Realizzazione mediante FPGA di un convertitore di velocità di campionamento

Philipp Jacobsohn
Field application engineer
Synplicity Deutschland
Pierluigi Lo Muzio
Specialista europeo DSP
Synplicity

Come implementare in maniera efficiente algoritmi audio mediante componenti logici programmabili o circuiti ASIC

Gli odierni FPGA, anche quelli a basso costo, mettono a disposizione una potenza di elaborazione di gran lunga superiore rispetto a quella fornita dai DSP (Digital Signal Processor). Questi FPGA dispongono di moltiplicatori dedicati e persino di blocchi DSP (MAC) che consentono di elaborare segnali a velocità di clock superiori a 250 MHz. Finora, comunque, queste potenzialità non sono state sfruttate per l'elaborazione del segnale audio. L'implementazione di un algoritmo audio che opera nel range dei kHz richiederebbe esattamente le stesse risorse necessarie per l'elaborazione di segnali a frequenze molto maggiori (nel range delle centinaia di MHz). Di conseguenza, componenti logici programmabili quali CPLD o FPGA vengono raramente utilizzati per elaborare segnali a bassa frequenza. Questo perché l'elaborazione parallela di operazioni matematiche in hardware non fornisce alcun vantaggio rispetto una realizzazione basata sui classici DSP. Le applicazioni audio sono caratterizzate da un numero così elevato di moltiplicazioni da richiedere dispositivi FPGA di grosso taglio. Finora, dunque, nelle applicazioni dove le frequenze di campionamento sono di valore ridotto, l'impiego di DSP risulta più efficace in termini sia economici sia di supporto software disponibile. La situazione ora è radicalmente cambiata grazie a Synplify DSP, un tool di Synplicity che permette di mappare in maniera efficiente sugli FPGA algoritmi caratterizzati da un gran numero di moltiplicazioni e bassa velocità di campionamento. Il tool è basato sul software Matlab/Simulink di Mathworks (Fig. 1). L'algoritmo è definito utilizzando un particolare blockset (insieme di blocchi funzionali) e/o una descrizione nel linguaggio di scripting "m" proprietario e successivamente convertito nel linguaggio di descrizione dell'hardware RTL. Il blockset permette di realizzare sistemi a singola velocità di campionamento (single rate) e a più velocità di campionamento (multi-rate).

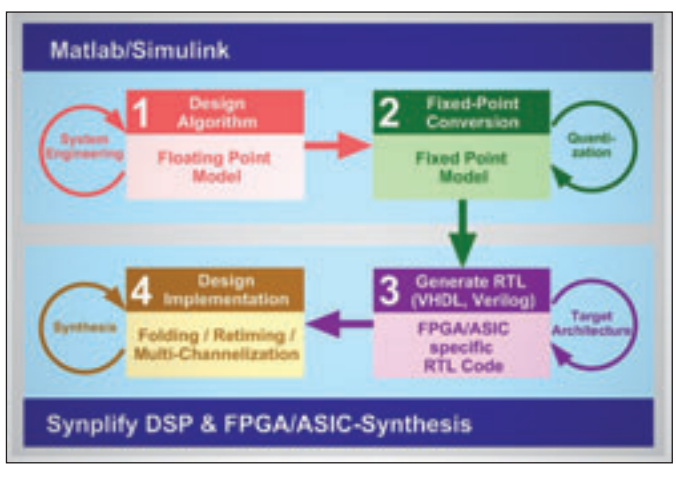

Fig. 1 – Il modello è implementato, quantificato e verificato in Matlab/Simulink. Synplify DSP è impiegato per convertire il modello in codice RTL. Il codice può essere ottimizzato in termini di spazio o di velocità

Esso non solo genera il codice VHDL e Verilog, ma è anche in grado di gestire la quantizzazione e si connette ai blockset dell'ambiente di sviluppo SimulinK richiesto per la simulazione.

