LA RICONFIGURAZIONE DINAMICA NEGLI FPGA

Una soluzione hardware innovativa per la gestione della riconfigurazione

a cura di
D. Micella, M. Di Ciano, L. Tosi*

Già riconoscibile nei temi di ricerca che vanno dal "reconfigurable computing" al "self adapting computing", la riconfigurazione dinamica parziale, che consente di riprogrammare solo parte di una FPGA mentre il resto del dispositivo continua a funzionare senza alcuna interruzione, ne è la frontiera più avanzata per i diversi vantaggi che offre e di cui, i principali, si possono riassumere in:

- riduzione dei costi di sviluppo,
- riduzione dei costi del dispositivo finale,
- riduzione del consumo di energia,
- incremento delle prestazioni computazionali,
- incremento dell'affidabilità del sistema,
- incremento delle funzionalità.

Poiché tali vantaggi non sono tutti perseguibili contemporaneamente, sarà necessario effettuare delle scelte di progettazione a seconda degli specifici obiettivi da raggiungere. Grazie all'applicazione di concetti alla base della riconfigurabilità è possibile concepire strutture hardware totalmente nuove, mutevoli e virtualizzabili da assegnare all'applicazione secondo specifiche esigenze e solo quando richiesto, rendendo molto più versatile il concetto di disponibilità delle risorse hardware. Sebbene, dal punto di vista hardware sia tecnicamente possibile e realizzabile su diversi dispositivi (per esempio le famiglie Virtex di Xilinx e le FPGA Atmel), presenta ancora notevoli difficoltà soprattutto per l'immaturità degli strumenti di sviluppo.

LA GESTIONE DEI MODULI HARDWARE

Il concetto alla base di un progetto parzialmente riconfigurabile è che le strutture hardware possono non essere tutte contemporaneamente presenti sul silicio, ma condividere le aree dell'FPGA con altre strutture in una sorta di time-sharing".

I moduli hardware vengono configurati all'interno dell'FPGA nel momento in cui servono, e rimossi quando hanno completato la loro funzione per dare spazio ad altri. Questa organizzazione è molto simile a quella dei processi software che condividono tra loro la risorsa CPU.
Supponiamo di voler realizzare un progetto dove su una singola FPGA debbano essere eseguite diverse operazioni (processi) come mostrato nella **Figura 1a**. Un'implementazione classica di questo sistema prevede che tutti i moduli siano implementati e che trovino il loro spazio all'interno del dispositivo, come illustrato nella **Figura 1b**. Un'implementazione parzialmente riconfigurabile permette invece di allocare spazio sull'FPGA solo per i processi (**Figura 1c)** che devono essere eseguiti contemporaneamente - A e B dell'esempio in Figura - riservando una sola area per tutti quelli che possono essere, invece, eseguiti sequenzialmente e che potranno condividere l'area riconfigurabile. Il risultato è un dispositivo più piccolo, che consuma meno, e che svolge esattamente le stesse funzioni del primo.
Ma l'utilizzo di queste tecniche apre la strada a nuove problematiche derivanti dal fatto che l'hardware necessario per eseguire

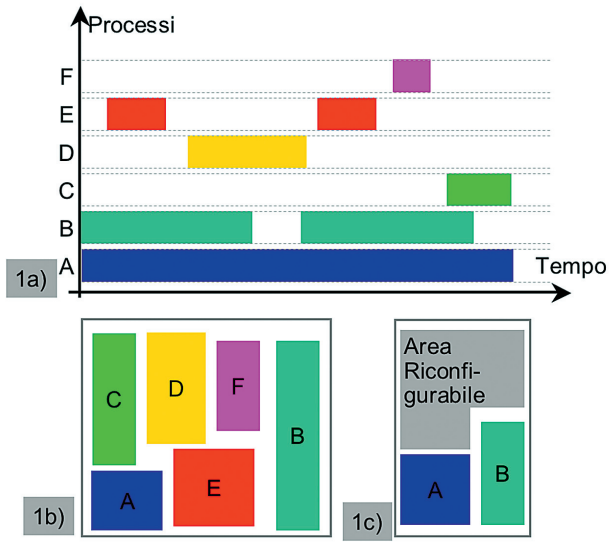


Figura 1- L'esempio di allocazione temporale dei processi riportata in1a) può essere risolta con la soluzione tradizionale in 1b) o con quella basata sulla riconfigurabilità dinamica in 1c).

