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

Giuseppe Di Guglielmo
Università degli Studi Di Verona
Dipartimento di Informatica

Revisione: venerdì 3 aprile 2009 - 14.00

Lezione 4: Sintesi automatica con LeonardoSpectrum

In questa lezione vengono introdotti gli aspetti basilari della sintesi di circuiti digitali da livello RT a livello Gate mediante LeonardoSpectrum. LeonardoSpectrum è un prodotto di Mentor Graphics.

Questo tutorial è stato realizzato utilizzando la versione **2008a** di **LeonardoSpectrum** su piattaforma Solaris. Di seguito i comandi riportati con sfondo grigio (`$`) sono da intendersi come comandi della console Solaris, mentre i comandi riportati con sfondo **rosa e grassetto** indicano i percorsi da seguire o i pulsanti dell'interfaccia grafica.

Introduzione

Dopo che la descrizione del sistema a livello RT è stata sottoposta a verifica, il dispositivo è pronto per la sintesi a livello porte logiche. Figura 1 schematizza l'intero flusso di progettazione di un sistema *embedded*, e evidenzia il punto esatto in cui è inserita l'attività di sintesi.

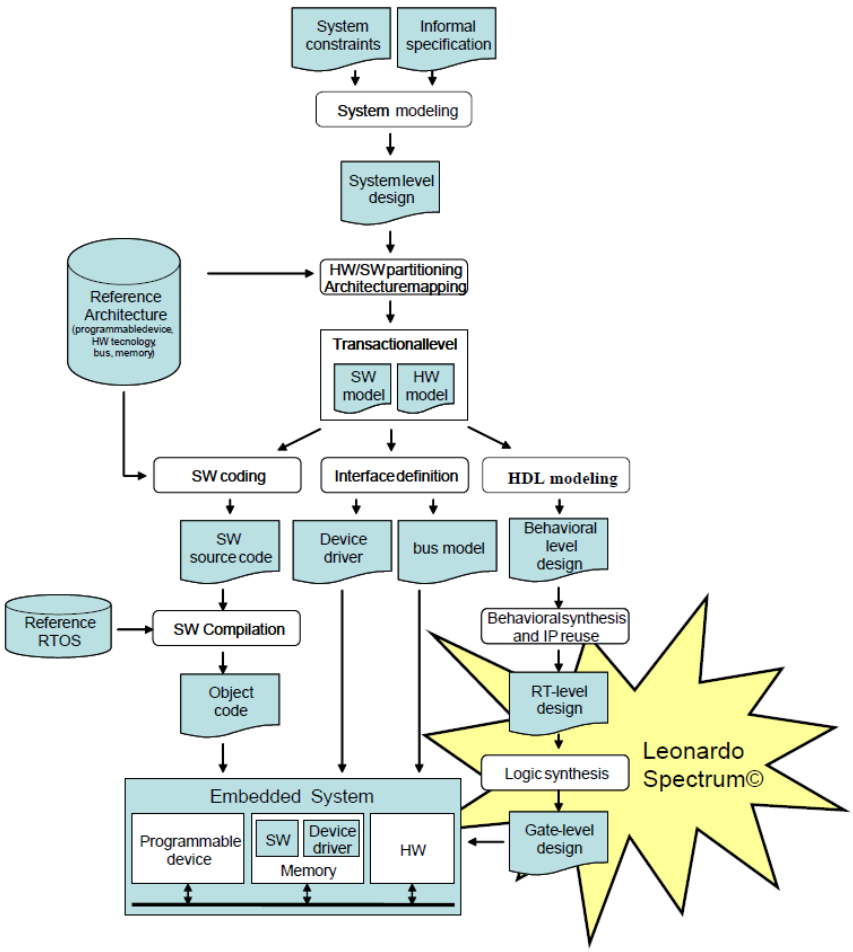

Figura 1: Flusso di progettazione di un sistema embedded.

