Architetture per la realizzazione di soluzioni di accelerazione hardware

Ian Ferguson
VP Advanced Products
QuickLogic

L'impiego di soluzioni hardware adatte, come ad esempio i componenti SoC della linea QuickMIPS di QuickLogic, permettono di accelerare le componenti di un algoritmo particolarmente critiche dal punto di vista delle prestazioni

In un gran numero di applicazioni embedded la necessità di dover elaborare dati a elevata velocità e garantire un elevato throughput a livello di sistema sono divenute così pressanti che l'implementazione di algoritmi esclusivamente via software utilizzando una singola CPU pone problemi di varia natura tra cui:
- costo e consumi di potenza: oltre ai problemi legati alla durata della batteria, particolarmente critici nel caso di piattaforme mobili, bisogna tener presente che per dissipare potenze più elevate è necessaria la presenza di dissipatori e altri dispositivi di raffreddamento;
- possibilità di fornire funzioni a valore aggiunto: se la CPU è completamente impegnata nella gestione di funzioni del sistema, i progettisti non possono aggiungere ulteriori funzionalità senza l'ausilio di ulteriori componenti.

Per risolvere questi problemi sono disponibili parecchie opzioni, brevemente descritte qui di seguito:
- utilizzo di un processore personalizzato: mediante la personalizzazione del set di istruzioni della CPU in funzione dell'applicazione considerata, è possibile migliorare sensibilmente l'efficienza di elaborazione dell'algoritmo. La disponibilità di tool di sviluppo per supportare questi core è cresciuta notevolmente negli ultimi anni. Tale strategia potrebbe "legare" un utente a una specifica implementazione che, a lungo andare, causerebbe problemi connessi all'impiego di software di tipo "legacy";
- implementazione hardware di alcuni elementi che richiedono prestazioni particolarmente spinte: tali elementi potrebbero essere realizzati tramite circuiti ASIC, ASIC strutturati, ASSP o FPGA;
- utilizzo di più core di Cpu: nel caso un'applicazione possa essere suddivisa in compiti ben definiti e in qualche misura indipendenti, è possibile migliorare il grado di efficienza con il quale un algoritmo può essere implementato. A

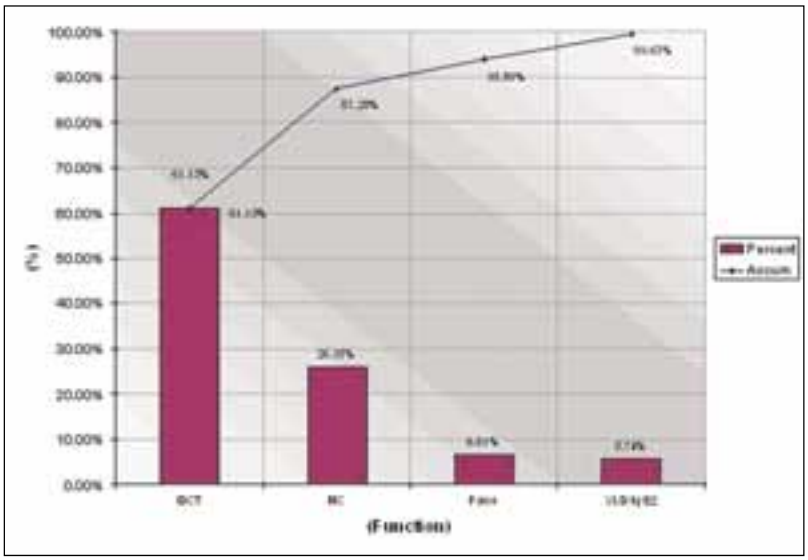

Fig. 1 - Istogramma del decodificatore software MPEG2

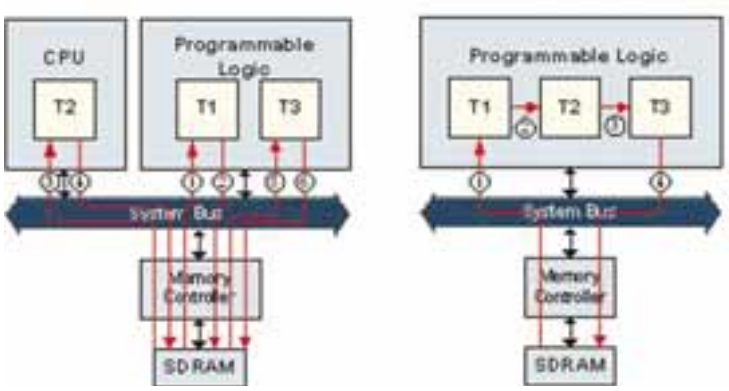

Fig. 2 - Rappresentazione del flusso di dati quando T1 e T3 sono implementate in hardware (a) e quando anche T2 è realizzata nello stesso modo (b)

