Il linguaggio SystemVerilog nel progetto di FPGA

Agostino Rolando
Technical engineer
Ingegneria - Design Methodologies & Tools - DSP design
Sede di Genova
Selex Communications
(Gruppo Finmeccanica)

SystemVerilog, potenziamento del consolidato linguaggio Verilog IEEE 1364, è il primo linguaggio standard unificato per la descrizione e la verifica di hardware per l'ambiente industriale (HDVL)

```
interface chip_bus (input wire clk);
  wire        request, grant, ready;
  wire [47:0] address;
  wire [63:0] data;
endinterface

module CPU (chip_bus io);
  ...
endmodule

module RAM (chip_bus pins);
  ...
endmodule

module top;
  wire clk;
  chip_bus a(clk);

  RAM mem(a);
  CPU cpu(a);
endmodule
```

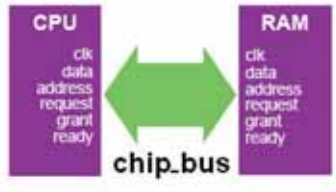

Fig. 1 - Esempio di codifica per la realizzazione di un sistema "bus oriented"

Sviluppato originariamente da Accellera allo scopo di incrementare sensibilmente la produttività progettuale di chip a elevato gate-count, SystemVerilog si basa su IP che utilizzano intensivamente le risorse di bus.
È orientato all'implementazione di chip con riguardo particolare al flusso di verifica ed è indicato nella descrizione progettuale di tipo system-level.
Fin dalla sua introduzione nel 2005, SystemVerilog è stato considerato il linguaggio ideale per la sua peculiarità di fornire una rappresentazione omogenea per le fasi di progetto e di verifica.
Nonostante i suoi molteplici aspetti positivi, quali la stretta parentela con le parti migliori del Verilog, con i linguaggi dotati dei costrutti ad "assertion" e il VHDL, l'impiego di SystemVerilog è stato piuttosto limitato, come in genere limitata è la diffusione delle metodologie innovative di progetto HDL-based. Tuttavia, la popolarità di questo linguaggio è in crescita da quando si è assistito a un incremento nell'utilizzo del tool, a cominciare in primis dai team di verifica per passare, in seguito, ai progettisti di ASIC e, attualmente, pure ai progettisti di FPGA.
Tra le principali estensioni di SystemVerilog nei riguardi del Verilog tradizionale, vi sono le seguenti:
- Supporto per modellazione e verifica al livello di astrazione transazionale: l'interfaccia diretta di programmazione di SystemVerilog (DPI) permette di richiamare funzioni C/C++ o SystemC e viceversa. SystemVerilog è così il primo linguaggio Verilog-based a consentire efficienti co-simulazioni con blocchi SystemC e rappresenta quindi un valido tramite fra l'approccio al disegno di tipo system-level e le fasi di implementazione e verifica a livello di chip.
- Un set di estensioni per affrontare requisiti avanzati di progetto. Ad esempio, modellazione di interfacce per semplificare lo sviluppo di progetti che fanno uso esteso di bus; rimozione delle restrizioni sulle connessioni alle porte di I/O dei moduli, permettendo di gestire quindi qualsiasi tipo di dato su ciascun lato della porta; tipi di dato esteso, per permettere la modellazione C; avan-