In questa fase il dispositivo descritto a livello RT viene tradotto in un insieme di blocchi elementari a livello porte logiche. Dal momento che sono disponibili differenti architetture, la stessa descrizione comportamentale può essere tradotta in differenti implementazioni a livello Gate. Una varietà di strumenti automatici aiutano il progettista in questo compito, offrendo la possibilità di sintetizzare per via automatica il codice RTL in base all'architettura *obiettivo* scelta. LeonardoSpectrum è uno fra i più diffusi e apprezzati strumenti per la sintesi.

Sintetizzabilità

A causa delle difficoltà nell'interpretazione della semantica dei linguaggi di descrizione dell'hardware (HDL), spesso la sintassi accettata da uno strumento di sintesi è un sottoinsieme dei costrutti sintetizzabili in un dato HDL. Per questo motivo, una fase di sintesi architetturale (*architectural synthesis*) è necessaria prima di sintetizzare il circuito a livello gate (si veda Figura 1).

LeonardoSpectrum

Una guida completa all'utilizzo di LeonardoSpectrum è disponibile in
`/opt/EDA_Software/mentor/leonardo/2008a/doc/`
Solo i concetti chiave del suo utilizzo vengono presentati in questa lezione.

Remote login

È necessario autenticarsi su una specifica macchina del laboratorio ESD (piattaforma Solaris) con un utente comune. Per far questo si può utilizzare una connessione *ssh* esportando il server grafico (*-X*) e richiedendo la compressione dei dati trasmessi (*-C*):

```
$ ssh -XC proxy@vlsi01.sci.univr.it
```

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

```
$ X -query vlsi01.sci.univr.it :2 vt8
```

La password per l'utente *proxy* verrà comunicata a lezione.

Impostazione variabili d'ambiente

È necessario esportare alcune variabili d'ambiente per poter utilizzare HDL Designer, 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 progettazione.

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. L'utilizzo di un utente comune richiede una piccola accortezza nell'organizzare lo spazio di lavoro: sarà necessario creare una directory distinta per ciascun gruppo di lavoro (ciascun gruppo ha un proprio identificativo, *group_id*):

```
$ mkdir -p spa_0809/lesson4/group_id
$ cd spa_0809/lesson4/group_id
```

Ora è necessario copiare i file di riferimento per quest'esercitazione mediante il comando *scp*, la cui sintassi è:

```
scp  user@host:remote_path local_path
```

Nel caso particolare sarà quindi

```
$ scp <user_id>@edalab-srv01:\
/home/gdg/teaching/spa_lab/aa_2008_09/lesson_4/vhdl.tgz .
$ gzip -d vhdl.tgz
$ tar xf vhdl.tar
```

Avvio di Leonardo Spectrum

Per avviare LeonardoSpectrum è sufficiente fornire il seguente comando sulla console linux:

```
$ leonardo
```

Avvertenza: attualmente il Laboratorio ESD dispone di 10 licenze di LeonardoSpectrum. Si chiede pertanto di eseguire questo tutorial in gruppi.

Dopo aver avviato l'applicativo, selezionare il tipo di licenza selezionando la voce
LeonardoSpectrum Level 3
e mantenendo selezionato
Run license selection next time

Eventuali *wizard* dovuti al primo avvio possono essere ignorati.

Sintesi di un descrizione RTL

Dopo l'avvio di LeonardoSpectrum, selezionare **Tools >> FlowTabs** per abilitare la procedura guidata di sintesi. A questo punto verranno visualizzate le seguenti linguette corrispondenti ciascuna a una fase del processo di sintesi:

- **Technology**
- **Input**
- **Constraints**
- **Optimize**
- **Report**
- **Output**

Technology

La prima scelta da effettuare è quella della *tecnologia* di riferimento per il circuito sintetizzato , le due principali categorie tra cui scegliere sono *Application Specific Integrated Circuit* (ASIC) e *Field Programmable Gate Array* (FPGA). La scelta fra queste due tecnologie richiede al progettista di valutare vincoli di costi, disponibilità di strumenti e efficienza relativi al prodotto finale.

