



Dynamic Reconfigurability in Embedded System Designs

Verbale della 3-giorni DRES D

DEL 31-07-2006 01-
08-2006 02-08-2006



Contents

Numero 11, Luglio 2006

CONTENUTI

Notizie e avvisi

11

1 Resoconto della 3-giorni DRES D

Durante i giorni di lunedì 31 luglio, martedì 1 agosto e mercoledì 2 agosto si è svolta la prima edizione della 3-giorni DRES D presso l'hotel Villa Gina di Goglio. Si è trattato di una serie di incontri che avevano l'intento di capire la direzione presa dal gruppo durante l'anno passato e verso dove si volesse andare durante il nuovo. Inoltre sono stati definiti gli aspetti logistici e divulgativi: roadmap, DRES D News, sito, survey, gadget, ecc. Infine sono state definite le metriche per la valutazione del lavoro e sono stati delineati i risultati attesi per il periodo 2006/2007. I partecipanti a questa iniziativa sono stati 15. L'apertura della 3-giorni DRES D è stata tenuta da Marco Santambrogio, che ha descritto l'outline di questa prima edizione. Tutto il materiale presentato durante i 3 giorni sarà disponibile sul sito del Laboratorio di Microarchitetture.

1.1 Resoconto della giornata di lunedì 31 luglio

1.1.1 Introduzione di Marco Santambrogio su DRES D

La prima presentazione della 3-giorni DRES D è stata svolta da Marco Santambrogio, che ha presentato un resoconto sul gruppo e sui risultati ottenuti e ha delineato le basi su cui è stato fondato il lavoro.

La filosofia di DRES D può essere riassunta dalle seguenti frasi:

- *Do or do not! There's no try!* (Master Yoda)
- *Nothing worth having comes easy.* (Scrubs)
- *I need to believe that something extraordinary is possible!* (Alicia Nash)

DRES D è un gruppo di persone che si occupano di reconfigurable computing, concentrandosi in particolare sulla riconfigurabilità dinamica parziale interna. Attualmente il gruppo è composto da 30 studenti non laureati (studenti della laurea di primo livello), 12 studenti laureati (studenti della laurea specialistica), 2 dottorandi, 3

ricercatori e 4 professori. Per quanto riguarda gli studenti non laureati si hanno le seguenti percentuali:

- 55% tesisti e 45% progettisti su un campione di 30 studenti del Laboratorio di Microarchitetture nell'anno accademico 2003/2004
- 57% tesisti e 43% progettisti su un campione di 75 studenti del Laboratorio di Microarchitetture nell'anno accademico 2004/2005
- 91% tesisti e 9% progettisti su un campione di 33 studenti del gruppo DRES D nell'anno accademico 2005/2006

Si hanno inoltre collaborazioni con università europee (Università di Paderborn e Heinz Nixoberf Institute) e americane (UIC e Northwestern). La collaborazione con HNI ha portato a lavorare su partitioning, scheduling e Linux sulla board Raptor2000, la collaborazione con UIC ha portato a lavorare su JHDL e su alcune metriche per il partizionamento grazie alle competenze di alcuni professori della UIC sulle memorie, infine la collaborazione con la Northwestern ha portato a lavorare su SyCERS e adaptive computing per l'accelerazione software, codesign e riconfigurabilità. In generale il Laboratorio di Microarchitetture ha partnership con università (Collegio Sant'Anna, Università degli studi di Milano, ALaRI, Universitaet Karlsruhe, ITTC, Indiana University, Purdue University Indianapolis) e aziende (Siemens Mobile, ATMEL, ST-Microelectronics, DA Sistemi, Xilinx, ImpulseC).

I risultati ottenuti all'interno del gruppo DRES D durante l'anno accademico 2005/2006 sono i seguenti: 10 paper, 1 journal e coinvolgimento in conferenze come chair 1 volta, membri del program committee 4 volte, reviewer 6 volte e revisori per transaction, nello specifico TVLSI, 1 volta.

