Laboratorio di Sistemi per la Progettazione Automatica
a.a. 2008/09

Giuseppe Di Guglielmo
Università degli Studi Di Verona
Dipartimento di Informatica

Lezione 1: Compilazione e simulazione design VHDL

In questa lezione vengono introdotti i comandi principali per la compilazione e la simulazione di descrizioni VHDL mediante ModelSim di Mentor Graphics.

ModelSim è un simulatore per descrizioni in linguaggi VHDL/(System)Verilog/SystemC. Il simulatore supporta modelli di tipo comportamentale (behavioral), register transfer level (RTL) e gate-level (netlist). Modelsim è distribuito per differenti piattaforme (Linux, Solaris, Windows) ed è installato nei laboratori del gruppo Electronic System Design (ESD) presso il Dipartimento di Informatica dell'Università di Verona.

Questo tutorial è indirizzato all'utente che non ha precedenti esperienze col simulatore Modelsim. Esso presenta il flusso di configurazione dell'ambiente, di compilazione del progetto e di simulazione dello stesso. L'esempio utilizzato in questo tutorial è un piccolo esempio descritto in VHDL e vengono presentati solamente i comandi basilari del simulatore.

Questo tutorial è stato realizzato utilizzando la versione **6.4b di Modelsim** su piattaforma Linux. Di seguito i comandi riportati con sfondo grigio (`$`) sono da intendersi come comandi della console Linux, mentre i comandi riportati con sfondo rosa (`>`) sono da intedersi come comandi di ModelSim.

L'esempio utilizzato in questo tutorial descrive l'algoritmo euclideo per il calcolo del Massimo Comun Divisore - *Greatest Common Divisor* (*GCD*). In Figura 1 viene riportato lo schema a blocchi rappresentante l'algoritmo.

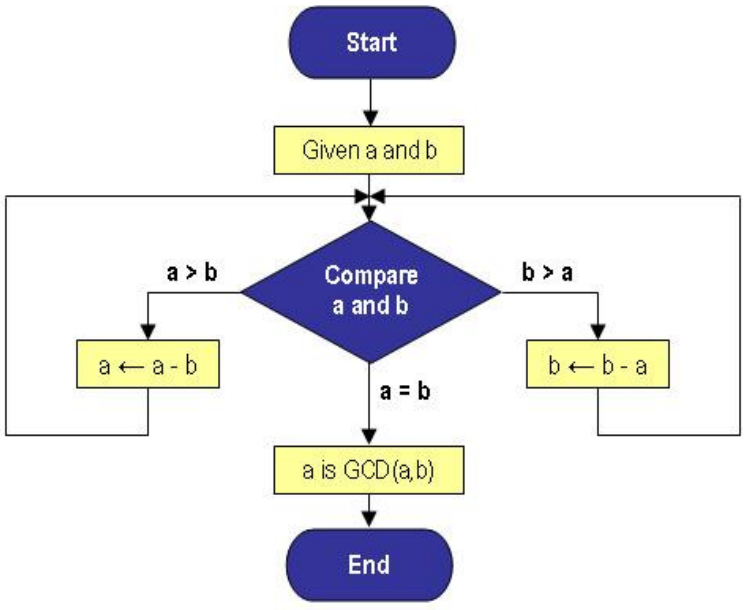

FIGURA 1: CALCOLO DEL MASSIMO COMUN DIVISORE (GCD)

Inizializzazione

In questa sezione viene descritto come creare una *working directory* e una *design library* in cui compilare il design. Per prima cosa è necessario autenticarsi su uno degli *host* del laboratorio ESD e quindi configurare l'ambiente.

Remote login

È necessario autenticarsi su una delle macchine del laboratorio ESD. Per far questo si può utilizzare una connessione *ssh* esportando il server grafico (*-X*) e richiedendo la compressione dei dati trasmessi (*-C*):

```
$ ssh -XC <_login_>@edalab-srv01.sci.univr.it
```

In alternativa, per esportare una sessione grafica, provare il comando:

```
$ X -query edalab-srv01.sci.univr.it
```

Impostazione variabili d'ambiente

È necessario esportare alcune variabili d'ambiente per poter utilizzare Modelsim, in particolare sul terminale aperto remotamente, eseguire il comando:

```
$ source /opt/EDA_Software/start_eda.bash
```

Scegliendo in sequenza:

```
Mentor Graphics(1) -> Latest Version(1)
```

Questo script deve essere eseguito ogni qualvolta si apre un terminale remoto prima della sessione di simulazione e/o compilazione.

