Modelli computazionali in SystemC

Mariano Severi

Alcune considerazioni sulle modalità di rappresentazione in linguaggio SystemC di alcuni dei più diffusi modelli computazionali

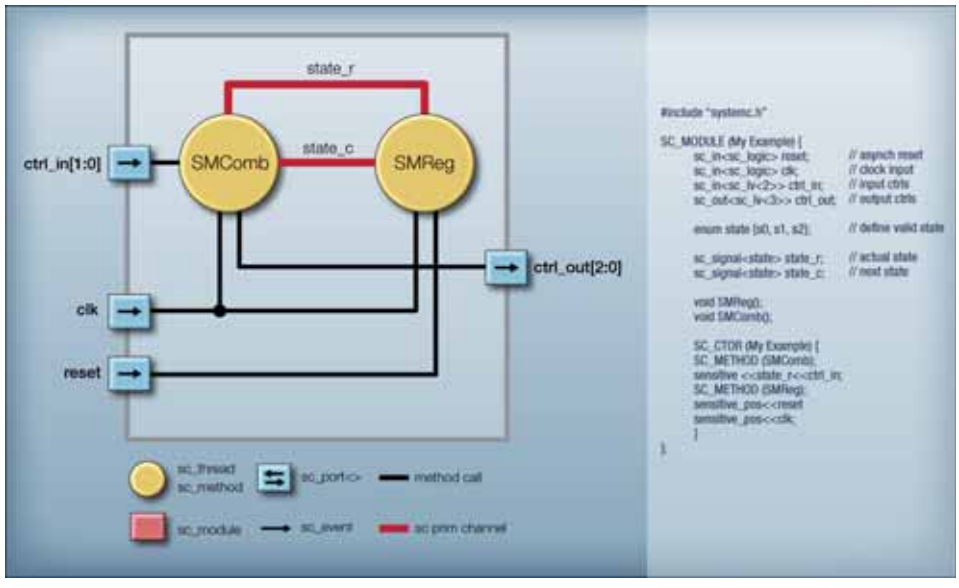

Fig. 1 - Esempio di descrizione RTL

La crescente complessità dei moderni dispositivi elettronici richiede la definizione di nuovi metodi di descrizione a livello sistema nelle fasi di sviluppo e verifica funzionale. Uno dei concetti principali in questo ambito è quello di 'modello computazionale' (MOC). Secondo una definizione non troppo formale ma piuttosto esemplificativa riportata in [5], un MOC è l'insieme:

- del modello di rappresentazione del tempo utilizzato e della filosofia di ordinamento degli eventi nella descrizione del sistema;
- dei metodi supportati per la comunicazione di processi concorrenti;
- delle regole di chiamata ed esecuzione dei processi.

I tradizionali linguaggi di descrizione HDL hanno ormai mostrato la loro inadeguatezza di fondo nella modellizzazione di sistemi complessi; questa passa anche attraverso l'incapacità di supportare adeguatamente MOC eterogenei. Una alternativa tra le più interessanti che sta affermandosi nel settore negli ultimi anni è rappresentata dal SystemC, una libreria di classi in ambiente C++ per la descrizione hardware. La struttura del kernel di simulazione basato sulla sincronizzazione mediante eventi, il supporto di un paradigma di tipo evaluate-update e la possibilità di definire interfacce, porte e canali custom rendono il nuovo linguaggio ideale per descrizioni a diversi livelli di astrazione. Il presente articolo riporta in particolare una breve panoramica che mostra come il linguaggio SystemC sia in grado di supportare alcuni dei principali modelli computazionali, tra cui descrizioni RTL (Register Transfer Level), reti KPN (Kahn Process Network) e modelli TLM (Transaction Level Model). La descrizione non è certamente esaustiva ma tende piuttosto a fornire alcuni spunti di approfondimento ed esempi. L'articolo presuppone una conoscenza elementare del linguaggio SystemC; una breve introduzione è riportata di seguito. Trattazioni più approfondite si trovano nella bibliografia citata come riferimento.