I progetti su cui DRES D ha lavorato e sta lavorando sono i seguenti:

- Caronte, le cui evoluzioni sono Acheronte e YaRA
- Linux

- SyCERS
- Reconfigurability for Reliability, R4R
- aspetti teorici
- ICAP con introduzione del DMA
- BiRF (Bitstream Relocation Filter)
- IP-Core Generator
- EDK System Creator
- Reconfigurable Core Placer Constraints Generator
- Automatic Driver Generator
- BAnMaT (Bitstream Analyzer Manipulator Tool)
- RecOnDemand

Al termine dell'introduzione di Marco si é quindi proseguito con alcune delle presentazioni dei lavori svolti durante questo anno.

1.1.2 Presentazione di Matteo Murgida e Alessandro Panella: IP-Core Generator

Gran parte del tempo impiegato nel flusso di progettazione di un componente hardware viene occupato nella definizione delle interconnessioni tra l'IP-Core e il sistema nel quale esso é collocato. Un core é tipicamente indipendente dall'architettura che lo circonda, quindi é utilizzabile su sistemi differenti, da qui nasce infatti tutto il lavoro per cercare di massimizzare il riuso dei core. L'IP-Core Generator nasce da queste considerazioni e permette di ottenere miglioramenti al flusso di progettazione di un IP-Core e si colloca all'interno del flusso di Caronte con l'obiettivo di avere un processo completamente automatizzato.

Un IP-Core generato tramite l'IP-Core Generator é composto dallo stub, che contiene il core, e dall'interfaccia. Lo stub opera una funzione di demultiplexer dei segnali in entrata e una funzione di multiplexer dei segnali in uscita. Possiamo suddividere i segnali del core in due gruppi: segnali custom, che sono propri della funzionalità implementata, e segnali standard, che sono il segnale di clock, il segnale di reset e il segnale di interrupt. Come requisito per l'interfacciamento tramite l'IP-Core Generator, l'IP-Core non deve contenere alcun riferimento alla modalità di interfacciamento con il sistema.

Il flusso di lavoro dell'IP-Core Generator prevede due fasi: Reader e Writer. Il Reader riceve in ingresso la descrizione VHDL del core. Da qui crea una stringa contenente la descrizione dell'entity del core priva di commenti affinché la presenza di commenti ambigui non venga male interpretata. Infine crea una lista dei generics del core e una lista dei segnali di I/O, che costituiscono gli ingressi del Writer. Dunque il Writer riceve in ingresso la descrizione VHDL del core e le due liste generate dal Reader. Crea i file VHDL che contengono il core, lo stub e l'IP-Core completo di interfaccia, facendo emergere eventuali generics presenti nel core con i relativi valori di default.

E' stata creata la parte di IP-Core Generator che lavora su IPIF e Pselect per OPB e Wishbone. In particolare vediamo il caso IPIF-OPB; una parte dell'IPIF é collegata al bus OPB, mentre l'altra parte, chiamata IPIC, é collegata al componente. Attualmente l'IP-Core Generator non sfrutta tutti i servizi tipici dell'IPIF, ad esempio non sfrutta ISC, S/W Reset e FIFO. Viene invece ampiamente utilizzato il supporto di decodifica degli indirizzi. Il tempo dell'esecuzione del tool é quasi costante, mentre l'occupazione relativa dell'interfaccia IPIF su OPB diminuisce all'aumentare delle dimensioni del core.

Gli sviluppi futuri di questo lavoro sono molteplici, basti pensare al rendere più flessibile e robusta la versione attuale del tool ed estendere il supporto ad altre interfacce e bus.

1.1.3 Presentazione di Massimo Morandi e Marco Novati: BiRF (Bitstream Relocation Filter)

BiRF é un filtro hardware per migliorare le prestazioni della riconfigurazione dinamica interna secondo l'approccio per colonne. Si pensi infatti che alcune delle famiglie di FPGA della Xilinx hanno 5 tipi di colonne divise in N frame a seconda del tipo e si ha un doppio indirizzo (major address e minor address). Inoltre si ha la necessit  di creare un gran numero di bitstream ottenuti come differenza ed infine, se si vogliono elevate prestazioni, non   ammissibile che l'intero processo di creazione dei bitstream avvenga a run time. La soluzione tradizionale a queste considerazioni prevede la memorizzazione di tutti i bitstream parziali che vengono creati a priori, cosa che comporta un notevole lavoro in fase di sintesi e un grande spreco di memoria sul dispositivo.