zata protezione per le IP per mezzo del nesting di moduli localmente ai moduli di appartenenza e, quindi, rendendoli non visibili alle altre parti del progetto.
- Un meccanismo innovativo per supportare la verifica assertion-based (ABV), consentendo quindi la metodologia "design for verification". In SystemVerilog, le informazioni relative alle "assertion" sono primitive del linguaggio, il che permette di alleviare il progettista dal dover realizzare moduli specifici, costrutti o chiamate PLI come viene fatto nel Verilog tradizionale. Le assertion embedded permettono di mettere in atto l'intento del progettista in termini di funzionalità e di constraint e vengono verificate in sede di simulazione prima dell'applicazione, a fronte di un qualunque metodo formale o dinamico. Questo approccio aiuta a evitare errori di compilazione, incrementa la precisione del test, semplifica i testbench e consente il riuso del test stesso. La completa controllabilità e osservabilità dei nodi circuitali interni consentita dall'ABV dà luogo a una significativa riduzione del tempo di debug progettuale, in alcuni casi fino al 50%. Un aspetto particolarmente importante è rappresentato dal fatto che una tale controllabilità e osservabilità estende il campo di applicabilità dei tool di advanced design & verification, essendo un'area di particolare interesse quella della sintesi assertion-based.
- Innovative caratteristiche per il supporto di modelli hardware e testbench che utilizzano tecniche di tipo object-oriented; i testbench hanno la possibilità di essere ri-usabili. Ad esempio, la combinazione della metodologia di interfaccia di SystemVerilog con le tecniche di creazione di testbench object-oriented permette di implementare con facilità un'efficace metodica di verifica constraint-driven. I constraint sono infatti immessi per mezzo di assertion embedded, le quali esprimono le proprietà del progetto che devono essere sottoposte a test di tipo "true" o solo coperte in sede di verifica. Tali assertion possono essere riusate nello sviluppo di un SoC o per un disegno IP-based. Fino a oggi, questa capacità non è stata uguagliata da altri tool o linguaggi implementativi.
Nel passato, i gruppi di progetto non si sono preoccupati di produrre codice particolarmente efficiente, delegando al testing le fasi di raffinamento. In seguito, ovviamente, i team di progetto hanno proceduto molto oltre questo approccio limitante, mettendo in atto una progettazione orientata al test e collaborando attivamente con i team di verifica. Ma le aspettative iniziali di riunire sinergicamente le fasi di design e di testing non hanno avuto gli esiti sperati in quanto i tool, le metodologie e i linguaggi non si sono armonizzati adeguatamente. SystemVerilog è stato pensato per riavvicinare le fasi suddette, poiché consente sia ai progettisti che si occupano di verifica che ai progettisti hardware di utilizzare il medesimo linguaggio HDL e di mantenerlo per tutta la catena di progetto, o per meglio dire, a "parlarsi" con lo stesso linguaggio.
Con le premesse suddette, ci si potrebbe aspettare che SystemVerilog si sia diffuso ovunque. In effetti non è proprio così: i progettisti firmware, infatti, costituiscono una categoria piuttosto tradizionalista e, a meno di un'esplicita costrizione, tendono a non cambiare strada.
Se possono raggiungere lo scopo con le metodologie e i tool che conoscono da anni, non si mettono in discussione convinti di perdere tempo ed efficienza. In apparenza sarebbero molteplici le ragioni a loro favore per non cambiare strada: l'esperienza acquisita con i tool e le metodologie, la comodità di poter lavorare con tecnologie consolidate, il fatto di poter riutilizzare parti di progetti già testati, e così via. Contribuisce a questa tendenza la carenza di un supporto EDA a vasto raggio per SystemVerilog. Anche se vi sono evidenti vantaggi, è sufficiente che vi sia un piccolo "bug" nella catena dei tool

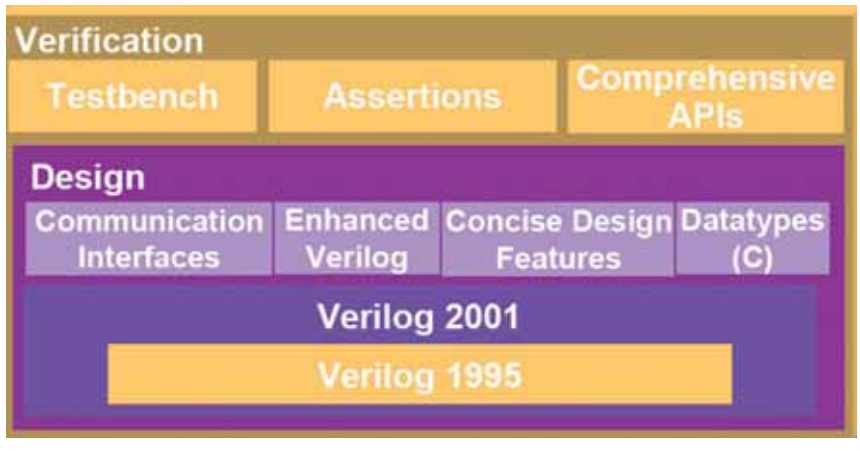

Fig. 2 - Parti della struttura di System Verilog

Verilog

```
module fifo (clk, rstp, din_src,
 din_dst, din_data,readp,writep,
 dout_src,dout_dst, dout_data,
 emptyp, fullp);
 input              clk;
 input              rstp;
 input bit [7:0]    din_src;
 input bit [7:0]    din_dst;
 input bit [31:0]   din_data;
 input              readp;
 input              writep;
 output bit [7:0]   dout_src;
 output bit [7:0]   dout_dst;
 output bit [31:0]  dout_data;
 output  logic      emptyp;
 output  logic      fullp;
.
.
.
```

System Verilog

```
typedef struct {
        bit [7:0] src;
        bit [7:0] dst;
        bit [31:0] data;
} packet_t;
```

```
module fifo (
 input              clk,
 input              rstp;
 input packet_t     din,
 input              readp;
 input              writep;
 output packet_t    dout;
 output  logic      emptyp;
 output  logic      fullp
);
```

Fig. 3 - La leggibilità del codice System Verilog è migliore rispetto al Verilog tradizionale

per convincere il progettista hardware che sia ancora "troppo presto" per passare a un nuovo strumento.

