Applicazione di Hardware Metamorfico per approccio esplorativo di progettazione di Sistemi Embedded

Enrico Giordano

1 Introduzione

Il flusso di progettazione standard di un sistema embedded si divide in tre step: analisi, progettazione e sintesi; questo flusso mira ad avere un sistema specifico per una situazione particolare e alla sintesi finale di tale sistema il meno generale possibile, in modo da renderlo massimamente performante per quel particolare ambiente. Il flusso standard prevede un particolare sforzo durante l'analisi, in particolare l'analisi ambientale, in quanto è necessario capire esattamente il contesto in cui verrà calato il sistema.

Con questa tesi, si vuole proporre un approccio alternativo il cui mezzo è l'utilizzo di hardware metamorfico e quindi di una particolare tecnica di progettazione basata sulla "configurabilità parziale" della tecnologia delle FPGA. Usando questo mezzo, si vuole progettare un sistema più generico possibile e, calato in un contesto particolare, sarà in grado di modificarsi in base alle esigenze del luogo, per poi generare un sistema embedded autoformato e specifico per l'ambiente. Basandosi sul concetto di "evoluzione", secondo cui un oggetto si modifica in base all'ambiente, si otterrà per ogni ambiente un particolare sistema embedded. Estraendo poi dalla FPGA il codice ottenuto durante l'evoluzione, si potrà ottenere il codice finale e specifico per una ASIC, che quindi non verrà più modificato.

Si presenteranno quindi, a livello descrittivo, le diverse tecniche di configurabilità parziale, introducendo poi un modello di progettazione che mira ad un approccio standardizzato per i futuri sviluppi di questa idea, per poi provare il tutto su un caso di studio reale.

2 Tecnologia attuale del silicio

Attualmente esistono 3 tecnologie differenti riguardo l'utilizzo del silicio per scopi computazionali: Hardware General Purpoise, Hardware Embedded e Hardware Riprogrammabile. Queste 3 tipologie differiscono sia per la composizione, sia per la progettazione ma anche per il settore di utilizzo.

2.1 Hardware General Purpoise

Questo tipo di Hardware lo si può trovare in processori e microprocessori utilizzati in strumenti di calcolo generici, come PC, moderni telefoni e televisori. La caratteristica

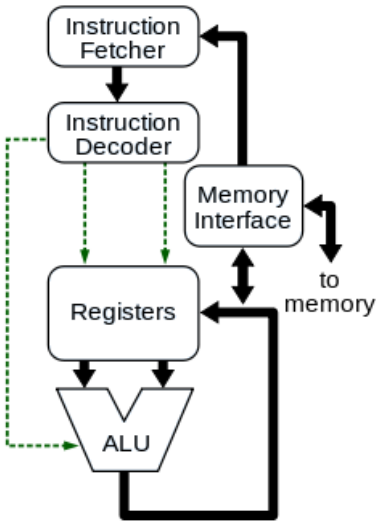

Figura 1: Architettura di una CPU

peculiare di questi è la generalità: non essendo stati progettati per un sistema specifico, devono permettere di eseguire più operazioni possibili ed essere riprogrammati a piacimento, al limite delle loro potenzialità.

Questi sono composti da un'architettura standard, rappresentata nella figura 1, e viene implementata in ogni CPU, ovvero ciò che implementa questa tecnologia.

Il ciclo di vita di questo hardware è contraddistinto da 3 fasi: fetch, decode, exec; ogni fase è contraddistinta da un insieme di componenti interni utilizzati. La fase di "fetch" consiste nel caricare l'istruzione corrente per essere eseguita; la fase di "decode" fa in modo che tutti i valori siano disponibili per il calcolo e che l'istruzione possa essere eseguita correttamente impostando i registri interni; la fase di "exec" permette, tramite i componenti di memoria e di calcolo (registri e ALU), di eseguire l'istruzione.

Questo tipo di hardware è molto utile per implementare sistemi generici: la filosofia alla base di questa tecnologia è l'utilizzo più generico possibile, mettendo in secondo piano l'ottimizzazione e il consumo, oltre al costo che cerca di stare relativamente nelle politiche di mercato.

