Riconfigurabilità Parziale Dinamica per lo sviluppo di dispositivi intelligenti

Inquadramento della ricerca

Il mercato dei dispositivi digitali diventa ogni giorno sempre più affamato di prestazioni: dimensioni ridotte, elevata capacità computazionale e bassa dissipazione di potenza. L'esempio classico è dato dai dispositivi embedded come *smartphone, tablet, PDA, console multimediali*, ecc.
Oggi i sistemi basati su microprocessore non sono in grado di fornire buone prestazioni in molti di questi domini. I microprocessori più moderni implementano generalmente un'architettura di tipo Reduced Instruction Set Computer (RISC) che è basata su un insieme fisso e relativamente limitato di istruzioni. Quest'insieme, e più precisamente l'Instruction Set Architecture (ISA), viene definito con l'intento di raggiungere la massima efficienza possibile per un ampio spettro di applicazioni, mitigando tra dimensione, costo e potenza dissipata. A causa della continua crescita dello spazio delle applicazioni, alcune aziende costruttrici di microprocessori hanno esteso l'ISA con istruzioni adatte all'elaborazione di grandi flussi di dati, tuttavia la definizione di un ISA che sia efficiente in tutti i domini applicativi, risulta essere praticamente impossibile.
Le tendenze attuali si stanno dirigendo verso il *multicore* (più processori integrati nello stesso chip) e i dispositivi riconfigurabili. Il paradigma *multicore* ha un grande potenziale, ma la sua efficacia è limitata principalmente dalla difficoltà di rendere parallela l'esecuzione dei diversi task del software, scritto per un'esecuzione sequenziale, e dalla difficoltà di rendere trasparente al programmatore la comunicazione e la sincronizzazione dei task eseguiti dai diversi processori.
Tra i diversi dispositivi riconfigurabili proposti dalla ricerca e dall'industria, i sistemi composti da una FPGA accoppiata con una o più CPU stanno emergendo sempre più, particolarmente nel campo dei sistemi embedded [1]. Le principali ragioni sono: tempi brevi di sviluppo, alta flessibilità e possibilità di poter sfruttare tali dispositivi sia come prototipi, durante la fase di sviluppo, che come prodotti finali. Le schede FPGA, infatti, possono essere riconfigurate, parzialmente o completamente, un numero idealmente infinito di volte, eliminando i componenti ed i collegamenti logici precedentemente realizzati e creandone di nuovi al fine di modificare o aggiungere nuove funzionalità all'architettura [2].
Esistono due tecniche di riconfigurazione parziale:

1) Statica: il dispositivo, per poter essere riconfigurato, necessita di essere riavviato;
2) Dinamica: la riconfigurazione del dispositivo avviene durante il suo normale funzionamento e quindi in tempo reale [3].

La riconfigurazione parziale statica è stata accantonata principalmente per due motivi: tempi lunghi per la riconfigurabilità e necessità di dover interrompere le applicazioni, entrambi dovuti al riavvio

del dispositivo. La riconfigurazione parziale dinamica, non solo supera questi limiti, ma introduce il vantaggio della riduzione dell'area globale dell'architettura, permettendo così la scelta di dispositivi più piccoli, con conseguente risparmio in termini di potenza dissipata e costo monetario.
La riconfigurazione parziale dinamica è indubbiamente una delle tecniche più innovative e come tale presenta ad oggi alcune problematiche aperte, come, ad esempio, quali strumenti fornire agli sviluppatori software senza che essi debbano necessariamente avere una profonda conoscenza dell'architettura di destinazione e delle tecniche di progettazione digitale, oppure quali strategie usare per velocizzare il processo di riconfigurazione, o infine come ottimizzare il consumo di potenza.