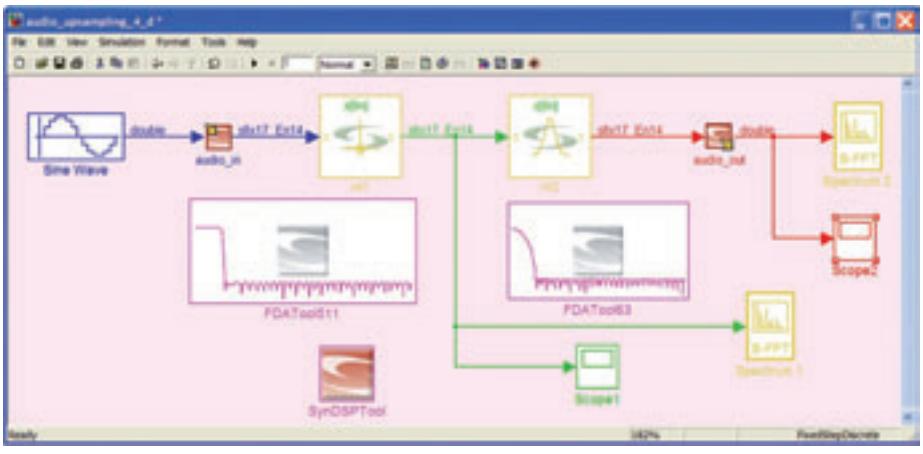

Fig. 2 – Per realizzare il convertitore di velocità di campionamento vengono usati i moduli del blockset Synplify DSP e il tool FDA di Simulink. Gli elementi del blockset di Simulino sono impiegati anche per la verifica

Un esempio: conversione della velocità di campionamento

Come esempio pratico si farà riferimento a un convertitore di velocità di campionamento per frequenze audio (Fig. 2). Un dispositivo di questo tipo può essere utilizzato per convertire qualsiasi velocità di clock a bassa frequenza in qualunque altra frequenza. Questi convertitori sono richiesti per la gestione di segnali caratterizzati da velocità di campionamento differenti. Per esempio, i segnali televisivi in Europa e negli Stati Uniti utilizzando diverse velocità di quadro (fps) per l'elaborazione delle immagini. Nel caso sia richiesta la conversione da un formato all'altro, non è sufficiente la semplice riproduzione dei dati della sorgente con la nuova velocità di campionamento. Sia le sequenze video sia i dati audio associati non risulteranno sincronizzati. Questo è il motivo per cui è necessaria la conversione della velocità di campionamento.

Quando si elaborano segnali audio, vengono utilizzate frequenze di campionamento di 44,1, 48, 96 e 192 kHz. Mentre la velocità di campionamento di 44,1 kHz viene usata per i CD, nelle applicazioni professionali si impiega una velocità di campionamento di 48 kHz per la registrazione dei segnali su nastri audio digitali (DAT – Digital Audio Tape). Durante la conversione, bisogna porre attenzione nel mantenere l'integrità dei segnali nell'intervallo tra 0 e 20 kHz. Per limitare il degrado qualitativo, è necessario minimizzare le modifiche alle informazioni contenute nel segnale.

L'implementazione del convertitore della velocità di campionamento per frequenze audio in un FPGA solleva due tipi di problemi.

A livello di algoritmo:
- il rapporto segnale/rumore deve essere il più elevato possibile;
- le variazioni a livello di informazione trasportate dal segnale originale devono essere minime;
- la descrizione dell'algoritmo deve essere efficiente. Il carico di risorse a livello dell'FPGA è funzione della qualità della descrizione;
- quantizzazione.

A livello di realizzazione:
- l'implementazione dell'algoritmo essere corretta dal punto di vista logico;
- requisiti in termini di risorse computazionali nell'FPGA;
- ottimizzazione in termini di velocità;
- latenza.

Questo tipo di conversione richiede un'elevata velocità di clock perché l'implementazione necessita (e dipende) dal fatto che il sovracampionamento risulti adeguato. La differenza tra la frequenza del sistema FPGA e le frequenze che devono essere convertite devono quindi risultare elevate. Per segnali audio di qualità CD, il rapporto tra segnale e rumore deve essere pari almeno a 100 dB. Nelle applicazioni di tipo professionale, questo rapporto può essere superiore anche a 120 dB. Altri segnali a bassa frequenza (ad esempio quelli relativi ad algoritmi di controllo) sono meno "esigenti" in termine di qualità del segnale rispetto ai segnali audio.