meno che non siano previsti volumi tali da giustificare l'impiego di una soluzione ASIC, la sola via percorribile è il ricorso a core di CPU di tipo proprietario da implementare all'interno di FPGA o a un ASSP in grado di soddisfare i requisiti imposti dal sistema.
Nel corso dell'articolo verrà descritto l'impiego di soluzioni hardware che permettono di accelerare le componenti dell'algoritmo critiche dal punto di vista delle prestazioni. Per illustrare il procedimento si farà riferimento a un progetto sviluppato da QuickLogic per la realizzazione di un decoder MPEG2 sotto forma di una combinazione ibrida tra moduli hardware e software. Tale soluzione è stata realizzata sfruttando QuickMIPS, componenti SoC di tipo programmabile sviluppati da QuickLogic. Tali dispositivi integrano una CPU a 32 bit e altre periferiche di uso comune, unitamente a una matrice logica programmabile. Quest'ultima, nella quale è implementato l'acceleratore hardware, è connessa al resto del dispositivo mediante un apposito bus di sistema on chip, denominato AMBA. Sebbene i risultati presentati in questo articolo si riferiscono al decodificatore MPEG2, il procedimento utilizzato e le considerazioni progettuali possono essere estese a un gran numero di applicazioni embedded.

Profilazione del codice

Il primo passo consiste nell'identificazione di quegli elementi dell'algoritmo più idonei per il processo di accelerazione. Per svolgere tale compito è necessario individuare le aree del codebase che occupano la maggior parte del tempo di esecuzione del microprocessore. Per ottenere ciò è necessario effettuare la profilazione (profiling) del codice. Sebbene l'operazione possa comportare un rallentamento dell'applicazione, in prima approssimazione permette di indicare gli hot spot (ovvero i punti che impegnano le maggiori risorse della Cpu) principali del codice.
Mediante un compilatore GNU-C è stato possibile inserire uno speciale codice per la misura delle statistiche all'interno degli eseguibili binari. Un programma realizzato sfruttando la profilazione genera un file gmon.out che contiene le statistiche raccolte durante l'esecuzione del programma.. L'utility gprof, che interpreta il file gmon.out, genera i file di testo che indicano il numero di volte che una specifica funzione viene eseguita e il tempo complessivo richiesto. Una volta raggruppate le singole funzioni relative al decoder MPEG2, è stato disegnato l'istogramma riportato in figura 1. In questo istogramma viene evidenziata la percentuale del tempo di esecuzione raggruppato per funzione e ordinato in maniera decrescente. Le quattro funzioni relative all'algoritmo MPEG2 prese in considerazione sono:
- iDCT - Trasformazione coseno inversa discreta
- MC - Compensazione del movimento
- Parse - Laddove avviene la decodifica della maggior parte dei livello superiori del flusso
- VLD/iQ/iZZ - Decodifica della lunghezza variabile (Variable Lenght Decode), quantizzazione inversa (inverse Quantization) e Zig-Zag inversa (inverse Zig Zag).

È possibile notare che queste quattro funzioni richiedono la quasi totalità (oltre il 99%) del tempo di esecuzione.

Realizzazione hardware

Un acceleratore hardware si occupa semplicemente di prelevare i dati, elaborarli e renderli disponibili in uscita. A seconda del particolare tipo di sistema, è necessaria la presenza di interfacce per i dati che devono essere inviati all'acceleratore.
Nel caso della piattaforma QuickMIPS, il modulo include un'interfaccia per bus AMBA che assicura la connettività con il core della CPU on chip e un engine DMA per l'ingresso e l'uscita dei dati dall'acceleratore.
Nel caso una funzione software (come la iDCT), che occupa una larga percentuale (in misura pari al 61% come riportato in Fig. 1) del tempo del microprocessore viene sostituita con una soluzione hardware, ciò non si traduce in un aumento delle prestazioni pari a 2,5. Ciò è imputabile principalmente a due fattori:
- l'overhead di comunicazione tra il core della CPU e l'acceleratore hardware
- l'efficienza del processo di memorizzazione/recupero dei dati dalla memoria e la posizione dei dati nel sistema.

I sistemi che fanno ricorso esclusivamente al software per l'implementazione degli algoritmi occupano gran parte del tempo nell'esecuzione delle operazioni matematiche richieste all'interno

della CPU. Nel momento in cui questi algoritmi vengono realizzati mediante hardware, la CPU tende ad assumere il ruolo di un controllore, coordinando e pianificando le operazioni dell'acceleratore hardware. Solitamente ciò viene eseguito mediante l'impostazioni delle operazioni DMA, la configurazione dei bit del registro e la collocazione delle istruzioni in code di comandi.