2.2 Hardware Embedded

Questo tipo di Hardware è contraddistinto dalla specificità dei suoi componenti interni in base al suo utilizzo; infatti si cerca di utilizzare questo hardware in ambienti o condizioni specifiche, in cui sono richieste operazioni particolari ma soprattutto è necessario ridurre i costi e il consumo dell'applicazione finale. La tecnologia che rappresenta questo tipo di hardware è una ASIC e difficilmente si trova un'architettura standard, proprio perché l'architettura è specifica per un certo comportamento. Esistono in commercio dei sistemi embedded basati su microprocessori RISC, come ad esempio i Cortex-M della ARM, che

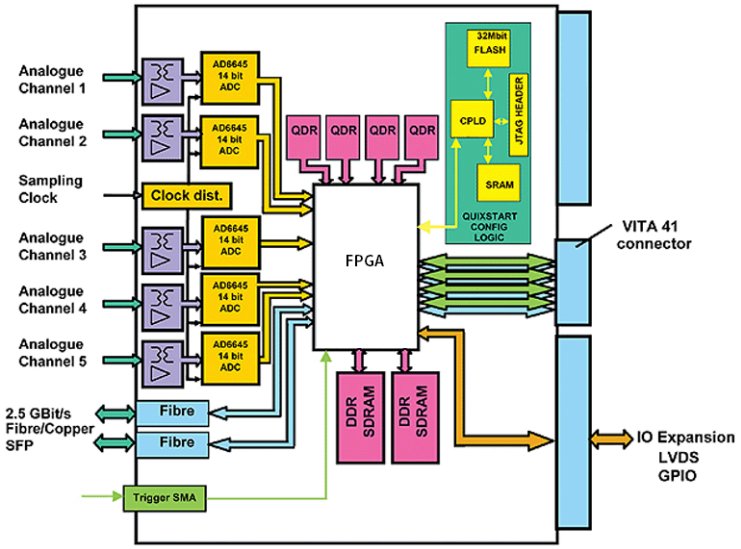

Figura 2: Architettura di una FPGA

hanno un'architettura molto simile a quella delle CPU. In generale comunque ogni ASIC è diversa, in quanto viene mappato su silicio uno specifico algoritmo o un'architettura più complessa.

Un' ASIC deve costare il meno possibile e deve consumare il meno possibile, in base alle esigenze richieste; ovviamente potrà eseguire solo il compito per cui è stata progettata; ha quindi una filosofia di progettazione opposta a quella dell'Hardware General Purpoise.

2.3 Hardware Riprogrammabile

L'ultimo tipo di Hardware è quello riprogrammabile, ovvero permette di essere configurato a livello di celle di memoria più volte per implementare diversi comportamenti. Di solito viene utilizzato sia in ambito embedded, per permettere più libertà di progettazione e più efficienza, ma anche a scopo sperimentale, perchè è utile per testare un sistema in fase di sviluppo.

Un esempio di architettura (dipende molto dal modello che si vuole utilizzare) lo si può osservare nella figura 2: la parte riconfigurabile, denotata con FPGA, consiste in un blocco di celle riconfigurabili, che, in base alla descrizione hardware caricata nella memoria flash, possono cambiare struttura. Attorno alla FPGA sono presenti vari moduli standard che possono servire per il consueto utilizzo in ambito di controlli, ma non sempre sono necessarie, in quanto si possono descrivere all'interno della parte riconfigurabile. Parte necessaria per il funzionamento corretto di una FPGA è la logica di controllo della parte riprogrammabile: deve esserci un hardware dedicato per rimappare questa parte, in modo che il processo di riconfigurazione avvenga più velocemente possibile; questo

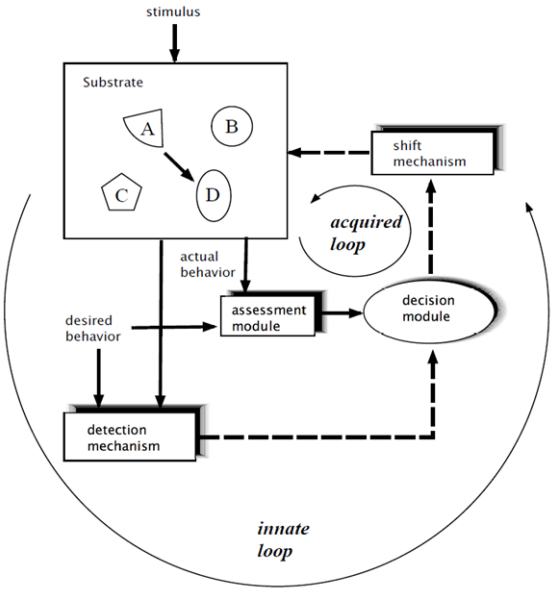

Figura 3: Ciclo evolutivo di un sistema metamorfico

si occuperà quindi di riconfigurare, in base al bitstream che rappresenta l'hardware, la parte riconfigurabile.