Una breve introduzione al linguaggio

Il SystemC è una libreria di classi in ambiente C++ che rende disponibili strutture per la modellizzazione di architetture concorrenti e la descrizione di sistemi complessi.

Gli elementi fondamentali di una descrizione in linguaggio SystemC sono moduli, processi, eventi, interfacce, porte e canali.
I moduli, in particolare, sono oggetti della classe SC_MODULE; essi costituiscono le unità funzionali minime, corrispondenti ai concetti di entità e architettura del VHDL.
Le interfacce sono classi astratte che ereditano da sc_interface e dichiarano le funzioni virtuali che rappresentano le operazioni eseguibili. I canali implementano tali metodi definendo il protocollo di accesso ai dati. Sono di tipo primitivo o gerarchico, ereditando dalla classi sc_prim_channel e sc_channel rispettivamente. I canali primitivi supportano un metodo di accesso ai dati di tipo request-update per modellizzare le architetture concorrenti e assicurare la coerenza dei dati; lo schema di simulazione che si ottiene è del tutto equivalente a quello del VHDL basato su delta-cycle. I canali gerarchici sono invece dei moduli che implementano una o più interfacce. Le porte, infine, sono oggetti che derivano direttamente od indirettamente dalla classe astratta sc_port_base e consentono l'accesso ai canali dall'esterno del modulo in cui sono contenuti; il SystemC implementa uno schema IMC (Interface Method Call) che consente di supportare in uno stesso framework descrizioni del sistema a diversi livelli di astrazione.
I processi sono funzioni membro di un modulo; sono utilizzati, ad esempio, per descrivere la logica del circuito. Nel costruttore del modulo -dichiarato mediante la macro SC_CTOR- sono registrati come metodi (SC_METHOD) o thread (SC_THREAD). Gli eventi sono oggetti della classe sc_event; i processi vengono richiamati dal kernel di simulazione quando è attivato un evento al quale sono stati dichiarati sensibili (sensitivity list). Mentre i thread possono essere interrotti mediante opportuna istruzione wait(delay), i metodi devono necessariamente terminare prima di poter rilasciare il controllo al kernel.
Oltre a registrare i processi, il costruttore di un modulo ne definisce l'architettura, istanziando canali primitivi o gerarchici ed eventuali altri moduli componenti e definendo le connessioni tra questi.

Modello RTL

Nell'accezione più generale, le rappresentazioni RTL, come noto, sono dei modelli computazionali in cui la caratteristica del sistema è descritta come un insieme di operazioni più o meno complesse eseguite su operandi memorizzati in registri; in alcuni contesti si sottintende, inoltre, che il modello sia sintetizzabile. Presso l'OSCI (Open SystemC Iniziative) - l'organizzazione indipendente no-profit che si occupa di promuovere il SystemC - è attivo un gruppo di lavoro che si occupa proprio della standardizzazione del sottoinsieme del linguaggio supportato dai tool di sintesi. In una descrizione RTL, l'evoluzione del sistema è dettata da un segnale di clock; in corrispondenza delle transizioni di tale segnale il contenuto dei registri viene modificato. L'insieme dei registri, della logica combinatoria che descrive le operazioni e delle connessioni tra questi è definito data-path. La descrizione RTL è accurata al livello di singolo ciclo di clock - nel senso che è sempre possibile definire lo stato del sistema in corrispondenza di ogni fronte attivo del clock - e di pin - nel senso che le porte di ingresso/uscita del modulo corrispondono esattamente ai segnali del circuito.
Ritardi di propagazione legati al trasferimento dei dati sono ignorati; la coerenza della descrizione nelle architetture concorrenti è basata su un meccanismo di aggiornamento delle informazioni scandito da delta-cycle. In linguaggio VHDL, ad esempio, il data-path è contenuto nell'architettura del componente mentre le porte di ingresso/uscita sono dichiarate nell'entità; la descrizione si basa sui concetti di signal per la rappresentazione delle informazioni e di process per l'astrazione delle operazioni eseguite su queste.
Il linguaggio SystemC segue uno schema piuttosto simile. I segnali sono rappresentati mediante canali primitivi di tipo sc_signal<T> che implementano l'interfaccia sc_signal_inout_if<T> che, a sua volta, eredita dalla classe sc_signal_in_if<T>. Tali interfacce definiscono i metodi astratti di accesso in lettura e scrittura al canale; T è il tipo di dato associato al segnale tra quelli supportati in descrizioni RTL come specifi-