Si ipotizzi ad esempio che vi siano tre trasformazioni specifiche, T1, T2 e T3 e che T1 e T3 occupino una larga fetta del tempo della CPU (il 30% ciascuna), mentre T2 richiede solo l'1% di questo tempo. Nel momento in cui T1, T2 e T3 trasferiscono fra di loro grandi quantità di dati l'esclusione di T2 dal processo di accelerazione può avere un impatto significativo, ben più importante rispetto allo scarno 1% del tempo di CPU. Si supponga che ogni trasformazione esegua la lettura di 1 kB di dati e quindi effettui la scrittura della medesima quantità. Se T1 e T3 sono implementate in hardware e T2 in software, il flusso dei dati darà luogo a un traffico pari a 6 Kb, come riportato in figura 2a. Le prestazioni non sono quelle previste poiché l'esclusione di T2 produce un trasferimento di dati aggiuntivo per il sistema. Nella figura 2b tutte le tre trasformazioni sono implementate in hardware e il trasferimento dei dati tra T1/T2 e T2/T3 è circoscritto all'interno dell'acceleratore hardware. Di conseguenza il bus di sistema viene attraversato da soli 2 Kb di dati.

Facendo riferimento alla figura 1, nel caso del decodificatore MPEG2, si nota che dopo la trasformata iDCT e la compensazione del movimento, la funzione

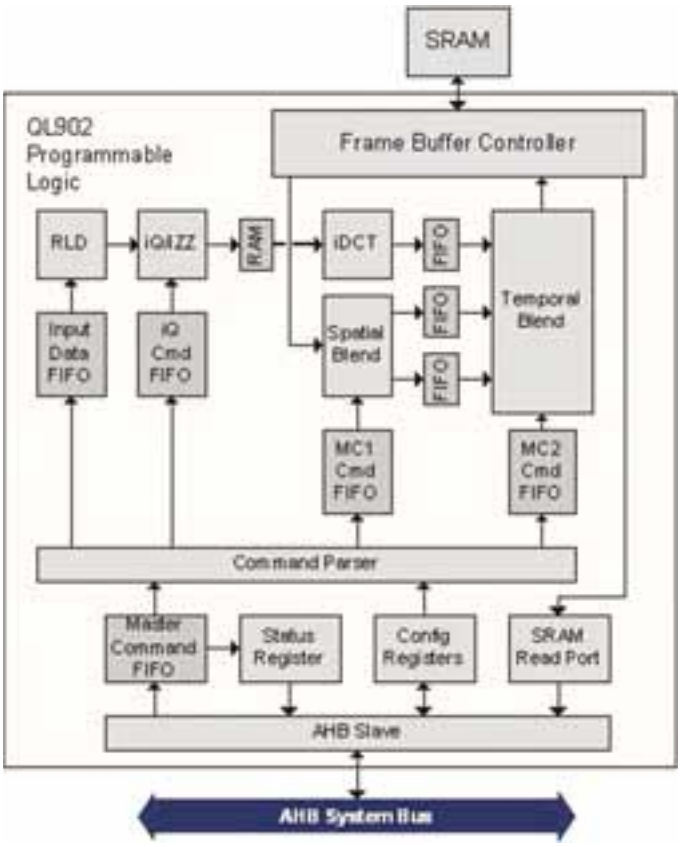

Fig. 3 - Schema a blocchi dell'acceleratore hardware utilizzato per la codifica MPEG2

Parse appare la candidata più probabile per l'accelerazione hardware. A causa del flusso di dati specifico dell'algoritmo, l'accelerazione della funzione Parse senza l'accelerazione di VLD/iQ/iZZ tenderebbe a creare il problema della localizzazione dei dati. Di conseguenza, la candidata migliore per l'accelerazione è proprio la funzione VLD/iQ/iZZ.

Funzionamento parallelo

Il metodo più semplice per partizionare il software in una soluzione ibrida hardware/software è generare un blocco hardware che imiti esattamente le operazioni di una chiamata di una funzione software. Questa metodologia viene denominata sostituzione di funzione. Tale approccio permette di minimizzare l'impatto sul software in quanto non risulta necessario riprogettare i livelli superiori del software. Oltre a ciò, a coloro che si occupano dello sviluppo dell'implementazione hardware non è richiesta una conoscenza particolare dell'algoritmo.