In base al suo utilizzo può assumere aspetti di un hardware general purpoise e di un hardware embedded; essendo una tecnologia piuttosto costosa si cerca di utilizzarla il meno possibile, o almeno solo in fase di test.

3 Il concetto di Hardware Metamorfico

Il concetto di "Hardware Metamorfico" nasce nei primi anni del 1990, per poi perdere parzialmente interesse in quanto veniva presentato come idea, non come tecnologia pratica. Con questo concetto si intende un tipo di Hardware capace di cambiare il proprio comportamento ("evolvere") utilizzando un "metodo" di riconfigurazione: i termini "evolvere" e "metodo" non sono mai stati definiti in maniera standard, l'articolo che cerca di porre chiarezza su tutto questo ambito [1] focalizza l'attenzione sulle diverse sfaccettature del termine "evoluzione", parlando di evoluzione "intrinseca" , "estrinseca" e "mixata", rimandando la definizione ad un articolo scientifico precedente [2].

Questo tipo di Hardware è tipico del mondo naturale, in quanto tutti gli oggetti e gli esseri viventi sono in grado di modificarsi in base all'ambiente che li circonda.

Il ciclo di vita di un Hardware Metamorfico può essere rappresentato come nella figura 3: il superciclo che compone questo sistema, chiamato "ciclo innato", ha un sotto-ciclo di acquisizione di stimoli, chiamato "ciclo di acquisizione", che, tramite il sottostrato di algoritmi di esecuzione, cambia il funzionamento del sistema in base al "comportamento

attuale" e il "comportamento desiderato"; con questi dati, tramite un modulo di valutazione degli stimoli e un meccanismo di investigazione (per valutare il comportamento desiderato rispetto al comportamento attuale), viene generato dal modulo di "decisione" il risultato che fa modificare l'intero sistema. Il ciclo innato successivamente fa rieseguire questo procedimento, in modo da permettere, potenzialmente all'infinito, la metamorfosi del sistema.

4 Hardware riconfigurabile

Per rendere utile l'hardware metamorfico, è necessario trovare una tecnologia che lo rappresenti, in modo da rendere concreto l'utilizzo di tale idea. Per renderla concreta, è necessaria la presenza di un oggetto che possa riprodurre sia il comportamento di silicio, sia il comportamento evolutivo dell'hardware metamorfico. Nel 1985, la Xilinx aveva creato le prime FPGA, rilasciando in commercio il primo modello "XC2064", utilizzabile perloppiù per la prototipizzazione di circuiti programmabili. Questa tecnologia sembra prestarsi coerentemente con l'idea di Hardware metamorfico, in quanto è in grado di poter cambiare il proprio comportamento a livello hardware cambiando la propria struttura interna. Il suo carattere riconfigurabile però è stato progettato e utilizzato per la prototipizzazione, non per l'effettivo mutamento di comportamento durante il suo ciclo di vita.

Nei primi anni '90 nasce una tecnica di progettazione chiamata "Riconfigurabilità", da qui poi la tecnologia di applicazione, appunto le FPGA. Questa tecnica consiste nell'utilizzare Hardware con tecnologia riprogrammabile che, tramite la memoria messa a disposizione dentro il sistema, riesce a modificare il suo comportamento. Non è un comportamento appreso completamente dall'ambiente, ma è un comportamento previsto e quindi implementato che va a sostituire un comportamento esistente; il ciclo di vita quindi non sarà composto da un vero e proprio "apprendimento", ma sarà una modifica prevista dal progettista. Questo può sembrare limitante, in quanto non c'è piena libertà da parte del dispositivo, ma non avrebbe senso far prendere iniziativa a questo (per i principi della computabilità, non sarebbe nemmeno possibile).

Quindi, più precisamente, si più parlare di "riconfigurabilità parziale" e "riconfigurabilità run-time": il primo termine pone enfasi su la parziale capacità del dispositivo di modificarsi, in quanto ci sarà una parte progettata che non deve modificarsi ma soprattutto perché il dispositivo non si modificherà completamente, avrà delle modifiche consentite prestabilite; il secondo termine indica il fatto di potersi modificare durante il suo ciclo di vita, cambiando la propria configurazione di porte logiche ottenendo quindi una forma diversa.