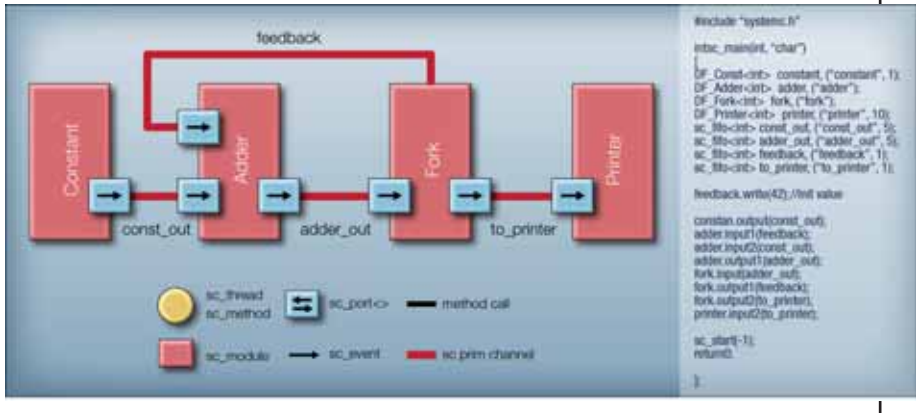

Fig. 2 - Esempio di descrizione Data-Flow

cato di seguito. Le porte per la connessione verso l'esterno dei moduli sono classi di tipo sc_in<T>, sc_out<T> e sc_inout<T> che derivano direttamente dalle interfacce sopra indicate.
Il data-path è descritto nel costruttore del modulo mediante processi SC_METHOD; la sensitivity list del processo è di tipo statico. Sono previsti diversi tipi di dati per la modellizzazione hardware.
Tra quelli propri del linguaggio SystemC quelli supportati in descrizioni RTL sono sc_bit, sc_bv, sc_logic, sc_lv, sc_int, sc_uint, sc_bigint, sc_biguint, sc_fixed, sc_unfixed; i tipi nativi C++ utilizzabili sono invece bool, int, unsigned int, long, unsigned long, signed char, unsigned char, short, unsigned short, enum e struct.
I tipi enumerativi, in particolare, sono utilizzati per la definizione degli stati di una macchina a stati finiti.
In questo caso, lo schema a due processi tipico della descrizione HDL può essere riprodotto registrando nel costruttore del modulo due metodi di cui uno, sensibile allo stato attuale e agli ingressi, per la descrizione delle equazione di prossimo stato e prossima uscita e uno, invece, sensibile al segnale di clock, per inferire i registri. In figura 1 è mostrato lo schema di una generica descrizione RTL basata su macchina a stati finiti; il codice sorgente di più alto livello è riportato a lato.