Creazione directory di lavoro

Per prima cosa, al fine di mantenere organizzati e ordinati i vari file delle lezioni, è necessario creare una directory di lavoro.

```
$ mkdir lesson_1
$ cd lesson_1
```

I modelli VHDL compilati sono archiviati dal simulatore all'interno di una *design library*, chiamata di default `work`. Per ogni nuova entità viene creata una directory all'interno di `work` contenente la versione compilata del modello.

Nel nostro caso creeremo una nuova *design library*, con nome differente da `work`. Sarà pertanto necessario *rimapparla* per permettere a ModelSim di localizzare i design compilati. La *design library* `work` viene mappata di default.

Per creare una nuova design library e mapparla, eseguire i seguenti comandi:

```
$ vlib my_lib
$ vmap work $PWD/my_lib
$ ls my_lib
_info
```

Il comando `vlib` crea la *design library* e genera i file di configurazione che il simulatore VHDL leggerà al momento dell'avvio. Una volta creata la cartella `my_lib`, questo comando non dovrà più essere eseguito nella directory contenente i sorgenti VHDL, salvo voler creare una nuova *design library*.

Durante il mapping (`vmap`) ModelSim copia un file chiamato `modelsim.ini` nella directory corrente. Quando ModelSim viene invocato, esso legge questo file e usa le informazioni in esso

contenute per localizzare le *design library*. Per visualizzare il contenuto del file `modelsim.ini` è sufficiente invocare senza parametri il comando `vmap`.

```
$ vmap
```

Il comando `vdir` permette di elencare il contenuto della *design library*.

```
$ vdir
```

Compilazione del design

L'esempio utilizzato per introdurre le funzionalità basilari del simulatore VHDL descrive un circuito per il calcolo a 8 bit del massimo comun divisore.
La descrizione del circuito è presente nel file `gcd8.vdhl` contenuto a sua volta nel pacchetto compresso:

```
/home/gdg/teaching/spa_lab/aa_2008_09/lesson_1/gcd.tgz
```

Una copia di tale pacchetto è riportata anche sulla pagina ufficiale del Laboratorio del corso di Sistemi per la Progettazione Automatica a.a. 2008/09.

Alcune osservazioni prima di iniziare la compilazione:

1. Il linguaggio VHDL è restrittivo rispetto all'ordine in cui i file vengono compilati. Questo comporta che un file incluso da altri file (come una libreria) deve essere compilato prima di questi.
2. Durante la compilazione gli errori sintattici vengono evidenziati arrestando la compilazione.
3. Dopo una compilazione corretta, la fase di simulazione/debugging degli errori di progetto può cominciare. Il ciclo di progettazione sarà classicamente quello illustrato in Figura 2.

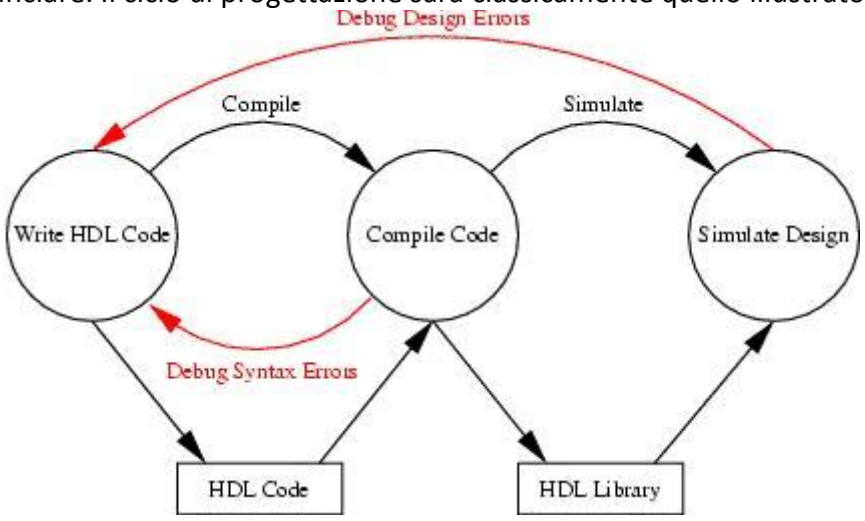

FIGURA 2: A CLASSICAL DESIGN CYCLE

Dopo aver decompresso il pacchetto, si deve eseguire il seguente comando per la compilazione:

```
$ vcom -93 gcd8.vhdl -source
```

Se la sua esecuzione produce il seguente risultato significa che la descrizione è stata compilata correttamente.

```
Model Technology ModelSim EE vcom 6.1b Compiler 2005.09 Sep 8 2005
2000.01 Jan 28 2000
-- Loading package standard
-- Compiling package gcd8_pack
-- Loading package std_logic_1164
-- Loading package gcd8_pack
-- Loading package numeric_bit
-- Compiling entity gcd8
```

```
-- Compiling architecture bhv of gcd8
```
Il flag `-source` richiede al compilatore di specificare l'eventuale linea su cui sono rilevati errori sintattici.