Una tale metodologia, comunque, non permette di ottenere il più elevato livello di prestazioni a livello di sistema poiché il core della CPU rimane nello stato di idle (ovvero a riposo) mentre l'acceleratore hardware sta eseguendo le proprie elaborazioni e solo un blocco hardware alla volta sarà attivo. Inoltre il flusso dei dati non è ottimale poiché ogni blocco deve comunicare con la memoria di sistema piuttosto che effettuare una comunicazione diretta tra blocco e blocco.

Di conseguenza questo approccio appare efficiente se applicato a funzioni autonome che eseguono numerose elaborazione su una piccola quantità di dati.

Per ottenere prestazioni più elevate a livello di sistema è necessaria l'esecuzione in parallelo (simultaneamente) degli elementi dell'algoritmo. Vi sono tre tipi di architetture di dispositivi per le quali è necessario tenere in considerazione l'impatto prodotto dalla simultaneità:

- simultaneità tra CPU e accelerazione hardware: in questo scenario la CPU e l'acceleratore hardware operano contemporaneamente. Ciò si può verificare quando il core della CPU avvia le operazioni hardware non appena i dati dipendenti sono al loro posto ed è in grado di eseguire i compiti nel tempo stesso in

cui l'acceleratore hardware effettua i calcoli;
- simultaneità tra più blocchi dell'acceleratore hardware: ciò richiede l'impiego di blocchi di accelerazione che possono agire indipendentemente l'uno dall'altro;
- all'interno di un blocco dell'acceleratore hardware: poiché spesso gli algoritmi possono essere partizionati sotto forma di una serie (o di una pipeline) di operazioni, è sovente vantaggioso sviluppare l'hardware sotto forma di pipeline di operazioni che possono essere eseguite simultaneamente. Ciò spesso richiede una memorizzazione temporanea (buffering) dei dati in modo da tener conto del fatto che differenti stadi della pipeline richiedono tempi diversi per il completamento e assicurare la sincronizzazione dei dati per il loro utilizzo da parte dell'unità di esecuzione successiva.

L'utilizzo di una metodologia che prevede l'esecuzione simultanea, oltre a richiedere la conoscenza dell'algoritmo, comporta un aggravio del lavoro ingegneristico necessario. Esistono comunque parecchi tool che permettono di automatizzare, in diversa misura, il processo di sostituzione di una funzione. Tra questi si possono segnalare Seamless (per il profiling) e ASAP (per la conversione hardware) di Mentor Graphics, oltre a numerose soluzioni proposte da Coware, Celoxica e Critical Blue. Se la strategia di sostituzione della funzione appare quella più idonea per ottenere le prestazioni richieste in termini di prestazioni e di consumi di potenza, allora questi tool permettono di ridurre sensibilmente gli sforzi richiesti per ottenere un'implementazione ibrida.

In questi casi sarà richiesto un investimento significativo in termini di risorse ingegneristiche che permettano di comprendere appieno l'architettura del dispositivo e la complessità dell'algoritmo per sviluppare un'implementazione ottimizzata. Un tale sforzo aggiuntivo è giustificato solamente se l'approccio che prevede la sostituzione della funzione non è in grado di soddisfare i requisiti richiesti in termini di prestazioni o consumi di potenza.

Il lavoro svolto da QuickLogic indica che, nel caso dell'algoritmo MPEG, questo sforzo aggiuntivo ha dato vita al modulo hardware rappresentato in figura 3, grazie al quale è stato possibile raddoppiare le prestazioni del sistema rispetto a quelle ottenibili con la semplice sostituzione della funzione.

Complessivamente, l'implementazione hardware/software sul SoC programmabile QuickMIPS ha permesso di ottenere prestazioni 10 volte superiori rispetto a quelle di una realizzazione esclusivamente software. Inoltre è stato possibile conseguire un aumento di un fattore pari a cinque a livello di prestazioni del sistema senza nessun incremento in termini di dissipazione di potenza. Evidentemente tuttavia, l'ulteriore incremento delle prestazioni raggiunto dipenderà dall'algoritmo.

In base all'esperienza maturata nel corso di questo progetto, l'entità del guadagno che è possibile ottenere dal punto di vista delle prestazioni utilizzando un'implementazione ibrida dipende fondamentalmente da due fattori:
- caratteristiche dell'algoritmo: efficienza del processo di trasferimento dei segmenti del code base in hardware, evitando i trasferimenti dei dati, onerosi da punto di vista temporale, tra i moduli;
- parallelismo: esso è in grado di garantire un sensibile incremento delle prestazioni rispetto a un approccio che prevede la sostituzione della funzione.

Comunque questo metodo richiede un investimento significativo in risorse e la riscrittura dell'algoritmo per garantire l'elaborazione parallela da parte del core della CPU e della logica che presiede l'accelerazione hardware.

QuickLogic (Silverstar)
readerservice.it n. 12