La soluzione su cui si basa questo lavoro sfrutta la rilocazione. La riconfigurazione per colonne si presta in modo naturale alla rilocazione e semplifica il problema rendendolo monodimensionale. Inoltre   la soluzione adottata per le FPGA della Xilinx. Esistono i seguenti strumenti di rilocazione software: PARBIT (Partial Bitfile Transformer) e BAnMaT (Bitstream Analyzer Manipulator Tool). Tuttavia, per ottenere prestazioni compatibili con la riconfigurazione interna, non   possibile utilizzare questi strumenti software.

L'idea alla base di questo lavoro   semplice, realizzare un componente HW che, presente all'interno dell'architettura mappata su FPGA, si preoccupi di creare a run-time, il bitstream di riconfigurazione corretto, rispetto alla richiesta della funzionalit  e delle risorse in quel momento disponibili sul dispositivo. Questa azione non consiste in altro che la ricostruzione delle informazioni all'interno del bitstream parziale, legate al suo nuovo posizionamento, nuovo rispetto a quello con cui era stato definito in fase di realizzazione. BiRF accetta in input il bitstream da rilocare diviso in blocchi da 32 bit, la colonna iniziale di destinazione del modulo, il numero di colonne CLB del dispositivo, il numero di colonne RAM del dispositivo e la memoria addizionale. Il parser   stato implementato come

una macchina a stati. Si   osservato che il tempo di rilocazione cresce linearmente all'aumentare della dimensione del bitstream.

Un possibile sviluppo futuro di questo lavoro costituisce la creazione di BiRF per un approccio 2D invece che 1D.

1.2 Presentazione di Alessandro Meroni: Reconfigurator Tool

Il Reconfigurator Tool   una utility che permette di effettuare riconfigurabilit  in modo da poter accedere alla scheda da qualunque prestazione. La riconfigurazione avviene attraverso il download di uno o pi  bitstream su dispositivi riconfigurabili.

Nella logica di funzionamento di questo tool i client mandano richieste ad un server e ricevono risposte. Il server effettua l'upload dei bitstream. Le azioni del lato client sono le seguenti: login, settaggio delle impostazioni del Reconfigurator Tool, creazione ed invio del job (ordine) al server e attesa dell'output da parte del server. Le azioni del lato server sono le seguenti: attesa di connessioni da parte di client, gestione di eventuali richieste di configurazione, ricezione del pacchetto job dal client, controllo della coda, esecuzione della riconfigurazione della scheda e invio dell'output al client.

Il Reconfigurator Tool   stato scritto in C++. Le classi principali del client sono le seguenti:

- mainDialog: si occupa della gestione delle connessioni e della ricezione delle informazioni
- downMode: si occupa della creazione e dell'invio del job in modalit  download
- serialMode: si occupa della creazione e dell'invio del job in modalit  seriale
- job:   una classe contenente tutte le informazioni per effettuare la riconfigurazione

Le classi principali del server sono le seguenti:

- serverGui: effettua la creazione del server e gestisce la connessione dei client
- SimpleServer: gestisce le nuove connessioni al server ed é responsabile della comunicazione con quel preciso client
- ClientSocket: crea una nuova istanza per ogni classe che si connette al server, interpreta i comandi ed esegue le richieste

Una prima versione del Reconfigurator Tool si basava principalmente sull'utilizzo di una console, attraverso la quale veniva eseguito il programma. I vantaggi sono i seguenti: controllo remoto, login degli utenti, riconfigurazione interna ed esterna, user accounting e gestione delle prioritá. Gli svantaggi sono i seguenti: stretta dipendenza da IMPACT, inesistenza dell'interfaccia grafica e poca intuitivitá. Ad oggi é necessario implementare il login, l'analisi del job e lo spooler, mentre sono in fase di testing la creazione, l'invio e la ricezione del job.

I possibili sviluppi futuri di questo lavoro sono l'agenda in supporto allo spooler, il remainder del job via e-mail e il controllo dell'output visivo attraverso webcam.