La tecnologia ASIC tradizionalmente viene adottata per implementare applicazioni specializzate su larga scala e offre le migliori prestazioni. Come rovescio della medaglia, il costo di tale tecnologia è elevato.

La tecnologia FPGA tradizionalmente viene adottata per applicazioni che non permettono tempi di progettazione lunghi e costosi. La possibilità di ri-programmare i dispositivi FPGA è il maggior vantaggio rispetto all'approccio ASIC. In particolare, nel caso di un errore identificato in fase di verifica, il dispositivo può essere riprogrammato e l'errore risolto.

In questa lezione verrà utilizzata come tecnologia di riferimento
ASIC >> Sample >> SCL05u
Infine, per caricare le librerie premere **Load Library**. La finestra in alto a destra riporterà le informazioni relative al processo di caricamento delle librerie. La finestra in basso a destra riporterà i comandi corrispondenti all'interazione con l'interfaccia grafica, questi comandi possono essere inseriti manualmente sempre nella stessa finestra.

Input

È necessario ora scegliere la descrizione RTL da sintetizzare, per far questo è sufficiente premere **Open files…**, selezionare `vhdl/fsm/fsm.vhdl` e infine premere **Read**. Durante questa fase vengono portate a termine due operazioni:

1. La descrizione HDL è verificata sintatticamente, le dipendenze vengono verificate e i parametri `generic` risolti.
2. La descrizione HDL viene sintetizzata in un formato generico (EDIF) e caricata in memoria. Il dispositivo è composto da componenti generiche, in seguito queste componenti verranno rimpiazzate con gli operatori della tecnologia selezionata.

Il formato EDIF (Electronic Design Interchange Format) viene tipicamente utilizzato come formato indipendente per il trasferimento e la rappresentazione di un progetto in forma di schematici e netlist.

Constraints

Il progettista può specificare alcuni vincoli costruttivi, quali ad esempio la frequenza di clock, la durata del ciclo di clock, i ritardi fra componenti.

Selezionare **Specify Clock Frequency**, inserire come valore *100 MHz* e quindi premere **Apply**.

Optimize

Ci sono essenzialmente due parametri principali di ottimizzazione in LeonardoSpectrum: area e ritardo. I campi **Optimize For >> Area** e **Optimize For >> Delay** permettono di scegliere quale criterio usare per la sintesi, mentre **Optimize Effort** permette di specificare il tempo speso dallo strumento di sintesi per ottimizzare a partire dai vincoli impostati.

Premere **Optimize** per iniziare l'attività di sintesi (per descrizioni complesse questa fase può richiedere tempi elevati).

Per visualizzare la rappresentazione grafica della netlist definita mediante le componenti della libreria di riferimento premere il pulsante **View Technology Schematic**.

Report

Si possono salvare le informazioni (log) di sintesi in un file testuale premendo su **Report Filename**, inserendo come file `vhdl/fsm/fsm.log` e infine premendo **Report Area**. A partire da queste informazioni il progettista può decidere se mantenere il prodotto della sintesi o effettuare delle modifiche dei vincoli e quindi procedere attraverso una nuova fase di ottimizzazione.

Output

In questa fase finale progettista può salvare il prodotto della sintesi in differenti formati. In particolare in questa lezione si richiede di scegliere come formato finale *VHDL* nel campo **Format**, di selezionare come Filename `vhdl/fsm/fsm_synth.vhdl` e infine di premere **Write**.

Sintesi Architetturale

La sintesi architetturale è una fase della progettazione di un sistema embedded in cui la descrizione algoritmica di un dispositivo viene interpretata al fine di produrre la descrizione del corrispondente componente hardware.