4.1 Metodi di riconfigurazione

Essendo la Riconfigurabilità una tecnica di progettazione, prevede vari metodi per essere implementata, che possono essere scelti sia per la tecnologia sia in base alle risorse

disponibili su essa. Questi metodi sono stati proposti dall'Altera [3] e dall'insegnante Dirk Koch , dell'Università di Manchester [4].

La prima tecnica è la "Riconfigurazione parziale dinamica", che consiste nel cambiare una parte della FPGA mentre il rimanente circuito continua il suo ciclo di vita. Per fare ciò, esiste una partizione riconfigurabile e in questa si carica il bitstream che corrisponde al nuovo circuito da una memoria esterna. Per far comunicare la parte statica con quella dinamica, è posto un bridge fisico tra di esse.

Il bridge fisico consiste in un bus, solitamente a 32 bit, che permette lo spostamento di file sintetizzati che rappresentano la nuova configurazione hardware. Questi file si trovano o nella memoria esterna o nella memoria interna, quindi vengono caricati runtime in base a condizioni scelte dal progettista. Solitamente il bus è di tecnologia DMA in modo da essere il più veloce possibile e il trasferimento avviene tramite diversi protocolli gestiti a livello hardware (solitamente il protocollo PCAP a 32 bit, ma dipende dall'architettura ed è trasparente al progettista).

Questa tecnica è molto efficace quando si ha a disposizione poca memoria per memorizzare il bitstream che rappresenta l'hardware e può essere utilizzata in quelle architetture che non dispongono di sufficiente memoria per caricare al completo tutto il sistema descritto.

La seconda tecnica è la "Rilocazione parziale di bitstream", che consiste nello "spostare" i moduli compilati che descrivono l'hardware (che rappresentano la parte riconfigurabile) da un'area della FPGA all'altra con le stesse dimensioni e proprietà. Questa tecnica è più limitante della precendente, un quanto si pone il vincolo di avere due moduli con stesse dimensioni e proprietà, però può essere utile per questioni di ottimizzazione di codice, in quanto i due moduli possono rappresentare la stessa unità ma al loro interno eseguono lo stesso algoritmo in maniera differente.

4.2 Readback

Dal modello "Virtex 6" delle FPGA Xilinx, è possibile leggere il contenuto della FPGA e quindi ottenere la configurazione attuale del sistema. Esistono due tipi di lettura: la "readback verify", che consiste nella lettura di tutte le celle di memoria, e la "readback capture", che permette di leggere le celle di memoria e lo stato dei registri, eseguendo un effettivo dump totale. Questa informazione è di particolare interesse per questo progetto, in quanto permette di ottenere la configurazione ottenuta dopo la metamorfosi, quindi ottenere un nuovo codice. Questa proprietà è stata in parte resa negativa dal punto di vista della sicurezza informatica, in quanto permette di estrapolare codice per poi potenzialmente farne un cattivo uso, però è molto utile per sapere l'andamento della metamorfosi.

4.3 Metodi di reallocazione

La parte cruciale quindi di queste tecniche è proprio la reallocazione dei moduli, in quanto la riconfigurabilità consiste proprio nello "spostare" un modulo da una sezione

attiva a una non attiva. Esistono due modalità di reallocazione: usando dei tools offerti dalla Xilinx e usando il bus.

I tools della Xilinx sono dei software che fungono da middleware per gestire lo spostamento da un'area all'altra e vengono dati in dotazione in base al sistema acquistato. Le direttive del bus sono delle macro da utilizzare durante l'esecuzione del codice per istruire il bus prima di far comunicare i diversi moduli e collegarli. Entrambe le modalità creano un "ponte" tra la zona statica e la zona dinamica della logica programmabile, in modo da interconnetterle per generare una nuova descrizione hardware.