1.3 Resoconto della giornata di martedí 1 agosto

1.3.1 Presentazione di Vincenzo Rana: Supporto di un sistema operativo per sistemi riconfigurabili distribuiti

Il lavoro presentato considera come sistema riconfigurabile la board Raptor2000, che é costituita da 1 FPGA master Virtex-2Pro (XC2VP20) e da 2 FPGA slave Virtex-II (XC2V4000).

I livelli del flusso di lavoro sono i seguenti:

- applicazioni
- gestione dei moduli
- allocazione
- posizionamento

- configurazione
- hardware

L'architettura software é costituita da applicazioni software che comunicano con il demone ROTFL (Reconfiguration On The FPGA with Linux) attraverso socket. Una volta stabilita la connessione, viene mandato un comando al demone, che elabora la richiesta e manda la risposta al client. La comunicazione tramite socket é lenta, quindi non si usano i socket per comandi di input e output sui device. Il demone ROTFL é composto da 3 manager:

- ROTFL_Module Manager: riceve la chiamata e verifica se sul dispositivo é già presente un modulo che esegue la funzione richiesta; se non esiste, passa la richiesta al ROTFL_Allocation Manager
- ROTFL_Allocation Manager
- ROTFL_Positioning Manager: prende dal repository il bitstream corretto per configurare il modulo

Nel kernel é presente il modulo LOL (Load On Linux), che permette di caricare e scaricare i driver in modo automatico.

Le performance sono le seguenti:

- 500 millisecondi per lo startup del demone, che viene eseguito 1 volta
- 650 millisecondi per il setup del device driver, che viene eseguito 1 volta per ogni tipo di modulo
- 3450 microsecondi per il loading del modulo se il modulo non é in cache
- 2500 microsecondi per il loading del modulo se il modulo é in cache
- 3.6 microsecondi per la lettura di 4 byte
- 2.7 microsecondi per la scrittura di 4 byte

Gli sviluppi futuri di questo lavoro sono i seguenti: miglioramento degli algoritmi per l'allocazione e il posizionamento, supporto ethernet, repository dei moduli dinamico e scenario distribuito con piú FPGA master.

1.3.2 Presentazione di Chiara Fornoni: Pip

Si hanno grafi con nodi che rappresentano le operazioni e si estraggono i task graph, che sono grafi ridotti al minimo. Una delle fasi del lavoro é la scelta dei template. L'obiettivo di questo lavoro é minimizzare il tempo di esecuzione tramite possibili disposizioni dei template sfruttando un'evoluzione delle politiche di creazione dei template. E' stato eseguito un confronto tra vari metodi per trovare la soluzione ottima, valutando i parametri di ogni metodo/algoritmo, valutando le prestazioni, estraendo le caratteristiche ottime ed effettuando test.

1.3.3 Presentazione di Matteo Giani: Identificazione di moduli per sistemi riconfigurabili guidata da strutture ricorrenti della specifica

L'obiettivo di questo lavoro é ottimizzare l'implementazione della specifica su dispositivi riconfigurabili. Gli estremi sono i seguenti: da una parte schedule ottimo ma area massima, dall'altra utilizzo di una sola unità funzionale ma prestazioni temporali attendibili pessime. Nel mezzo c'è uno spazio di soluzioni all'interno del quale é possibile immaginare di tracciare una curva che identifica per ogni area la prestazione temporale attendibile migliore. Il problema é costituito dall'introduzione di vincoli temporali per il progettista. Tramite la riconfigurazione dinamica si vuole mostrare al progettista un'area maggiore di quella a disposizione. E' però necessario ottimizzare i tempi di riconfigurazione.

Si parte da una specifica in C o in SystemC e si passa ad una rappresentazione intermedia, che in questo caso é un Data Flow Graph. Si passa poi ad un DFG partizionato e infine all'implementazione su moduli riconfigurabili. Per passare dalla specifica al DFG, sono stati utilizzati il framework PandA, che usa C++, C++ STL e Boost

Graph Library, e il parser del gcc. Per passare dal DFG al DFG partizionato, sono stati ricercati i sottografi isomorfi. Si vuole cercare di identificare strutture ricorrenti all'interno della specifica per minimizzare il tempo di riconfigurazione e riusare i moduli riconfigurabili. Le metriche per la scelta dei template sono le seguenti: scegliere prima i template piú grossi, scegliere prima i template piú ripetuti, considerare il peso della comunicazione. Sono stati considerati due algoritmi per la scelta dei template: albero rovesciato e forma libera.