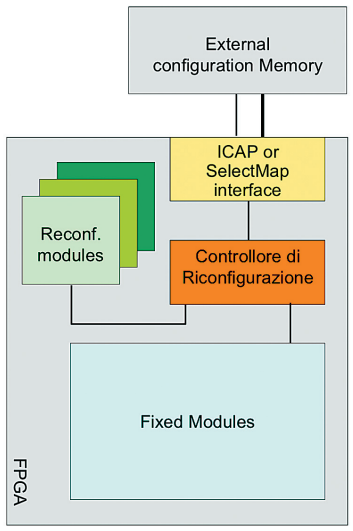


Figura 2 - L'architettura Hw del "Conduttore di riconfigurazione".

una determinata funzione non è sempre disponibile. Questo implica la necessità di realizzare, attraverso un controllore dedicato, un sistema di accodamento delle strutture hardware che devono avere accesso all'area riconfigurabile.

LA RICERCA DI SOLUZIONI INNOVATIVE

All'interno del progetto di ricerca europeo **Recops** (programma Europa N° 05/102.016/010) insieme a tutti gli aspetti di progettazione, sviluppo e ricerca di metodologie applicabili in ambito militare, sono state studiate e sperimentate, presso i laboratori di **Tecnopolis Csata** e **Cesvit Microelettronica** alcune soluzioni per ridurre al minimo la logica necessaria per il controllo della riconfigurazione parziale sulle FPGA **Virtex4** di **Xilinx**.
Le soluzioni fino ad oggi disponibili in letteratura per controllare il processo di riconfigurazione prevedevano l'uso di un microprocessore, interno o esterno, con un software ad hoc che all'occorrenza preleva i dati dalla memoria di configurazione e li presenta sull'interfaccia Icap ***(Internal Configuration Port)*** attraverso un Bus di comunicazione, o sulla porta SelectMap in caso di processore esterno. Però, utilizzando questa soluzione, le aree di silicio risparmiate utilizzando le tecniche di "silicon-sharing" si equivalgono, o sono addirittura inferiori, alle risorse necessarie per la realizzazione del controllore di riconfigurazione, così come proposto in letteratura.
Nell'ambito del progetto Recops si è sviluppata una soluzione alternativa che utilizza una Intellectual Property realizzata all'interno di una porzione fissa di FPGA. Questa IP, denominata "controllore di riconfigurazione", deve amministrare le richieste di riconfigurazione prodotte dalla parte fissa del progetto e renderle operative. Questa IP può essere suddivisa in due moduli distinti: l'interfaccia con la memoria e la porta di configurazione ***(si veda la* Figura 2*)***.

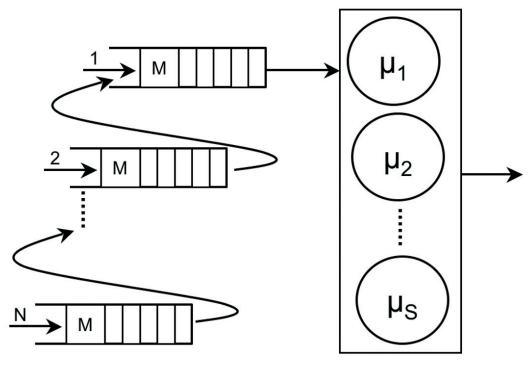


Figura 3 - Profilo FIFO con priorità e S regioni riconfigurabili.

INTERFACCIA CON LA MEMORIA E LA PORTA DI CONFIGURAZIONE

Il primo sub-modulo è quello che realizza fisicamente la riconfigurazione pilotando le porte della memoria che contiene i bitstreams e le porte dell'FPGA che devono ricevere i dati. Questo sub-modulo deve essere progettato per garantire la medesima interfaccia al modulo di schedulazione indipendentemente dalla memoria utilizzata (Eprom, platform flash, linear flash, RAM) e dalla porta di configurazione (Slave SelectMap 8bit, Slave SelectMap 32bit, ICAP 8bit o ICAP 32bit)
Il modulo di schedulazione deve invece collezionare le richieste di riconfigurazione e processarle secondo la politica prescelta. Questa divisione concettuale è utile nel caso si vogliano realizzare più IP combinando le diverse possibilità di ciascun sub-modulo.