Stato dell'arte

Gli studi sulla problematica riguardante quali strumenti fornire agli sviluppatori software, hanno prodotto i seguenti tools:

- JBits [4], uno strumento potente e flessibile sviluppato dalla Xilinx, che permette la configurazione di elementi logici quali registri, flip-flop, LUT ecc., fungendo da interfaccia tra il programmatore e il file di configurazione del dispositivo (*bitstream*). In pratica, permette al programmatore di vedere il bitstream non come un flusso di bit, ma, più intuitivamente, come un array di risorse configurabili. JBits può operare sia su bitstream creati da altri tools (la Xilinx mette a disposizione i tools ISE e EDK [5]), sia con bitstream letti a run-time dall'FPGA. In realtà non opera direttamente sull'FPGA: carica da questa o dall'hard-disk il bitstream nella Ram, lo modifica e infine riconfigura l'FPGA o crea un nuovo bitstream sull'hard-disk;
- JHDL [6] [7], sviluppato presso i laboratori dell'Università di Brigham Young. Basato su Java, è nato con lo scopo di fornire ai progettisti software gli strumenti per realizzare applicazioni capaci di modificare dinamicamente, e in maniera naturale, il comportamento dei dispositivi riconfigurabili utilizzando le astrazioni disponibili nei classici linguaggi di programmazione object-oriented. Infatti, JHDL gestisce le risorse di tali dispositivi nello stesso modo con cui i linguaggi di programmazione di alto livello gestiscono la memoria: ogni parte riprogrammabile è configurata all'interno della piattaforma mediante l'invocazione del suo costruttore, che effettivamente ne costruisce un'istanza sul dispositivo, mentre l'invocazione del distruttore libera la risorsa. La differenza più evidente che si evince dall'analisi di JHDL rispetto ai tools della Xilinx, risiede nell'approccio implementativo: mentre in questi ultimi per la realizzazione dell'architettura fisica è impiegato un linguaggio di tipo HDL, con la possibilità di sintesi sia strutturale sia comportamentale, JHDL supporta solo la sintesi strutturale, ma permette ai progettisti di specificare

quali circuiti realizzare senza definire nessun dettaglio applicativo sulle operazioni da eseguire sulla piattaforma;
- JHDLBits [8], sviluppato presso i laboratori della Virginia Tech, con lo scopo, come indicato dal nome stesso, di integrare i tools JHDL e JBits, in modo tale da creare un ambiente in grado di consentire sia l'utilizzo dei meccanismi di basso livello per l'accesso ed il controllo a run-time delle risorse riconfigurabili fornito da JBits, che nel contempo la visibilità strutturale di alto livello della descrizione dei componenti logici implementata in JHDL.

In merito alla problematica riguardante quali strategie usare per velocizzare il processo di riconfigurazione, i ricercatori dell'Università di Carbondale hanno elaborato un algoritmo euristico basato sulla manipolazione dell'ordine degli ingressi delle LUTs [9] per minimizzare la dimensione del bitstream. Per comprendere meglio, osserviamo che la memoria di configurazione delle schede FPGA è organizzata in *frames*. Il numero dei frames che richiedono di essere modificati è strettamente legato a quello dei minitermini della funzione xor tra la funzione implementata dalla LUT prima della riconfigurazione e quella dopo la riconfigurazione. Riordinare gli ingressi permette di diminuire tale valore. Provare tutte le possibili permutazioni degli ingressi per ogni LUT, però, sarebbe troppo lento, pertanto l'algoritmo analizza per ogni LUT il numero di frames che hanno bisogno di essere modificati e, dopo averne determinato il valore più piccolo, lo imposta come numero massimo di frames modificabili per terminare le permutazioni.
Recenti studi effettuati della Xilinx hanno evidenziato che in un sistema riconfigurabile le due principali fonti di consumo di potenza sono quella statica e quella dinamica; la prima fonte però risulta essere spesso preponderante [10]. Un successivo studio [11], inoltre, ha osservato che il consumo di potenza statica è fortemente legato alla dimensione del bitstream. Le tecniche di minimizzazione del file di configurazione pertanto presentano il duplice vantaggio di velocizzare la riconfigurazione e di minimizzare il consumo di potenza.
Sebbene la riconfigurabilità a run-time abbia insita nella sua logica l'ottimizzazione del consumo di potenza, il processo di riconfigurazione introduce esso stesso un ulteriore dissipazione di potenza, pertanto non è sempre immediatamente chiaro quando questa tecnica possa portare reali benefici. Interessanti studi e prove sperimentali condotte da un gruppo di ricercatori della Microsoft [12] hanno dimostrato che la riconfigurabilità a run-time effettivamente produce il risultato atteso se la velocità di configurazione del dispositivo è sufficientemente elevata.