Uno sguardo all'algoritmo

Per consentire l'elaborazione è necessario convertire i segnali caratterizzati da differenti velocità di campionamento. I segnali audio sono caratterizzati da velocità di campionamento di 44,1 kHz (nel caso di CD) o 48 kHz (DAT).

Per convertire le frequenze (ricampionamento asincrono) si fa ricorso a filtri FIR polifase. L'algoritmo di conversione prevede due fasi (Fig. 3). Nella prima si procede al sovracampionamento delle frequenze mentre la seconda (interpolazione lineare) è necessaria per generare una frequenza diversa a partire da una data frequenza. Le due frequenze sono

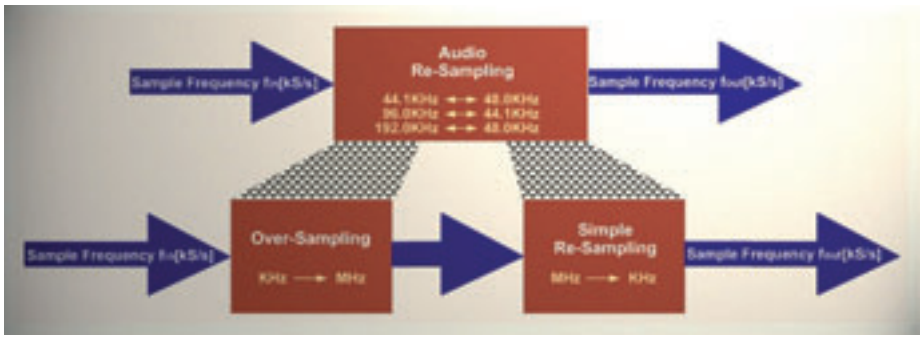

Fig. 3 – Il convertitore di velocità di campionamento è realizzato in due fasi per migliorare l'efficienza.
Fase 1 – Sovracampionamento
Fase 2 – Interpolazione lineare

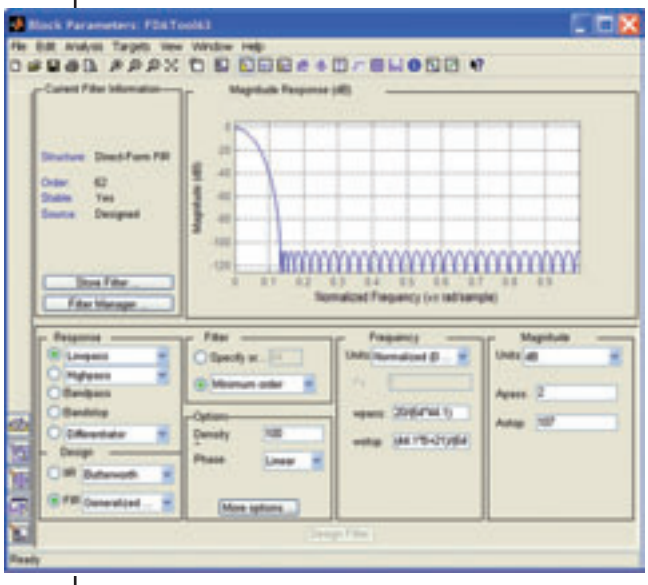

Fig. 4 – I filtri vengono realizzati mediante il tool FDA (Filter Design and Analysis) di Simulink

asincrone l'una rispetto all'altra. Il ricampionamento del segnale in una sola fase richiede molte risorse in quanto il filtro risulta più complesso. Una realizzazione di questo tipo richiederebbe parecchi milioni di moltiplicazioni. Si tratta quindi di una descrizione inefficiente che deve essere evitata. Se nella seconda fase viene prevista un'interpolazione lineare, le strutture risultanti sono decisamente più semplici.