IL MODULO DI SCHEDULAZIONE

Un'analisi più approfondita merita invece la scelta della politica di schedulazione dei diversi moduli riconfigurabili. Malgrado non siano disponibili molti studi sulle politiche di schedulazione per i moduli hardware, molti concetti possono essere ereditati dall'esperienza del mondo dei processi software all'interno dei sistemi operativi. I concetti che regolano questi due mondi sono pressoché gli stessi, ciò che cambia sono le condizioni operative. Infatti, se per il mondo del software il context switch overhead è trascurabile rispetto al tempo di elaborazione, permettendo a centinaia di processi di "contendersi" contemporaneamente l'utilizzo della CPU, questo non è vero per il mondo dei processi hardware, dove il tempo di "elaborazione" è molto breve e quello di riconfigurazione non è trascurabile (tempo di download del bitstream parziale). Queste considerazioni ci portano a escludere tutte quelle politiche che prevedono frequenti cambi di configurazione e un comportamento di tipo preemptive.

Una soluzione molto efficiente e versatile è quella di utilizzare un algoritmo FIFO con priorità: possiamo implementare quindi N code con differenti priorità con un numero massimo di richieste in coda pari ad M. Dal lato serventi, possiamo supporre di avere S serventi costituiti da S differenti zone riconfigurabili ognuna in grado di ospitare un modulo parziale. Nella **Figura 3** viene mostrata una rappresentazione della situazione descritta.
Al momento della realizzazione del progetto bisogna valutare come dimensionare l'algoritmo di schedulazione, in quale memoria salvare i bitstreams parziali e quale interfaccia di riconfigurazione usare a seconda degli obiettivi che si vogliono raggiungere.

IL CASO DI STUDIO

Particolare enfasi è stata posta nella realizzazione e sperimentazione di un controllore che fosse in grado di minimizzare le risorse necessarie per la sua realizzazione. Per questo motivo sono state effettuate le seguenti scelte:

- una sola zona riconfigurabile,
- nessun accodamento delle richieste,
- utilizzo della PlatformFlash per immagazzinare i bitstreams parziali.

Quest'ultima scelta ci consente di gestire la riconfigurazione parziale senza dover aggiungere memorie esterne rispetto ad una implementazione statica (la PlatformFlash contiene anche il bitstream totale).
La soluzione proposta, studiata nei laboratori di **Tecnopolis**, è in via di sperimentazione da parte di **Cesvit Microelettronica** all'interno di un dimostratore tecnologico anch'esso progettato e sviluppato nell'ambito del progetto Recops. L'idea di base è quella di utilizzare la capacità di gestire le revisioni della PlatformFlash, la revisione numero 0 deve contenere il bitstream totale, nelle altre revisioni saranno memorizzati i bitstream parziali. Per permettere all'FPGA di riconfigurare se stessa è necessario connettere 2 pin di I/O ai bit di selezione della revisione della PlatformFlash Rev_Sel[1:0].
Un ulteriore segnale deve collegare l'FPGA al pin OE_RESET della memoria; questo pin presetta il contatore della memoria all'indirizzo della revisione selezionata.
Nella **Figura 4** è rappresentato lo schema a blocchi del collegamento PlatformFlash - FPGA e dei vari moduli all'interno dell'FPGA stesso.

