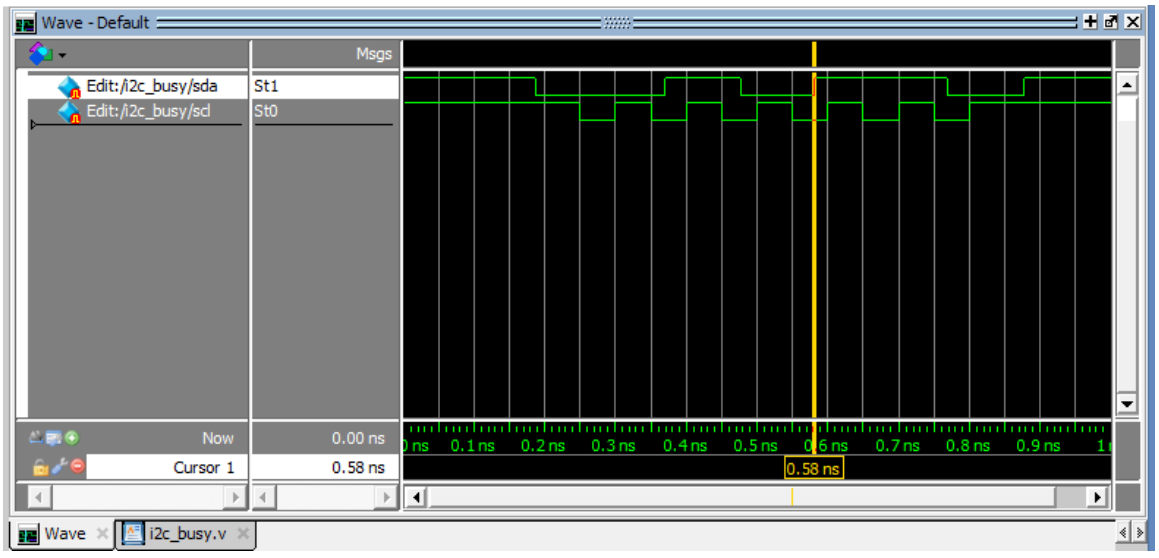
- Modificare il valore di questa porzione

    `Wave > Wave Editor > Invert`

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

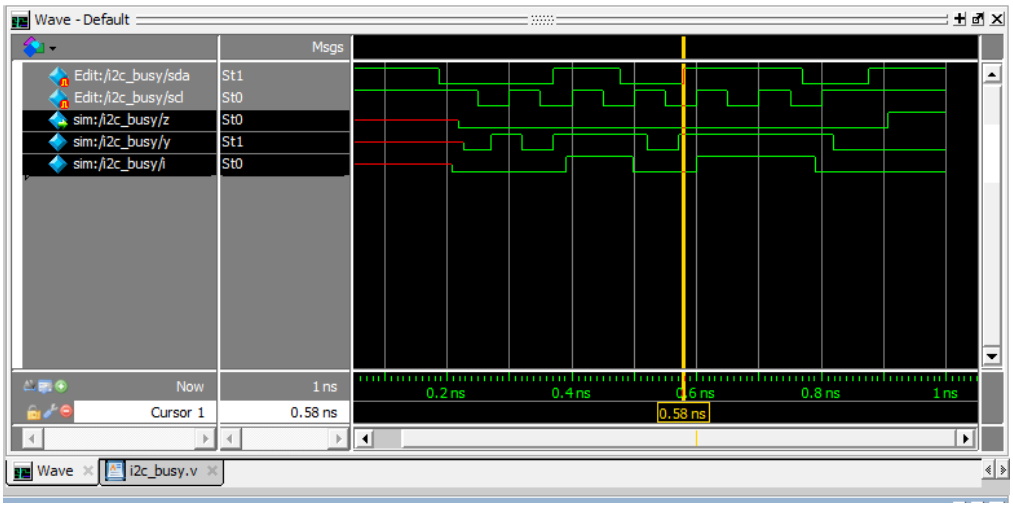
`Window > Toolbars > Wave Edit`

- Utilizzare questa tecnica per definire completamente le f.d.o con cui stimolare il circuito



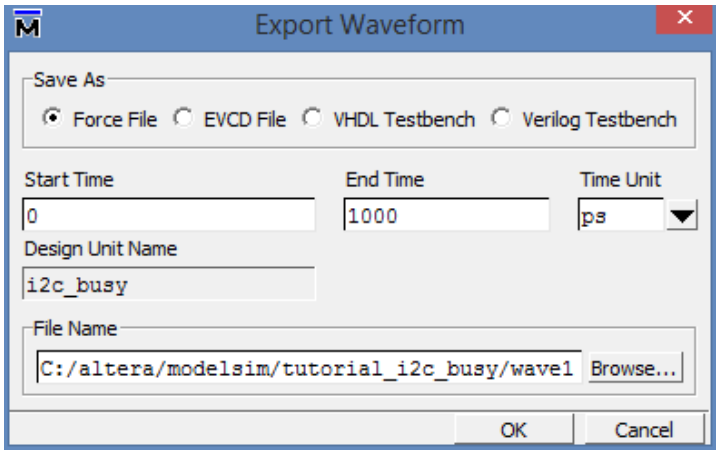
- Trascinare ora dalla finestra "Sim" alla finestra "Wave" tutti i segnali che si intendono monitorare (y,i e z)