Attrattive di SystemVerilog

SystemVerilog riveste particolari attrattive nelle due aree che hanno dato maggiore impulso al suo sviluppo: la verifica e il design. L'interesse da parte degli ingegneri di test è evidente, potendo disporre di un linguaggio assertion-based che è consistente con il progetto, oltre a costrutti avanzati per i testbench e un'interfaccia flessibile verso altri linguaggi come C e C++.

Gli ingegneri di progetto trovano attraente SystemVerilog per la sua concisione, la leggibilità e l'elevato livello di astrazione insieme al supporto per tipi di dato user-defined ed enumerativi e alle estensioni C-like. SystemVerilog fornisce inoltre costrutti di ausilio alla documentazione del progetto e alla sua simulazione. Incorporare queste possibilità nel codice sorgente consente il suo riutilizzo e semplifica il debug.

Ma vi è un aspetto di SystemVerilog che viene frequentemente addotto come motivazione principale per migrare verso questo nuovo HDL: l'innovativo costrutto di interfaccia. A differenza del metodo tradizionale, foriero di errori, consistente nel copia-incolla di un gran numero di segnali di interfaccia e del relativo codice associato a ciascun blocco, un'interfaccia può essere definita una volta per tutte e facilmente chiamata attraverso l'intero disegno mediante un semplice costrutto. Sono passati i tempi in cui una modifica nell'interfaccia si doveva comunicare a tutti i progettisti del team ed editare manualmente ciascuno per il proprio codice di competenza e i relativi testbench.

Per ingegneri responsabili sia del testing che del progetto ha senso utilizzare un unico linguaggio; perciò, questa tipologia di gruppo di lavoro è stata la prima ad adottare SystemVerilog.

Il successivo gruppo ad adottare SystemVerilog è stato quello dei progettisti ASIC non direttamente coinvolti nel testing. Avendo a che fare con progetti di ampie dimensioni e di una certa complessità, questa categoria di progettisti è stata attratta dalla possibilità di scrivere del codice leggibile e molto più compatto, quindi facile da comprendere. In questo modo, le migliorie che si ritrovano in SystemVerilog rispetto a Verilog, spesso e volentieri derivate dall'importazione di costrutti propri del VHDL, hanno reso molto più agevole il lavoro del progettista e velocizzato le fasi di stesura e simulazione.

Conseguentemente, una volta che i progettisti di ASIC hanno iniziato a lavorare con SystemVerilog, la successiva categoria che ha adottato questo linguaggio ha riguardato quei gruppi di lavoro interessati alla prototipazione di ASIC tramite FPGA. Questi team hanno utilizzato il medesimo codice SystemVerilog prodotto dai progettisti di ASIC per sintetizzarlo su FPGA.

La categoria più recente ad adottare SystemVerilog è stata quella dei progettisti di FPGA. Tuttavia, i primi a impiegare questo linguaggio sono stati frenati dalla carenza di un supporto completo per il tool attraverso l'intero flusso progettuale, il che si evidenzia soprattutto nell'area della sintesi di FPGA.

La buona notizia è che oggigiorno sono disponibili tool per SystemVerilog in grado di supportare il flusso di progetto per FPGA in tutte le sue fasi.

Le energie recentemente profuse da alcuni vendor EDA hanno migliorato di molto il supporto a SystemVerilog, permettendo a coloro che già lo utilizzavano da tempo di utilizzarne tutte le caratteristiche nella progettazione di FPGA. Oltre a essere in grado di simulare un progetto scritto in SystemVerilog, ora è possibile sintetizzarlo su una qualunque architettura target su FPGA con risultati di tutto rispetto. Un numero crescente di aziende leader nel settore stanno iniziando ad adottare SystemVerilog nelle proprie attività di progettazione FPGA.

Selex Communications
readerservice.it n. 41