Per comprendere la necessità di tale fase si consideri la descrizione algoritmica del dispositivo *gcd8* descritto in

`vhdl/gcd8/gcd8.vhdl`

e il flusso di sintesi proposto in questa lezione.

In particolare si riporta di seguito il processo contenuto nel modulo `gcd8`.

```
24. process
25. variable x, y, temp :
26.   UNSIGNED (SIZE-1 DOWNTO 0);
27. begin
28.   wait until clock = '1';
29.   x := xi;
30.   y := yi;
31.   while (x > 0) loop
32.     if (x <= y) then
33.       temp:=y;
34.       y := x;
35.       x:=temp;
36.     end if;
36.     x := x - y;
38.   end loop;
39.   ou <= y;
40. end process;
```

LeonardoSpectrum durante la fase di **Input** rileva un problema nella descrizione:

```
[...]
-- Compiling root entity gcd8(gcd8)
"/export/home/proxy/spa_0809/lesson4/gdg/vhdl/gcd8/gcd8.vhdl", line 31:
Error, expression does not evaluate to a constant.
[...]
```

Il sintetizzatore non riesce a determinare a priori se il ciclo *while* è “srotolabile” (cioè se esso viene eseguito un numero finito di volte).

Una possibile soluzione a questo problema è modificare la descrizione in modo da permetterne la sintetizzabilità, come proposto di seguito:

```
24. process
25. variable x, y, temp :
```

```
26.   UNSIGNED (SIZE-1 DOWNTO 0);
27. begin
28.   wait until clock = '1';
29.   x := xi;
30.   y := yi;
31.   for i in 0 to 255 loop
32.     if ( x /= y ) then
33.       if (x <= y) then
34.         temp:=y;
35.         y := x;
36.         x:=temp;
37.       end if;
38.       x := x - y;
39.     else
40.       ou <= y;
41.     end if;
42.   end loop;
43.   ou <= y;
44. end process;
```

In questo caso particolare, per rendere sintetizzabile la descrizione, si è osservato che entrambi gli ingressi del modulo GCD8 sono interi senza segno rappresentati su *8 bit* corrispondenti al tipo `UNSIGNED (7 DOWNTO 0)` definito nella libreria `IEEE.std_logic_arith`. Il massimo valore rappresentabile pertanto è 255 (= 2^8-1) e rappresenta il massimo numero di volte che il ciclo `while ( x > 0 )` potrebbe essere eseguito nel caso in cui venissero forniti come ingressi *x <= 255* e *y <= 1*.

Analogamente al caso dei "cicli non srotolabili", vi sono altre situazioni in cui l'utilizzo di costrutti HDL può dare luogo a descrizioni non sintetizzabili. Tali descrizioni generalmente possono essere modificate dal progettista per essere rese sintetizzabili. Ad esempio il costrutto

```
wait for 0 ns;
```

non è permesso nella fase di sintesi di una descrizione HDL, ma è necessario in fase di simulazione per assicurare il passaggio di un tempo Δ prima dell'aggiornamento dei segnali. Per permettere la sintesi è sufficiente commentare tale costrutto.

Analisi delle Netlist

Si consideri la descrizione HDL

```
vhdl/ex/Fig8_14.vhd
```

La netlist generata a partire dalla libreria *ASIC – Sample – SCL05u*, così come viene rappresentata da LeonardoSpectrum, è riportata in Figura 2.

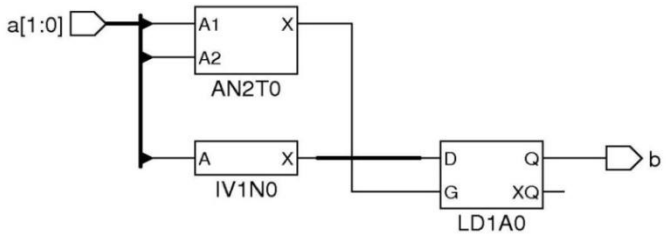

Figura 2: Esempio di netlist visualizzabile in LeonardoSpectrum

Le componenti (*Technology Cells*) di tale netlist possono essere analizzate mediante la funzione *Design Browser*, per far ciò