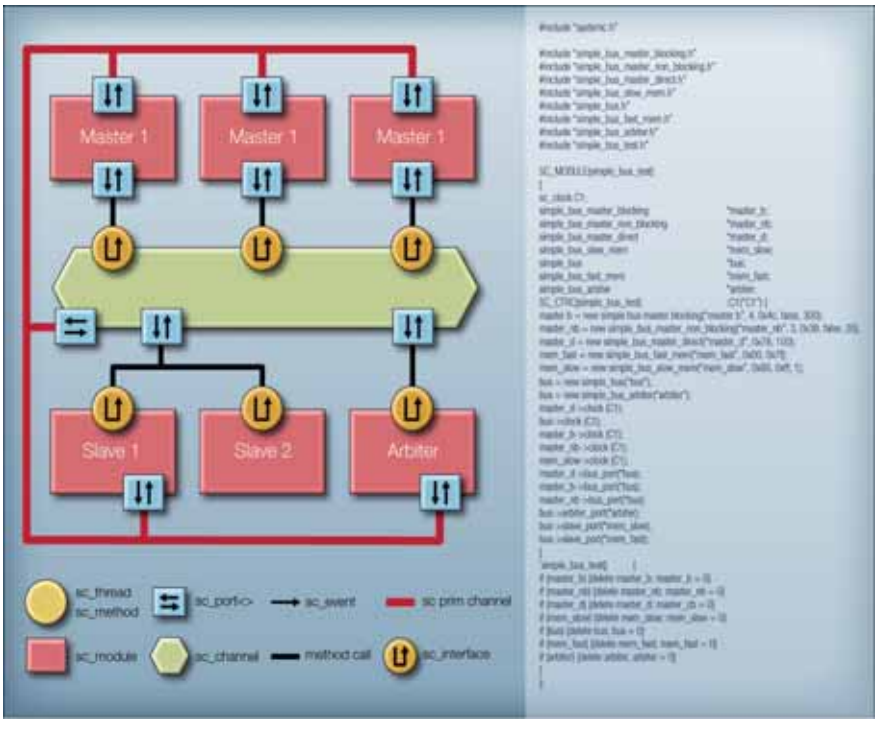

Fig. 3 - Esempio di descrizione TLM

Modello KPN

Sebbene esistano definizioni piuttosto formali, in maniera semplificata si può considerare come KPN un qualsiasi insieme di processi concorrenti e sequenziali che elaborino un flusso di informazioni in ingresso secondo uno schema produttore/consumatore. I processi comunicano mediante code first-in-first-out di lunghezza infinita; i metodi di accesso in scrittura sono di tipo non-blocking, gli accessi in lettura di tipo blocking. Le reti KPN rappresentano un'architettura spesso utilizzata per la descrizione di applicazioni di digital signal processing, in sistemi embedded o in ambito multimediale. Il modello non presuppone alcuna idea di tempo; gli eventi dipendono tra loro su base causale, il che, automaticamente, implica l'ordinamento con cui si verificano. Una variante delle reti KPN sono i modelli data-flow, rappresentabili mediante grafi diretti; i nodi corrispondono ai processi mentre gli archi definiscono la sequenza ordinata di eventi. Le reti SDF (Static Data Flow), come caso particolare, prevedono a ogni nodo un numero finito di dati (token) in ingresso/uscita; tale restrizione consente più agevolmente la verifica funzionale del modello.
In linguaggio SystemC un generico modello KPN/data-flow può essere descritto mediante processi di tipo SC_THREAD e canali di tipo sc_fifo<T>. Tale canale implementa infatti le interfacce sc_fifo_in<T> e sc_fifo_out<T> per la descrizione di una memoria di tipo first-in-first-out; T, come in precedenza, rappresenta il tipo di dato.
Tra i metodi supportati dal canale vi sono write() e read() che consentono l'accesso in scrittura/lettura alla memoria, num_free() e num_available() che ritornano, rispettivamente, il numero di locazioni libere e occupate. Tuttavia, diversamente dall'assunzione generale