La prima fase, ovvero il sovracampionamento, deve essere descritta nel modo più efficiente possibile. Si tratta del solo modo per ottenere una realizzazione mediante FPGA ottimizzata dal punto di vista delle risorse. Il numero di operazioni di calcolo necessarie può essere ridotto drasticamente se questa parte del circuito viene implementata sfruttando più stadi in cascata invece che in una singola fase di sovracampionamento. Il numero di moltiplicatori richiesti aumenta in funzione dell'ordine del filtro. Ogni stadio (tap) del filtro comporta l'utilizzo di un blocco DSP o di un moltiplicatore 18 x 18. Quando si pongono in cascata gli stadi di sovracampionamento, l'ordine dei filtri diminuisce. In teoria, una realizzazione ottimale del filtro, col minor numero di operazioni, prevede il maggior numero possibile di stadi individuali. Nella letteratura tecnica esiste un'ampia trattazione delle deduzioni matematiche che permettono di ridurre le operazioni di calcolo. In questa sede è sufficiente sapere che, benché, dal punto di vista computazionale, sia realmente necessario frammentare il sovracampionamento in una cascata di tanti stadi di sovracampionamento, il loro numero non deve essere esagerato ai fini di una efficiente implementazione sull'FPGA. Infatti, nel caso sia presente un numero elevato di stadi in cascata, potrebbero essere richieste maggiori risorse dell'FPGA in quanto è necessario ricorrere a circuiti logici aggiuntivi per la realizzazione. Se come architettura target è stato scelto un FPGA di basso costo, per i singoli passi di sovracampionamento il numero ottimale è risultato essere due.

Il circuito complessivo risulta composto da due filtri relativamente semplici e da una interpolazione lineare: una struttura di questo tipo può essere implementata in maniera efficiente in un FPGA.

Fase di realizzazione

Il circuito viene realizzato all'interno dell'ambiente Simulink utilizzando il blockset Synplify DSP (un insieme di blocchi funzionali comunemente impiegati in progetti DSP) e il tool per l'analisi e il progetto di filtri (FDA – Filter Design and Analysis) di Simulink (Fig. 4). Questo tool consente di generare e verificare qualsiasi tipo di filtri FIR e IIR. Esso fa parte del Signal Processing Toolbox di Simulink utilizzato da Synplify DSP per realizzare le strutture del filtro. Tutti i componenti circuitali del blockset Synplify DSP o del tool FDA definiti tra una porta di ingresso (PortIN) e di uscita (PortOUT) generano un codice Verilog o VHDL. Gli elementi FFT e SCOPE dei blockset Simulink vengono utilizzati per l'analisi di spettro e la verifica della risposta dinamica. Tali blocchi vengono impiegati esclusivamente per la verifica funzionale, compresi gli effetti della conversione da virgola mobile a virgola fissa (quantizzazione) e non sono realizzati in hardware. Per l'implementazione della prima parte dell'algoritmo sono previsti due filtri FIR, caratterizzati il primo da 512 stadi (tap) e il secondo da 64 stadi. Il codice RTL per il sovracampionamento conterrebbe quindi un totale di 576 moltiplicazioni, numero che spiega il motivo per cui l'uso di FPGA non appare particolarmente adatto. Solamente i componenti logici programmabili più complessi sono in grado di fornire sufficienti risorse, in termini di moltiplicatori, per realizzare il circuito: Fpga di questo tipo (come ad esempio EP3SE110 della serie Stratix-3 di Altera con 896 moltiplicatori 18 x 18 e XC5VSX95TXC della linea Virtex-5 di Xilinx con 640 moltiplicatori 25 x 18) hanno un costo parecchio elevato. Tutte le moltiplicazioni che non vengono mappate su strutture hardware dedicate (blocchi DSP) devono essere realizzate a partire da risorse logiche (LUT, registri). Di conseguenza, a fronte di un numero elevato di componenti, si ottiene una velocità di clock massima di valore ridotto.

Invece l'opzione di folding di Synplify DSP consente di ridurre drasticamente il numero di moltiplicatori utilizzati (Fig. 5). Questa ottimizzazione risulta particolarmente vantaggiosa per circuiti operanti a basse frequenze di campionamento. L'idea di base è particolarmente semplice. Di solito si adopera un moltiplicatore per eseguire una singola operazione anche quando la frequenza di campionamento è nel range dei kHz. Gli FPGA possono operare con velocità che possono arrivare anche alle centinaia di MHz. Nel caso il moltiplicatore hardware operi alla frequenza di sistema dell'FPGA, le moltiplicazioni possono essere eseguite sequenzialmente sfruttando