- premere il pulsante **Design Browser**;
- selezionare la componente *LD1A0* corrispondente all'elemento di memoria della netlist (Figura 2);
- seguire il percorso **work>>latch_example>>test1>>Cells>>lat_b(LD1A0)>>Pins**

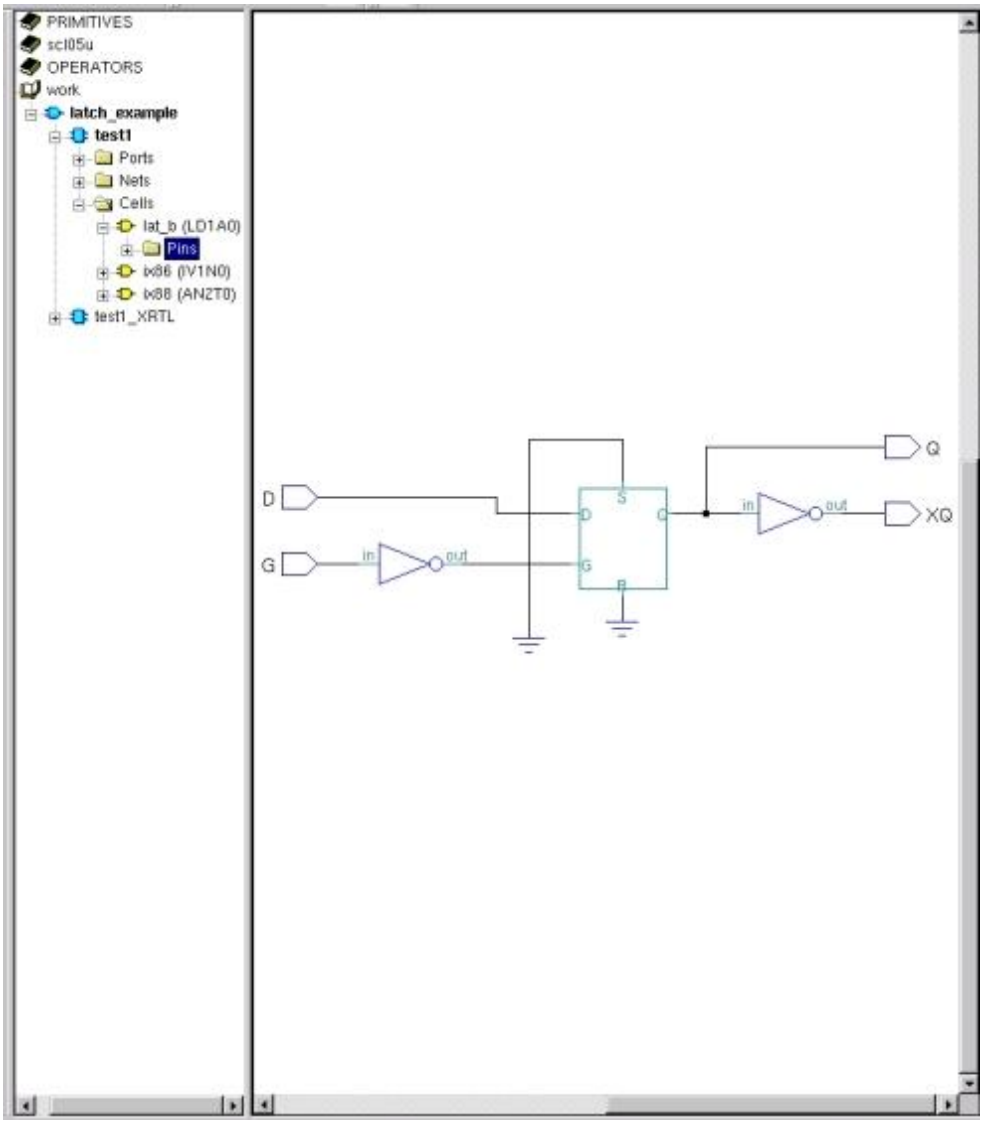

Figura 3: Pannello Design Browser.