delle reti KPN, entrambi i metodi di accesso sono di tipo blocking. Il controllo di flusso dei dati attraverso il canale avviene sulla base dei metodi data_read_event() e data_written_event() che ritornano riferimenti a eventi la cui notifica segnala accessi in lettura e scrittura; i processi di lettura e scrittura restano quindi in attesa di tali eventi in condizioni di fifo vuota o piena, rispettivamente.
La definizione di metodi blocking impone, tuttavia, l'utilizzo di SC_THREAD per la descrizione dei processi della rete; i metodi blocking, infatti, possono arrestare il processo nel quale sono definiti. Come sottolineato in precedenza, invece, i processi SC_METHOD devono terminare prima di restituire il controllo al kernel di simulazione. Inoltre, nelle architetture che richiedono una inizializzazione del canale, questa deve essere chiamata prima di iniziare la simulazione della rete. In caso contrario, la natura blocking dei metodi di accesso ai dati determinerebbe una condizione di stallo irreversibile; l'esempio riportato di seguito chiarisce questo aspetto.
I canali sc_fifo<T> descrivono memorie di lunghezza finita che deve essere dichiarata quando il canale è istanziato. Nel caso di rete SDF, noto il numero massimo di token al nodo, tale limitazione non pone evidentemente alcun problema; nel caso, invece, di un flusso infinito la coerenza del modello è assicurata dalla natura blocking del metodi di scrittura che previene la perdita di dati. Il modello data-flow considerato è evidentemente untimed. Se non si verificano errori, termina quando sono stati consumati tutti i dati e prodotti tutti i risultati; a questo punto infatti non ci sono più eventi in coda e quindi processi da attivare. Modelli timed equivalenti possono facilmente essere derivati introducendo delle istruzione wait() nei thread che descrivono i processi della rete; in questo caso la simulazione deve essere eseguita definendo un intervallo temporale opportuno.
La figura 2 mostra come semplice esempio di rete KPN il modello di un accumulatore; il relativo codice sorgente per la descrizione in ambiente SystemC è riportato a lato. Per brevità è omessa la descrizione dei singoli moduli; i dettagli si trovano in [5]. Constant è un generatore di dato costante, adder un addizionatore, fork replica sulle porte di uscita il dato in ingresso, printer invia su schermo il risultato del processo. Si noti l'inizializzazione del canale feedback che si rende necessaria per evitare lo stallo della simulazione; il modulo adder infatti legge dai canali in ingresso prima di scrivere il canale in uscita.

Modello TLM

I metodi TLM sono metodi di descrizione a livello sistema basati su transazioni, ovvero rappresentazioni astratte dei modi di comunicazione. Queste tendono a enfatizzare le funzionalità del flusso di informazioni tra i diversi processi piuttosto che i dettagli implementativi del protocollo; il controllo e i dati sono pas-