Gli sviluppi futuri di questo lavoro sono i seguenti: raffinamento delle metriche di scelta dei template, euristiche per la scelta dei moduli fissi/riconfigurabili, scheduling e posizionamento dei moduli riconfigurabili on line.

1.3.4 Presentazione di Laura Frigerio: Un nuovo framework per il design e la simulazione di sistemi hardware/software complessi

Visto che i sistemi digitali sono sempre piú complessi e le richieste di mercato sono piú stringenti, si cercano nuovi flussi di codesign. Il classico flusso di codesign é il seguente:

- specifica
- partizionamento
- descrizione hardware e descrizione software in parallelo
- sintesi hardware e sintesi software in parallelo
- integrazione
- simulazione

Un altro flusso tradizionale prevede un'unica descrizione hardware e software dopo la specifica. Innalzando il livello di astrazione, si utilizzano SystemC e Matlab. Lo svantaggio nell'innalzamento del livello di astrazione costituisce nel fatto che i risultati di sintesi non sono sempre accettabili.

La metodologia proposta vuole avere un alto livello di astrazione, ottenere buoni risultati di sintesi e usare back annotation. Il flusso proposto é il seguente:

- specifica con una schematic entry
- analisi dello spazio delle soluzioni tramite POET per la parte software e XRR per la parte hardware
- partizionamento a livello di funzionalità
- incapsulamento software in SystemC, traduzione hardware da RTL a SystemC e generazione delle interfacce tramite RoadRunner
- modello SystemC
- simulazione timing ad alto livello
- back-end software e back-end hardware

Attualmente é stata sviluppata la parte hardware del flusso proposto. I moduli che compongono la libreria sono IP atomici (o atomi) e IP molecolari (o molecole). Durante la strutturazione del sistema il designer deve istanziare gli atomi e le molecole, connetterli e assegnare valori ai parametri. Un parametro puó essere esplorabile, cioè essere lasciato non assegnato nella fase di design, o non esplorabile. Ci sono 3 azioni collegate all'uso dei core:

- sviluppo, che é l'implementazione di una classe C++ che genera il codice VHDL per il core
- istanziamento
- incapsulamento

Possiamo avere moduli META, che descrivono la funzionalità, e moduli CONCRETE, che implementano la funzionalità definendo tutti i dettagli. Ogni modulo META ha uno o piú moduli CONCRETE, che sono implementazioni di una particolare architettura. La scelta dei moduli CONCRETE é determinata da area, tempo e potenza.

Alla fine del processo abbiamo il codice VHDL del sistema. E' possibile identificare i colli di bottiglia del design e le parti che occupano piú area. Attualmente sono stati implementati adder, encoder/decoder, CORDIC, approssimatori di funzioni, memorie, moltiplicatori, ecc.

Il framework é basato sul toolset RoadRunner, che é composto da RRCORE, che é una collezione di core, e RRCACHE, che é una cache. I work in progress sono i seguenti: definizione dell'interfaccia grafica, definizione della netlist XML e definizione della cache. Possibili lavori futuri sono l'estensione del flusso al dominio software e la definizione dei vincoli legati alla propagazione degli errori.

1.3.5 Presentazione di Christian Bruno: Realizzazione di un componente software per DIOPSIS 740, riconoscimento delle forme con sensori sonar

Il DIOPSIS 740 é un chip biprocessore composto da un processore ARM 7, che é un processore general purpose la cui architettura é basata sulla tecnologia RISC, e mAGIC, che é un DSP ad alte prestazioni. I vantaggi sono il parallelismo, la sincronizzazione e l'elevata potenza di calcolo. Il problema é la comunicazione. Il DIOPSIS é montato sulla scheda JTST (Jig Test), che é una scheda prototipale che mette a disposizione alcune periferiche per testare e valutare le prestazioni della DIOPSIS.