Figura 3 mostra l'implementazione della componente *LD1A0* mediante le primitive della logica circuitale.

Un ulteriore analisi si può fare generando la descrizione HDL del circuito, esplicitando la definizione delle componenti della libreria scelta. Per fare questo, in fase di **Output**, selezionare **Downto >> Primitives** e quindi scegliere come formato finale *VHDL* nel campo **Format**, prima di premere **Write**.

È possibile applicare il flusso di sintesi proposto a ciascuna delle descrizioni HDL presenti nella cartella

```
vhdl/roth/
```

Seguono alcune considerazioni relative a ciascuna descrizione:

- `vhdl/ex/Fig8_14.vhd`
  La sintesi di questa descrizione produce una netlist contenente un elemento di memoria. Questo avviene tipicamente quando, per qualche ramo condizionale, non viene specificato il comportamento dei segnali di uscita, in tal caso i segnali devono mantenere il valore dell'istante precedente.

```
case a is
      when 0 => b <= '1';
      when 1 => b <= '0';
      when 2 => b <= '1';
      when others => null; -- per le restanti alternative
                           -- non è specificato il valore
                           -- di b
end case;
```

  Si confronti tale codice con il comportamento della netlist in Figura 2:
  - a fronte di un ingresso `A[1:0] = 11` (rappresentazione binaria di 3)
  - la componente *AN2T0* (AND logico) ha in uscita il valore `1`
  - a sua volta `1` sarà il valore assegnato alla porta in ingresso *G* per l'elemento di memoria *LD1A0* (il segnale G è asserito a livello basso, vedi l'invertitore in ingresso a G nello schema di LD1A0)
  - per come è definito l'elemento *LD1A0* esso mantiene il valore sulla sua uscita *Q*
  - Q è collegata a sua volta alla porta in uscita *b*
- `vhdl/ex/Fig8_15a.vhd`
  A differenza del caso precedente, in questa descrizione per ogni ramo condizionale è specificato il comportamento delle porte in uscita e di conseguenza non vengono introdotti elementi di memoria.
- `vhdl/ex/Fig8_18.vhd`
  In questa descrizione è riportata la macchina a stati finiti delle lezioni precedenti. Tale FSM si discosta da quelle già viste solo per il fatto che per ciascuno stato viene esplicitata la codifica:

```
constant S0: s_type := 0;  -- 000
constant S1: s_type := 4;  -- 100
constant S2: s_type := 5;  -- 101
constant S3: s_type := 7;  -- 111
constant S4: s_type := 6;  -- 110
constant S5: s_type := 3;  -- 011
constant s6: s_type := 2;  -- 010
```

  Il prodotto della sintesi automatica di questa descrizione evidenzia la presenza di tre elementi di memoria, come del resto era atteso.

Esercizi

- Applicare manualmente le regole di sintesi alla descrizione in
  `vhdl/ex/Fig8_17.vhd`
  e confrontare il risultato con il prodotto della sintesi automatica.
- In queste lezioni, il modulo *GCD8* è stato presentato mediante differenti descrizioni HDL

- a livello RT - non sintetizzabile:
  `vhdl/gcd8/gcd8.vhdl`
- a livello RT - sintetizzabile:
  `vhdl/gcd8/gcd8_behav_synth.vhdl`
- a livello porte logiche, prodotto della sintesi automatica mediante LeonardoSpectrum
  `vhdl/gcd8/gcd8_synth.vhdl`

Applicare manualmente le regole di sintesi alla descrizione a livello RT (sintetizzabile), codificare il risultato in VHDL e confrontare il comportamento con quello delle altre tre versioni, mediante la simulazione in ModelSim.