Nel caso si stesse lavorando con un progetto con molti file la gestione della compilazione potrebbe essere scomoda (dipendenze etc.). È possibile pertanto farsi generare automaticamente un file *Makefile* che gestisca correttamente le dipendenze del nostro progetto.
```
$ vmake my_lib > Makefile
```
In seguito dopo ogni modifica a qualche file di progetto è sufficiente invocare il comando make per ricompilare correttamente il progetto stesso.

Simulazione

Il simulatore VHDL viene avviato con il comando:
```
$ vsim &
```

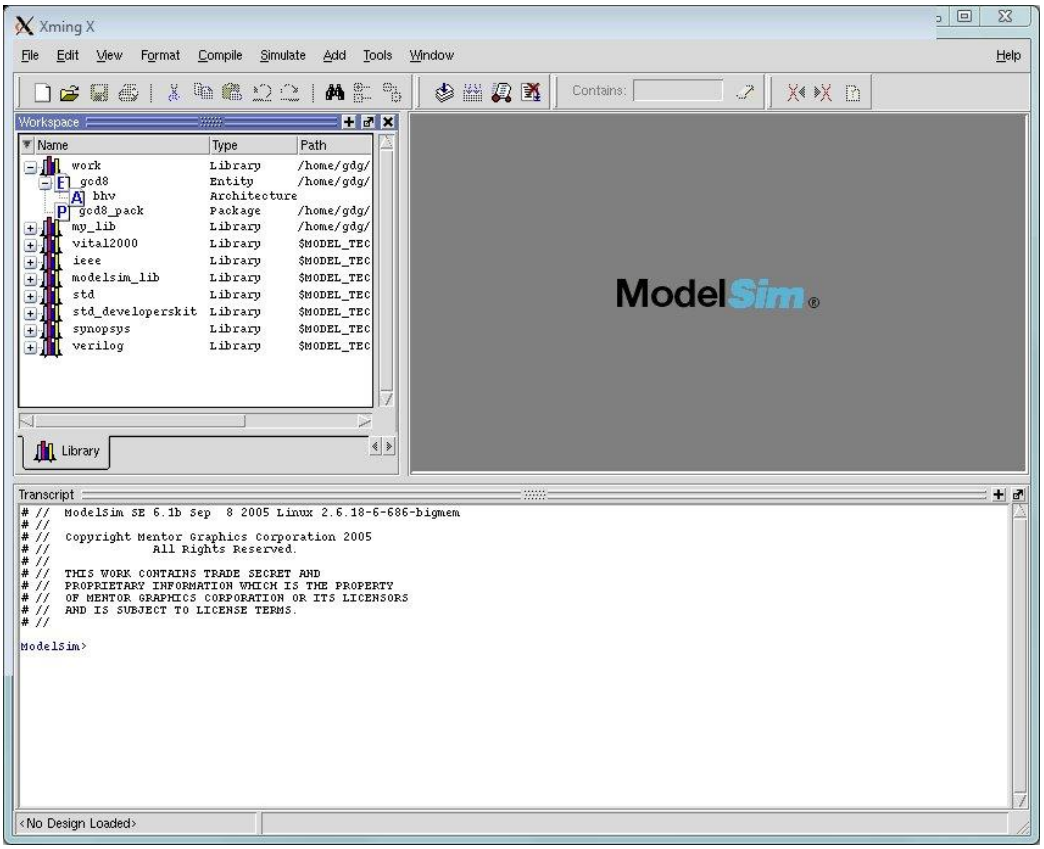

FIGURA 3: MODELSIM

La finestra di default all'avvio dovrebbe essere simile a quella in Figura 3, fatto salvo un messaggio di benvenuto al primo avvio.

In figura, nel pannello *Workspace,* la libreria `work` è aperta. Dal momento che la libreria `my_lib` è stata mappata su work, queste due librerie conterranno gli stessi design.

Il pannello *Transcript*, invece, mostra i messaggi del simulatore e accetta i comandi dell'utente (*ModelSim>* prompt).