Progetto di ricerca

Finalità di questo progetto di ricerca è l'analisi e lo sviluppo, attraverso la riconfigurabilità parziale dinamica, di un framework che permetta la realizzazione di dispositivi intelligenti, in grado di

modificare la propria configurazione secondo determinate condizioni o necessità che il dispositivo stesso valuta: carico di lavoro, dominio applicativo, ecc.

Il primo obiettivo sarà un'analisi esaustiva delle più moderne piattaforme riconfigurabili, delle tecniche di riconfigurabilità dinamica più performanti e dei più innovativi linguaggi di progettazione dei circuiti digitali [14] che supportino la sintesi comportamentale, cioè la possibilità di poter descrivere il funzionamento del dispositivo attraverso algoritmi. Si studieranno, inoltre, vari tools in grado di analizzare il codice sorgente e di determinare quali task associare alla CPU e quali invece alla FPGA, nonché frameworks che, senza stravolgere le tecniche di progettazione hardware attualmente già esistenti, mirino a fornire una virtualizzazione delle risorse riconfigurabili disponibili permettendo di disaccoppiare la progettazione dei circuiti logici da quella delle applicazioni che li utilizzano, così che le due progettazioni possano essere sviluppate in maniera indipendente.

Da una prima analisi dello stato dell'arte sui linguaggi si evince che JHDL è il più idoneo, pertanto, una prima sperimentazione si procederà all'estensione delle sue caratteristiche attraverso l'implementazione di un framework in grado di interpretare un sottoinsieme dei comandi disponibili in Java, in particolare quel sottoinsieme che può essere supportato da un sintetizzatore HDL. In questo modo, l'algoritmo potrà essere facilmente tradotto attraverso una sostituzione testuale diretta. Il vantaggio di questa strategia, oltre alla semplicità ed immediatezza, è che permetterebbe ai circuiti di poter essere simulati in un contesto naturale insieme ad altri circuiti e fornirebbe anche un chiaro percorso di sintesi comportamentale che sfrutterebbe gli strumenti attualmente disponibili.

Successivamente alla realizzazione del framework, si passerà allo sviluppo dei dispositivi intelligenti attraverso tecniche di intelligenza artificiale, ricerca operativa e software computing [15], [18],[19]. La prima soluzione che sarà analizzata sarà lo sviluppo di una macchina a stati finiti interna alla FPGA, direttamente in un'area non riprogrammabile, interfacciata con ICAP (Internal Configuration Access Port). L'utilizzo della macchina a stati finiti e ICAP permetterebbe l'eliminazione della CPU, per il controllo della gestione della configurazione, e il collegamento diretto tra la porta di configurazione interna e la memoria esterna, il cui contenuto, però, dovrà essere leggermente modificato per realizzare un vero e proprio microcodice per comandare ICAP. Ciò eliminerebbe i ben noti problemi concernenti limitazioni in termini di frequenza di lavoro, sicurezza dei dati e accesso sicuro alla memoria.

In questa fase dovranno, inoltre, essere affrontate e risolte le seguenti problematiche: limitazioni nella dimensione delle aree riconfigurabili della scheda e, intimamente legato al precedente, limitazioni nella fasi di *placement* e *routing* dei circuiti nelle suddette aree riconfigurabili.

- Per comprendere bene le limitazioni di dimensione e *placement* ricordiamo che la memoria di configurazione delle schede FPGA è organizzata in frames che rappresentano la più piccola porzione del dispositivo che può essere configurata singolarmente e dinamicamente. La suddivisione in frames è molto importante per il partizionamento della FPGA dato che è indispensabile che ogni singola regione dinamicamente riconfigurabile coincida, sia nelle dimensioni che nei confini, con un numero intero di frames. Più è grande il singolo frame minore sarà la granularità delle regioni riconfigurabili.
- Per quanto riguarda il *routing*, invece, sarà necessario fare in modo che i segnali che attraversano i confini di una regione riconfigurabile siano consistenti con il resto del progetto. I problemi potrebbero infatti nascere se si riconfigura una di queste regioni introducendo dei segnali che attraversano i confini in punti differenti. Questo produrrebbe una chiara inconsistenza nel routing che forzerebbe l'eliminazione del segnale.