Durante il power-up, la PlatformFlash scarica il bitstream totale (salvato nella revisone 0) grazie ai resistori di pull-down. Appena l'FPGA è stata configurata, il controllore di

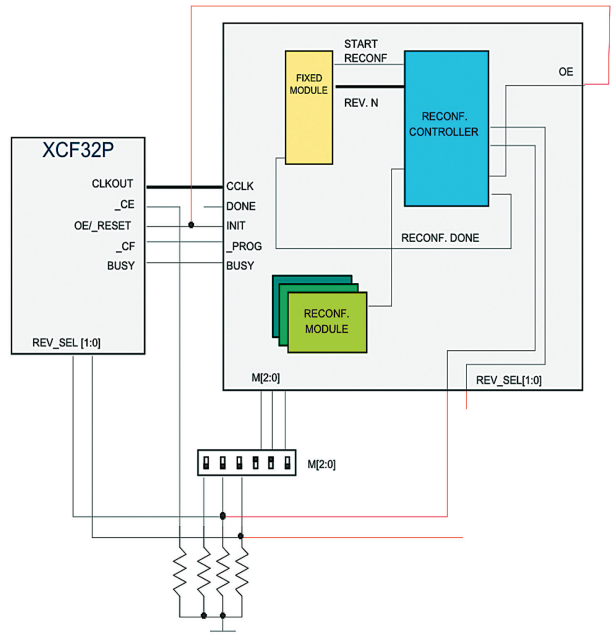


Figura 4 - Il conduttore di riconfigurazione associato alla Platform flash.

riconfigurazione prende il controllo delle linee di selezione della revisione e del pin di OE_RESET. Il processo di riconfigurazione ha inizio con una richiesta da parte del modulo fisso del progetto, che alza il segnale StartReconf e scrive il numero della prossima revisione da caricare sulle linee REV.N; il controllore aspetta un segnale di abilitazione dal modulo riconfigurabile attivo, setta le linee dell'indirizzo della revisione selezionata e pilota l'OE_RESET. A questo punto inizia il download del bitstream parziale attraverso la porta SelectMap preventivamente settata in "persist mode". Alla fine del download il controller invia un reset esplicito al modulo appena configurato (il global reset dell'FPGA non è attivo) e comunica ai moduli attivi che il processo di riconfigurazione è terminato.

I PRO E I CONTRO

In sintesi, possiamo raggruppare i vantaggi della soluzione proposta evidenziando:

- una dimensione estremamente ridotta del controllore di riconfigurazione (FSM a 10 stati + contatore a 30bit);
- il non uso della porta ICAP e nessuna risorsa di routing utilizzata per connettere ICAP al BUS di dati;
- nessun impiego di processori interni o esterni;
- nessun software per controllare il processo di riconfigurazione;
- alta velocità di download 320Mbps (8bit @ 40MHz). Due aspetti meno positivi meritano una breve sintesi, ricordando che la soluzione proposta:
- non supporta la massima velocità di download delle FPGA Virtex 4 (32 bit @ 100MHz);
- il numero massimo di bitstream parziali è limitato dal numero massimo di revisioni supportate all'interno della platformflash (1 bitstream totale + 3 parziali).

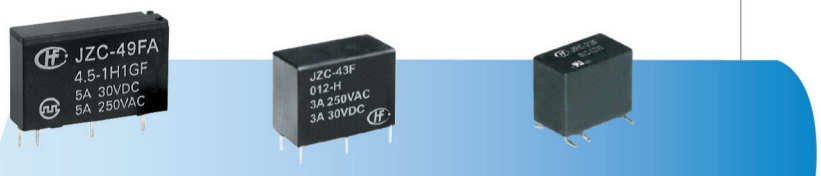
[Donato Milella e Marco Di Ciano sono di Tecnopolis Csata, mentre Leonardo Tosi è di Cesvit Microelettronica]*

Vector Electronic Srl

ELECTRONICS COMPONENTS REPRESENTATIVE

distributore esclusivo di: **HONGFA Relay**

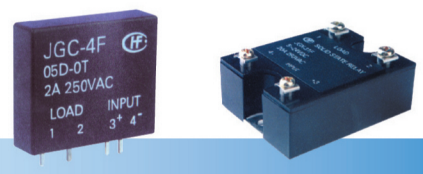
Interfacce



Potenza

Automotive

Stato Solido



Controllo Motori

Un grande magazzino a Milano ...per ogni richiesta

VECTOR Electronic Srl 20037 Paderno Dugnano MI (ITALY) via Pietro Nenni 19 Tel. 026625001 Fax 02 66202851
e-mail: vector@vector.it - www.vector.it

HONGFA Relay · KEKO VARICON · Kingbright · Diotec Semiconductors · EAST · HAHN · PANJIT SEMI CONDUCTOR