Altri pannelli possono essere visualizzati dalla *Menubar* (*View*) o mediante il comando `view` da impartire sul panello *Transcript*. Più in generale, tutti i comandi possono essere eseguiti in due o

tre modi differenti: possono essere sempre impartiti mediante il *prompt* di Modelsim in *Transcript*. Alcuni comandi possono essere trovati nelle varie barre dei menu.

Caricare il design

Per caricare il design `gcd8`, è sufficiente il seguente comando:

```
ModelSim> vsim my_lib.gcd8
```

Si può anche osservare che il prompt in *Transcript* è cambiato da *ModelSim* a *VSIM*: ora si è in modalità simulatore.

Preparare la simulazione

Dopo aver caricato un design nel simulatore ma prima della simulazione è necessario configurare l'ambiente di simulazione/debugging. Pertanto è opportuno aprire le finestre che si ritengono necessarie, selezionare i segnali/variabili da tracciare etc.

Una buona scelta all'inizio è aprire le seguenti finestre:

- *Wave*, per visualizzare le forme d'onda
- *Objects*, per visualizzare i segnali e i loro valori
- *Source*, per visualizzare linea per linea il codice sorgente
- *Locals*, per visualizzare le variabili e i loro valori

```
VSIM> view objects
VSIM> view locals
VSIM> view source
VSIM> view wave -undock # Detach wave as a separate window
```

Per indicare al simulatore di quali segnali visualizzare le forme d'onda:

```
VSIM> add wave sim:/gcd8/xi
```

Oppure per visualizzare tutti i segnali:

```
VSIM> add wave *
```

Generazione degli stimoli

Vi è un ultima cosa da fare prima di iniziare la simulazione: è necessario generare gli stimoli per le porte in ingresso del design. Per far questo è possibile utilizzare il comando `force`, secondo la seguente sintassi:

```
force <nome porta/segnale> <valore> <tempo relativo>
```

per esempio i seguenti comandi forzeranno a *4* l'ingresso `xi` e a *8* l'ingresso `yi` del `gcd8` all'istante di simulazione *10* (unità di tempo = *ns*):

```
VSIM> force xi 00000100 10
```

Un po' più complesso è il seguente utilizzo del comando `force`:

```
VSIM> force reset 1 0, 0 {45 ns}
```

In questo caso il segnale di reset è posto a *1* al tempo *0* e a *0* al tempo *45ns*.

Infine:

```
VSIM> force clock 1 20, 0 40 –repeat 40
```

Il segnale di clock generato, è riportato in Figura 4:

- prima il clock è posto a *1* al tempo *20ns*;
- poi il clock è posto a *0* al tempo *40ns*;

- si ripete il ciclo ogni *40ns*

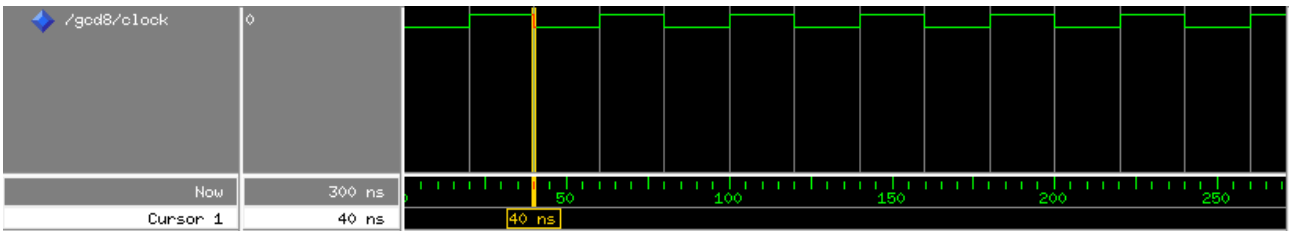

FIGURA 4: ANDAMENTO DEL SEGNALE DI CLOCK CON PERIODO 40NS

Nota:

- l'unità di misura (risoluzione) di default è definita nel file `modelsim.ini` ed è *ns*;
- l'unità di misura può essere omessa se si fa riferimento al default;
- il flag `-repeat` indica ogni quanto tempo si ripeterà il ciclo.

Avviare la simulazione

La simulazione del dispositivo si esegue mediante il comando `run` passandovi la durata della simulazione:

```
VSIM> run 400
```

Nota:

- la simulazione dura fino all'istante di tempo specificato come parametro al comando `run`. Se alcuni segnali sono stai forzati per un certo istante successivo a quello di termine simulazione, essi non verranno presi in considerazione;
- per i segnali i cui valori sono stati specificati mediante flag `-repeat` il loro comportamento periodico si protrae fino all'istante di fine simulazione.