1.3.6 Presentazione di Carlo Curino: JHDL (Just another Language Description Language)

JHDL é una tool suite e un design environment basato su Java. I vantaggi che comporta sono i seguenti:

- si tratta di un linguaggio generalmente conosciuto;
- é una ricca catena di tool;
- le librerie GUI sono integrate;

- il caricamento e lo scaricamento delle funzionalità sono mappati sull'istanziamento di oggetti.

JHDL può essere integrato in Acheronte nella parte di design, in particolare simulazione, debugging e hardware techmap. Non impatta sulle performance di Acheronte e non sostituisce lo sviluppo dei core VHDL, ma rende più veloce lo sviluppo dei core.

1.4 Resoconto della giornata di mercoledì 2 agosto

1.4.1 Presentazione finale

Nella prima parte di questa presentazione e in istanti successivi sono state discusse le definizioni legate al reconfigurable computing. Tali discussioni proseguiranno in una serie di incontri tra Marco Santambrogio, Vincenzo Rana e Carlo Curino per la realizzazione di un documento che definisca in maniera univoca i concetti e le definizioni base per il lavoro legato al reconfigurable computing del nostro gruppo di ricerca.

Sono state poi presentate due roadmap, una definita a marzo 2006 e l'altra delineata a luglio 2006. Gli obiettivi e le relative deadline sono i seguenti. Per quanto riguarda la roadmap definita a marzo:

- partecipazione dei docenti alle riunioni DRESA (anno accademico 2006/2007)
- lezioni di reconfigurable computing all'interno di un corso di laurea specialistica (anno accademico 2006/2007)
- workshop DRESA (anno accademico 2006/2007)
- collaborazioni con altri settori del DEI (anno accademico 2006/2007)
- collaborazioni con altre università (anno accademico 2006/2007)
- meeting DRESA della durata di 3 giorni (anno accademico 2007/2008)

- corso di reconfigurable computing (anno accademico 2007/2008)

Per quanto riguarda la roadmap definita a luglio:

- 3-giorni DRESA (obiettivo raggiunto)
- workshop DRESA (obiettivo raggiunto)
- collaborazione con altre sezioni del DEI (obiettivo raggiunto)
- collaborazione con altre università (obiettivo raggiunto)
- poster sul lavoro all'interno di DRESA (settembre 2006)
- riunioni DRESA aperte al pubblico (settembre 2006)
- DRESA News (ottobre 2006)
- CD DRESA per gli studenti e ISO scaricabili dal sito del Laboratorio di Microarchitetture (novembre 2006)
- definizione della 3-giorni DRESA 2007 (dicembre 2006)
- flyer per tutti i progetti (dicembre 2006)
- gadget (dicembre 2006)
- lezioni di reconfigurable computing tenute all'interno di un corso (dicembre 2007)
- corso di reconfigurable computing (dicembre 2008)

Infine sono state definite le metriche per la valutazione di DRESA, suddivise come segue:

- didattica: numero di studenti per i progetti (Reti Logiche A, Laboratorio Software, High Performance Processors and Systems, Sistemi embedded, Metodologie di Progetto Hardware e Software, Metodologie di Progetto Hardware), numero di studenti per la tesi di primo livello, % numero di progettisti - numero di tesisti, numero di studenti rimanenti dopo il terzo anno per la tesi di laurea specialistica

- conferenze e affini: numero di articoli a conferenza, numero di articoli a rivista, numero di reviewer, numero di membri del Program Committee, numero di organizzatori di conferenze (conference chair, session chair, ecc.)
- gestione: numero di riunioni, numero di presentazioni
- manualistica
- qualità del codice

Notizie e avvisi

Notizie

- Tutte le slide di supporto alle presentazioni avvenute durante la 3-giorni DRESA saranno disponibili sul sito del Laboratorio di Microarchitetture appena possibile. Perciò chi ha presentato durante la 3-giorni DRESA é pregato di inviare il materiale a Marco Santambrogio: marco.santambrogio@polimi.it.

Link consigliati

- Microarchitectures Laboratory web site: <http://www.microlab-mi.net/>