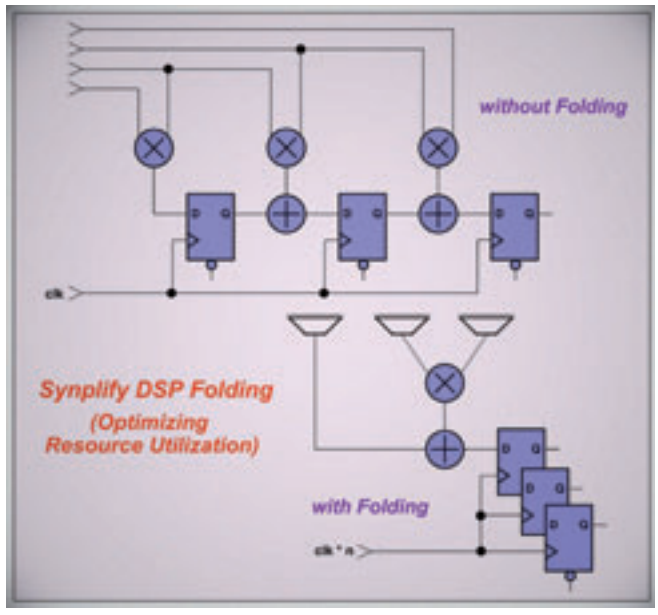

Fig. 5 – È possibile ridurre drasticamente le risorse dell'FPGA mediante l'opzione di folding

un processo di multiplex a divisione di tempo. Un esempio pratico serve a chiarire il concetto. Se la frequenza di campionamento del circuito è di 3 MHz e l'FPGA può operare a una frequenza massima di 120 MHz, ciascun moltiplicatore hardware può eseguire 40 operazioni di calcolo nel caso venga utilizzata per la moltiplicazione la frequenza del sistema. L'hardware richiesto viene di conseguenza ridotto di un fattore pari a 40. I segnali audio non vengono elaborati utilizzando frequenze di campionamento nel range dei MHz. Come accennato in precedenza, le frequenze di campionamento sono pari a 44,1, 48, 96 o 192 kHz. Ciò significa che un convertitore di velocità di campionamento (o qualsiasi altro circuito che utilizza frequenze di campionamento di valore ridotto) può essere "ripiegato" (folded) in modo tale da ridurre al minimo il numero di moltiplicatori hardware richiesti. In questo modo risulta possibile la realizzazione anche sui più piccoli FPGA low cost, che rappresentano così una reale alternativa ai DSP.
Naturalmente risulta anche possibile trasferire algoritmi particolarmente onerosi in termini computazionali da un DSP a un FPGA, riducendo così il carico sul processore. Poiché la caratteristica di "folding" di Synplify DSP è in grado di supportare sistemi operanti a differenti velocità (multi-rate), anche in questo caso risulta possibile ridurre il numero di moltiplicatori richiesti rispetto al caso di un sistema con una sola frequenza di campionamento. L'operazione di sovracampionamento viene eseguita mediante due filtri FIR che operano a differenti frequenze di campionamento. Il filtro che opera alla frequenza di campionamento maggiore viene "ripiegato" utilizzando un fattore di "folding" specificato dall'utilizzatore. L'altro filtro, invece, viene "ripiegato" utilizzando un fattore corrispondentemente più elevato. Questo fattore si ottiene moltiplicando la differenza tra la frequenza di campionamento dei due filtri per il fattore di "folding" definito dall'utente. Per esempio, se la frequenza di campionamento di un filtro è superiore di un fattore pari a 8 rispetto a quella dell'altro filtro, il filtro più veloce viene "ripiegato" di un fattore pari a 8 mentre quello più lento di un fattore pari a 64.
In questo modo risulta persino possibile generare circuiti ottimizzati in termini di spazio anche quando ci sono delle parti che operano a velocità di campionamento molto alte, che solitamente non possono essere sottoposte all'operazione di "folding". Per esempio, se la velocità di campionamento di un sistema è pari a 200 MHz e il fattore di "folding" utilizzato è pari a 2, la frequenza del sistema aumenta a 400 MHz. Una velocità, questa, difficile da raggiungere anche da FPGA con prestazioni particolarmente spinte.
L'alternativa è definire un fattore di "folding" pari a 1. Quindi, quei componenti del circuito che operano alle velocità di campionamento più elevate non subiscono il processo di "ripiegamento".
Invece, tutti i componenti del sistema multi-rate, operanti a frequenze di campionamento inferiori, possono trarre numerosi vantaggi dalla funzione di "folding"e dalla possibilità di generare un'implementazione ottimizzata in termini di ingombri. L'utente deve solamente definire il fattore di "folding"