sati in blocco tra i diversi processi in corrispondenza di ben definiti punti di sincronizzazione. Initiator e target –secondo una nomenclatura piuttosto diffusa anche in altri ambiti– sono i processi dove inizia e termina una transazione.
In funzione del diverso modo di sincronizzazione dei processi, si distingue tra descrizioni untimed, loosely-timed e approximately timed. I modelli untimed, in particolare, non presuppongono alcuna nozione di tempo; ogni processo si svolge fino a che non ritorna il controllo al kernel di simulazione in corrispondenza di punti di sincronizzazione definiti esplicitamente dall'utente. In linguaggio SystemC, ad esempio, questo avviene mediante chiamata alla funzione wait() che sospende l'esecuzione del thread. Nei modelli loosely-timed (chiamati anche Programmers View), invece, oltre a tale meccanismo, è prevista la sincronizzazione dei processi sulla base di intervalli regolari di tempo; il processo viene sospeso esplicitamente o al termine dell'intervallo. Nelle descrizioni approximately-timed (chiamate anche Programmers View with Time), infine, le interazioni tra i processi sono annotate con ritardo specifico; in linguaggio SystemC le funzioni wait(delay) e notify(delay) servono a tale scopo.
Tipicamente è sufficiente annotare i ritardi come latenza di istruzione della transazione al nodo initiator o ritardo nell'accettazione di questa al target.
Le descrizioni TLM risultano in genere piuttosto semplici, eliminando i dettagli implementativi, sebbene accurate dal punto di vista funzionale. Tra le applicazioni principali figurano la realizzazione di modelli per l'analisi e la verifica dell'architettura hardware, lo sviluppo delle architetture e le analisi delle prestazioni dei componenti software.
La struttura del linguaggio SystemC basata su canali e interfacce rende piuttosto agevole la descrizione di modelli TLM. Le transazioni infatti possono essere descritte mediante chiamate a funzioni membro dell'interfaccia associata al canale che connette initiator e target; i dati possono essere rappresentati come strutture e passati semplicemente mediante puntatori. L'utilizzo, inoltre, dei tipi nativi in linguaggio C++, la descrizione di processi come SC_METHOD laddove non siano impiegati interfacce di tipo blocking, la definizione di sensitivity list in maniera dinamica consentono di ottenere descrizioni con tempi di simulazione piuttosto brevi. Nella distribuzione standard del linguaggio SystemC è incluso un semplice esempio di descrizione TLM di un protocollo di comunicazione per bus in applicazioni System-On-Chip. In figura 3 è mostrato uno schema a blocchi del sistema descritto unitamente al codice sorgente di livello più alto; una discussione piuttosto dettagliata del modello è riportata in [5]. Il bus è descritto come un canale gerarchico che implementa le interfacce blocking, non-blocking e dirette utilizzate dai moduli initiator (master); presenta inoltre due porte per la connessione dei dispositivi slave e dell'arbiter. I moduli slave implementano quindi le interfacce target.
Un segnale di clock è distribuito ai moduli che lo richiedano; è previsto un meccanismo di sincronizzazione a due-fasi con processi sensibili ad uno o l'altro dei fronti attivi di tale segnale.
I modelli TLM hanno trovato rapida diffusione, come è facile immaginare visti i vantaggi offerti. Presso l'OSCI è attivo un working group per la definizione e lo sviluppo di metodologie e librerie add-on per la modellizzazione TLM; l'obiettivo è la definizione di API e strutture dati che favoriscano la standardizzazione e l'interoperabilità dei modelli. Recentemente è stata rilasciata in versione draft la nuova release 2.0 della specifica. Definisce un insieme di interfacce di tipo blocking e non-blocking per il trasporto delle transazioni tra i processi; sono supportati modelli PV e PVT. È inoltre prevista una interfaccia DMI (Direct Memory Interface) che consente a un initiator di accedere direttamente mediante puntatore a un'area di memoria del processo target. La presentazione dello standard esula gli scopi del presente articolo; il riferimento allo specifica completa dove sono indicati tutti i dettagli è riportato in [3].
L'articolo ha introdotto in linea generale le modalità di rappresentazione in linguaggio SystemC di alcuni dei più diffusi modelli computazionali. La rappresentazione più indicata per un sistema dipende evidentemente dal tipo di applicazione. Allo stesso tempo, però, la possibilità di supportare diverse rappresentazioni in uno stesso ambiente consente la creazione di modelli flessibili e riutizzabili. L'obiettivo, al solito, resta quello di ridurre il time-to-market, favorendo il riutilizzo e semplificando le fasi di analisi architetturale e verifica funzionale; in questo senso il linguaggio SystemC sembra rappresentare oggi uno degli ambienti di sviluppo più adeguati.

Riferimenti

[1] www.systemc.org
[2] IEEE 1666 - 2005(tm) Standard SystemC Language Reference Manual (LRM) http://standards.ieee.org/getieee/1666/index.html.
[3] OSCI TLM2 user manual November, 2007
[4] Black,C.D. e Donovan,J. "SystemC – From the ground up" Kluwer Academic Publishers 2004
[5] Grötcker,T et alii "System Design with SystemC" – Kluwer Academic Publishers 2004
[6] Ghenassia, F "Transaction-Level Modelling with SystemC" Springer, 2005