Il simulatore può essere ri-inizializzato con il commando `restart`:

```
VSIM> restart -f
```

Il tempo di simulazione e i valori dei segnali e delle porte vengo inizializzati con i valori di default.

Nota: il flag `-f` serve a forzare la ri-inizializzazione, evitando richieste di conferma.

Creare un file di macro (.do) e modalità batch

Al fine di automatizzare la simulazione è possibile inserire i comandi in uno script, tipicamente il nome dello script per la generazione di stimoli è `stimuli.do`.
Per eseguire uno script eseguire il seguente comando dal prompt:

```
VSIM> do stimuli.do
```

I file di macro trovano applicazione, in particolar modo, se si esegue ModelSim in *modalità batch*. Un file di macro viene processato, senza avviare l'interfaccia grafica, invocando unicamente il simulatore (*VSIM*):

```
$ vsim my_lib.gcd8  -wlf trace.wlf -do gcd/stimuli.do -c
```

In questo caso la simulazione avviene senza fornire informazioni all'utente ma salvando i risultati della simulazione nel file `trace.wlf`.

Nota: perché il file WLF venga generato è necessario che almeno un segnale sia stato posto sotto osservazione mediante il comando `add wave <segnale>`.

Per visualizzare in un secondo momento il prodotto della simulazione:

```
$ vsim trace.wlf
```

Debugging

ModelSim offre varie alternative per effettuare debugging del design: analisi dei segnali o delle variabili, utilizzo di breakpoint, esecuzione codice linea per linea, utilizzo checkpoint etc.

Attenzione: prima che un errore possa essere corretto esso deve essere rilevato (detected). Nel caso di design veramente semplici è sufficiente stimolare gli ingressi con pochi vettori di test ed osservare le forme d'onda, ma per design reali è necessario scrivere un testbench al fine di automatizzare il processo.

Documentazione

La documentazione di Modelsim 6.4b è disponibile in formato PDF:

- `/opt/EDA_Software/mentor/modeltech/6.4b/modeltech/docs/pdfdocs`
- Una guida rapida (Quick Reference) ai comandi di Modelsim 6.4b è allegata al materiale fornito con la lezione.

Help in linea

Una guida testuale di ogni comando della console del simulatore può essere richiesta digitando `help` e il nome del comando

```
VSIM> help do
```

Sommario dei comandi di Modelsim

Questo è il sommario dei comadi base di Modelsim attinenti a questa lezione.

Comandi per la creazione della *design library* e per la compilazione.

`vlib` - Crea una nuova *design library.*
`vmap` - Mappa il nome della *design library* e il *path* alla stessa.
`vdir` - Lista il contenuto di una *design library*.
`vdel` - Rimuove un design da una *design library*.
`vmake` - Crea un Makefile per la design library specificata.
`vcom` - Compila il codice VHDL nella design library specificata.
`vsim` - Invoca il simulatore VSIM.
`verror i` - Stampa le informazioni riguardo messaggi di errore/warning, *i* = numero a 4-cifre

Comandi di simulazione.

`view` - Crea una finestra (wave, list, source etc).
`add wave` - Aggiunge un oggetto alla finestra Wave.
`force` - Forza uno stimolo per un segnale della descrizione VHDL.
`change` - Cambia il valore per una variabile, costante o generic VHDL.
`run` - Avvia la simulazione.
`restart` - Ricarica e riavvia la simulazione, l'opzione –f evita la richiesta di conferma.

`do` - Esegue un file di macro.

Comandi di debugging.

`examine` - Esamina il valore di una variabile o segnale.
`bp` - Aggiunge un breakpoint.
`step` - Esegue la prossima istruzione (dopo aver inserito un breakpoint).
`disablebp` - Disabilita un breakpoint.
`enablebp` - Abilita un breakpoint.
`checkpoint` - Salva uno stato del simulatore.
`restore` - Ripristino dello stato del simulatore.

Esercizi

- Discussione del *testbench* per il circuito gcd8.
- Compilare e simulare i circuiti
  `/home/gdg/teaching/spa_lab/aa_2008_09/lesson_1/bxx.tgz`
- Esplorare le funzionalità di debugging di Modelsim.
- Case Study: *diffeq*.
  `home/gdg/teaching/spa_lab/aa_2008_09/case_study/diffeq.tgz`
  - Compilazione e simulazione (il design è corredato da un testbench!)
  - Creazione di nuovi testset.
  - Correlazione con altri corsi (Calcolo Numerico).