Successivamente all'analisi e alle soluzioni proposte ci si propone di rendere il dispositivo sufficientemente performante per poter essere utilizzato in ambienti mobili [20],[21]. Le analisi affrontate nello stato dell'arte saranno un buon punto di partenza. In particolare si cercherà di minimizzare il file di configurazione partendo dal lavoro svolto in [9] e, come suggerito dagli stessi autori, fondendo l'algoritmo di basso livello proposto con altri di alto livello. Minimizzare il bitstream, come già ampiamente evidenziato, porterà sicuramente notevoli vantaggi non solo nel risparmio di tempo necessario per la riconfigurazione (aspetto critico per molte applicazioni, come quelle multimediali di streaming o di video processing), ma anche nella dissipazione di potenza.
Per l'ottimizzazione del consumo di potenza, saranno inoltre analizzati anche gli algoritmi descritti in [13]. Questi però sono indirizzati verso specifiche applicazioni e fortemente legati ad alcune architetture. Si cercherà pertanto di generalizzarli ed estenderli a più architetture e più domini applicativi [16],[17]. Al fine di trovare il miglior risultato possibile, inoltre, potrebbe essere adottato un altro espediente come, per esempio, la variazione della frequenza di clock in base al carico di lavoro.

Bibliografia

[1] P. Garcia, K. Compton, M. Schulte, E. Blem, W. Fu: "An Overview of Reconfigurable Hardware in Embedded Systems", *EURASIP Journal on Embedded Systems, Volume 2006 Issue 1*, 2006.

[2] S. Hauck: "The Roles of FPGA's in Reprogrammable Systems", *Proceeding of the IEEE, Volume 86 Issue 4*, 1998.

[3] J.C. Ferreira, M.M. Silva: “Run-time reconfiguration support for FPGAs with embedded CPUs: The hardware layer”, *Proceeding of the 19th IEEE International on Parallel and Distributed Processing Symposium*, 2005.

[4] S. Guccione, D. Levi, P. Sundararajan: “JBits: Java based interface for reconfigurable computing”, *2nd Annual Military and Aerospace Applications of Programmable Devices and Technologies Conference (MAPLD)*, 1999.

[5] Xcell Journal, issue 55 Forth Quarter 2005: www.xilinx.com/publications/archives/**xcell**/**Xcell55**.pdf

[6] P. Bellows, B. Hutchings: “JHDL - An HDL for Riconfigurable Systems”, In J. M. Arnold and K. L. Pocek, editors, *Proceedings of IEEE Workshop on FPGAs for Custom Computing Machines, pages 175-184*, Napa, CA, 1998.

[7] P. Bellows, B. Hutchings, J. Hawkins, S. Hemmert, B. Nelson, M. Rytting: “A CAD Suite for High-Performance FPGA Design”, *Proceedings of the Seventh Annual IEEE Symposium on Field-Programmable Custom Computing Machines, p.12*, April 21-23, 1999.

[8] A. Poetter, J. Hunter, C. Patterson, P. Athanas, B. Nelson, N. Steiner: “JHDLBits: The Merging of two Worlds”, *Field-Programmable Logic and Applications, (Springer-Verlag), pp. 414–423,* August 2004.

[9] K. P Raghraman, H. Wang, S. Traugodas: “A Novel Approach to Minimizing Reconfiguration Cost for LUT-Based FPGAs”, *Proceeding of the 18th International Conference on VLSI Design held jointly with 4th International Conference on Embedded System Design*, 2005.

[10] A. Telikepalli: “Power vs performance: the 90nm inflection point”, Xilinx Inc., Tech. Rep., April 2005.