- Nella finestra di console digitare
    - `Run -all`



Se si vogliono simulare più soluzioni circuitali attraverso i medesimi stimoli è possibile salvare il file che descrive gli stimoli.

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



  - OK

Il file così generato è nel formato .tcl e si compone di una serie di istruzioni che in pratica descrivono a modelsim quali passi seguire per realizzare varie operazioni (nel nostro caso la definizione delle f.d.o).

4. *Simulazione di un circuito attraverso un file di stimoli già salvato*

Si concluda la simulazione in corso

```
Simulate > End Simulate
```

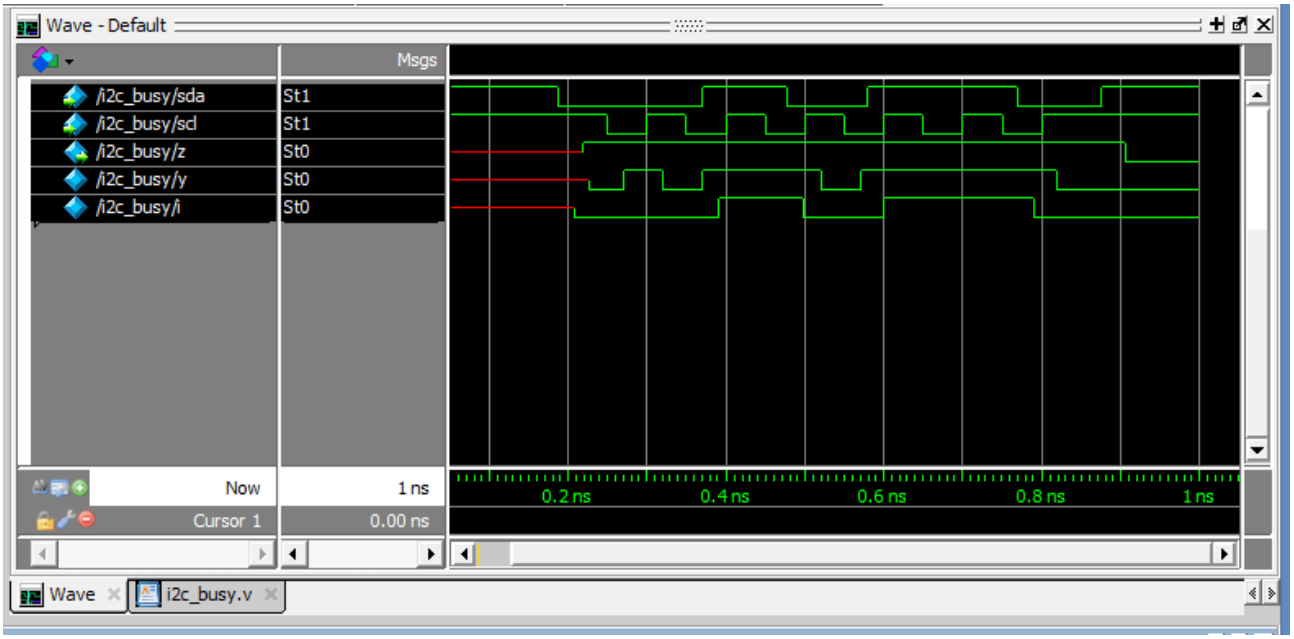
- Si ri-editi il file originale togliendo la negazione sull'uscita, si ricompili e si faccia ripartire la simulazione.
- Si trascinino i segnali da visualizzare all'interno della finestra wave

- Nella finestra di console si digiti:

  `> do nomefile.tcl` (dove ovviamente si faccia riferimento al nome dato al file che descriveva le f.d.o

  E successivamente

  > run –all



Bibliografia

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

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

[3] UM10204- I2C-bus specification and user manual Rev. 6 — 4 April 2014
http://www.nxp.com/documents/user_manual/UM10204.pdf

[4] Rajeev Madhavan - Quick Reference for Verilog HDL
http://web.stanford.edu/class/ee183/handouts_win2003/VerilogQuickRef.pdf

[5] Synopsys - FPGA Compiler II /FPGA Express Verilog HDL Reference Manual
http://classes.soe.ucsc.edu/cmpe225/Fall01/synver.pdf