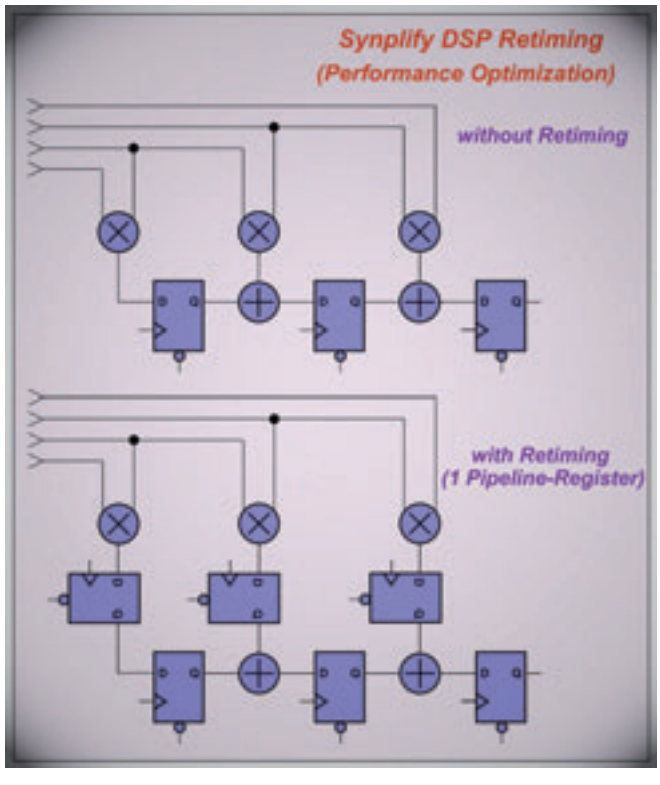

Fig. 6 – Utilizzando la caratteristiche di retiming, l'utente può definire la massima latenza consentita per il circuito. Synplify DSP aggiunge automaticamente stadi di pipeline finché non viene raggiunta la frequenza desiderata

per il sistema nel suo complesso: il "folding", poi, si propaga automaticamente attraverso tutte le frequenze di campionamento.
La caratteristica di folding può essere abbinata con un'altra funzione di ottimizzazione che è la ritemporizzazione (retiming). Se un sistema non soddisfa i requisiti di massima frequenza di lavoro, è possibile aggiungere stadi di pipeline finché non si raggiunge la velocità desiderata (Fig. 6). Ciò risulta importante nel caso di circuiti caratterizzati da un fattore di "folding" elevato che devono quindi operare a una velocità di sistema molto alta. Naturalmente la funzione di retiming può essere utilizzata anche senza il "folding" al fine di poter raggiungere la massima frequenza di lavoro del circuito sull' FPGA. L'aggiunta di stadi di pipeline permette di ridurre il numero di porte combinatorie tra due registri (livelli logici) fino a quando i ritardi imputabili a queste porte non hanno più alcuna influenza sulla velocità di clock. Nel momento in cui genera il codice RTL, SynplifyDSP esegue un'analisi di temporizzazione che prende in considerazione la frequenza di campionamento desiderata, il fattore di "folding" e l'architettura dell'FPGA. Un circuito mappato su un FPGA ad alta velocità, come ad esempio Virtex-5 di Xilinx o Stratix-3 di Altera può essere ottimizzato utilizzando un numero inferiore di stadi di pipeline rispetto a quelle necessarie per un identico circuito implementato su un FPGA low cost (ECP-2 di Lattice o Spartan-3 di Xilinx). Gli FPGA mettono a disposizione un gran numero di registri che possono venire sfruttati per questa ottimizzazione. A differenza di moltiplicatori o LUT, che possono esaurirsi rapidamente, i registri sono disponibili in gran quantità: di conseguenza, la velocità di clock può essere aumentata in maniera considerevole con uno sforzo minimo mediante i registri. Naturalmente, l'aggiunta di stadi di pipeline contribuisce a incrementare la latenza del sistema. Con l'introduzione di un fattore di ritemporizzazione pari a 8, per esempio, il risultato del calcolo