[11] T. Tuan, B. Lai: “Leakage power analysis of 90nm FPGA”, *Proceeding of IEEE on Custom Integrated Circuits Conference*, 2003.

[12] S. Liu, R.N. Pittman, A. Forin: “Energy Reduction with Run-Time Partial Reconfiguration,” *MSR-TR-2009-2017, Microsoft Research, WA, September 2009. Presented at the 18th Symposium on Field-Programmable Gate Arrays, Monterey, CA, February 2010. Revised for the 21st International Conference on Application-specific Systems, Architectures and Processors, Rennes, France*, July 2010.

[13] D. Chen, J. Cong, Y. Fan: “Low-Power High-Level Synthesis for FPGA Architectures”, *Proceedings of the 2003 international symposium on Low power electronics and design,* August 25-27, 2003, Seoul, Korea.

[14] F. Qi, X. Zhang, S. Wang, X. Mao: “RCC: A New Programming Language for Reconfigurable Computing”, *Proceeding of the 11th IEEE International Conference on High Performance Computing and Communications*, July 2009.

[15] C. Militello, V. Conti, S. Vitabile and F. Sorbello, “Embedded Access Points for Trusted Data and Resources Access in HPC Systems”, The Journal of Supercomputing - An international journal of High-Performance Computer Design, Analysis and Use, Springer Netherlands Publisher, 2011, ISSN 0920-8542, Vol. 55, N° 1, pp. 4 – 27, (ISSN Online 1573-0484), DOI:10.1007/s11227-009-0379-1

[16] C. Militello, V. Conti, S. Vitabile, F. Sorbello, “An Embedded Iris Recognizer for Portable and Mobile Devices”, Special Issue on "Frontiers in Complex, Intelligent and Software Intensive Systems" of International Journal of Computer Systems Science & Engineering, Vol. 25, n° 2, pp. 119-131, © 2010 CRL Publishing Ltd., ISSN: 0267-6192

[17] V. Conti, C. Militello, S. Vitabile and F. Sorbello, “A Multimodal Technique for an Embedded Fingerprint Recognizer in Mobile Payment Systems”, International Journal on Mobile Information Systems - Vol. 5, No. 2, 2009, pp. 105-124, IOS Press Ed., ISSN: 1574-017X, DOI:10.3233/MIS-2009-0076

[18] V. Conti, S. Vitabile, L. Agnello, F. Sorbello, “Fingerprint and Iris based Authentication in Inter-cooperative Emerging e-Infrastructures”, Studies in Computational Intelligence (Internet of things and inter-cooperative computational technologies for collective intelligence), ISBN 978-3-642-34951-5, ISSN 1860-949X, Vol. 460, 2013, pp. 433-462, Springer Berlin Heidelberg Editor

[19] S. Vitabile, V. Conti, M. Collotta, G. Scatà, S. Andolina, A. Gentile, F. Sorbello, “A Real-Time Network Architecture for Biometric Data Delivery in Ambient Intelligent”, Journal of Ambient Intelligence and Humanized Computing (AIHC), © Springer-Verlag Editor, June 2013, Vol. 4, Issue 3, pp. 303-321, Print ISSN 1868-5137, Online ISSN 1868-5145, DOI: 10.1007/s12652-011-0104-9

[20] M. Collotta, V. Conti, G. Scatà, G. Pau, S. Vitabile, "Smart Wireless Sensor Networks and Biometric Authentication for Real Time Traffic Light Junctions Management", International Journal of Intelligent Information and Database Systems, Copyright © 2013 Inderscience Enterprises Ltd., ISSN online: 1751-5866, ISSN print: 1751-5858, Vol. 7, No. 5, 2013, pp. 454-478, DOI 10.1504/IJIIDS.2013.056392

[21] S. Vitabile, V. Conti, G. Lentini, F., Sorbello, “An Intelligent Sensor for Fingerprint Recognition”, Proc. of on International Conference on Embedded and Ubiquitous Computing (EUC-05), Lecture Note in Computer Science (LNCS), Springer-Verlag, vol. 3824, pp. 27-36, ISBN 3-540-30807-5, 2005, DOI: 10.1007/11596356_6