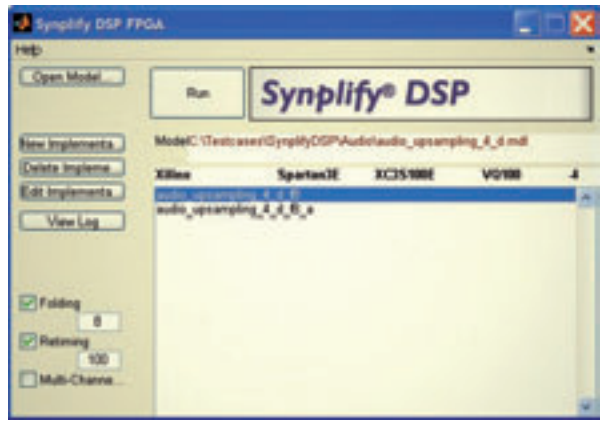

Fig. 7 – Utilizzando il software Synplify DSP è possibile convertire un modello Simulink o un codice M in una descrizione RTL generica (VHDL/Verilog). In uscita è anche disponibile un testbench RTL

apparirà sull'uscita dell'FPGA dopo 8 cicli di clock di sistema (e non cicli di frequenza di campionamento). Questo fatto deve essere tenuto in considerazione quando si integra un circuito nel sistema. Naturalmente è importante assicurare che le ottimizzazioni descritte in precedenza non abbiano effetto sul modello descritto in Matlab/Simulink. L'algoritmo viene implementato e verificato all'interno di Simulink. La verifica consente di validare l'algoritmo e rappresentare l'impatto degli effetti del fenomeno della quantizzazione. Il blockset Synplify DSP permette di effettuare la conversione da virgola mobile a virgola fissa utilizzando il troncamento (eliminazione dei bit non rilevanti), l'arrotondamento (in caso di underflow) o la saturazione (in caso di overflow).
Non appena la simulazione indica che l'algoritmo funziona nel modo desiderato, è possibile generare automaticamente il codice RTL. L'ottimizzazione del codice VHDL o Verilog può modificare la latenza, ma non il funzionamento del circuito.

Uno sguardo al tool

Synplify DSP è basato sul software Matlab/Simulink sviluppato da Mathworks. Un blockset fornisce una libreria di componenti standard che possono essere impiegati per realizzare algoritmi complessi. Oltre ai componenti standard quali somma, guadagno e ritardo, la libreria include numerose funzioni complesse quali filtri FIR o IIR e l'algoritmo Cordic. Tutte le risorse, comprese la FFT o la codifica/decodifica Reed-Solomon, possono essere liberamente parametrizzate. È anche possibile creare librerie definite dall'utente oppure integrare il codice VHDL o Verilog esistente in un modello Simulink. Synplicity mette a anche a disposizione, a titolo gratuito, un'ampia gamma di circuiti di riferimento per applicazioni quali elaborazione dell'immagine, SDR (Software Defined Radio) e audio. L'applicazione descritta in questo articolo, ovvero il convertitore di velocità di campionamento, utilizzabile per tutte le frequenza audio, è disponibile gratuitamente sotto forma di design di riferimento.
Il tool consente l'implementazione di sistemi operanti a velocità singola o multipla. Grazie all'utilizzo di tecniche di folding, di separazione in più canali (multichannelization) o ritemporizzazione è possibile ottimizzare il codice in termini di occupazione di spazio o di velocità. Il codice generato è sempre di tipo generico, non cifrato, che può essere sintetizzato mediante i più diffusi tool. Per ottenere i migliori risultati con gli FPGA, è consigliata l'adozione di Synplify Pro, in grado di supportare i componenti programmabili dei più importanti fornitori di questo settore – Actel, Altera, Lattice, QuickLogic e Xilinx. Inoltre, è anche disponibile una versione dell'ambiente di sviluppo espressamente concepita per i dispositivi ASIC.

Bibliografia

[1] Fliege, N.J.: Multi-rate Digital Signal Processing
[2] Marven, Craig; Ewers, Gillian: A simple Approach to Digital Signal Processing
[3] http://www.synplicity.com/literature/

Synplicity (Edaway)
readerservice.it n. 6