Entrambi i metodi si basano sulla riconfigurazione runtime della parte di logica programmabile tramite lo stesso hardware della FPGA che si occupa della riconfigurazione statica (quindi durante la progettazione da parte del progettista), ma in maniera automatica: il modulo di controllo di riconfigurazione sceglie quale può essere la zona di logica programmabile corretta (la prima che ha abbastanza spazio per contenere la nuova logica) e genera il segnale di riconfigurazione; al momento del segnale, si esegue la procedura di riconfigurazione come da prassi ma il bitstream da caricare non deriva da un programma esterno via JTAG bensì dalla memoria interna. Quindi ciò che viene caricato non viene generato runtime, è già presente all'interno della memoria, ma il comportamento globale del sistema cambia in quanto viene caricata una porzione di bitstream che andrà ad alterare il circuito generato dal bitstream statico (che non cambia mai) e quello dinamico (caricato runtime).

4.4 Aree di reallocazione

Il processo di reallocazione visto finora è riassumibile con la figura 4, in cui si può vedere come la parte di "processing system" (PS) configuri l'informazione da inviare alla parte di "programmable logic" (PL) tramite il bridge; caricato il file che rappresenta la nuova parte sintetizzata di programma dalla memoria, si invia alla parte da configurare PL utilizzando il protocollo del bridge. Le aree precedentemente descritte quindi si trovano all'interno di PL, dove risiede l'effettiva parte logica. Queste aree normalmente possiedono una frammentazione interna ed esterna, in quando il codice non sempre occupa tutti i settori; proprio per questo si può deframmentare e ottenere dei settori vuoti in cui inserire runtime il codice.

Spostare un nuovo settore dentro la parte di logica programmata significa quindi configurare hardware precedentemente inattivo, dando quindi una nuova configurazione hardware e quindi un diverso comportamento del sistema. In base alla grandezza delle aree da spostare, possono coesistere all'interno dell'area PL, ma non avrebbe senso perché potrebbe significare avere hardware non utilizzato e configurato, quindi si spreca silicio.

5 La progettazione

Poichè non esiste un metodo standard per progettare, sviluppare o semplicemente simulare questi sistemi, si vuole creare un nuovo flusso di progettazione per gestire al meglio la fase di sviluppo. Poiché questi sistemi fanno parte di un sottoinsieme dei Sistemi

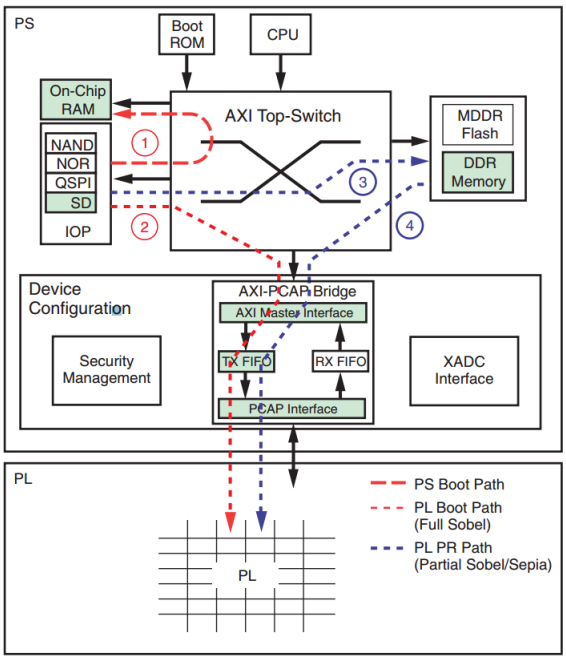

Figura 4: Flusso di riconfigurazione

Embedded generici, si utilizzerà ciò che è noto durante la fase di sviluppo di un normale Sistema Embedded per poi aggiungere ciò che serve per specializzare tale sistema in uno Metamorfico.

Il punto di partenza della progettazione non dovrà essere l'analisi dell'ambiente in cui calare il sistema da progettare, bensì sarà scindere cosa deve essere statico e cosa deve essere dinamico: una business logic, ad esempio, può essere una parte statica, mentre un insieme di range su cui fare misure può essere una parte dinamica perché dipende dall'ambiente. Una volta scelte le due parti, si procede con la progettazione come da prassi di un sistema embedded, progettando contemporaneamente queste parti: in questo modo, durante la progettazione, si può capire se le scelte adottate sono state corrette. Infine si configura l'hardware in modo da prepararlo per la sua futura metamorfosi, impostando la parte riconfigurabile, per poi osservare come cambia durante l'acquisizione delle informazioni.

Il punto di arrivo sarà quello di ottenere diversi hardware per diversi ambienti; a seguito dell'adattamento del sistema quindi si potrà essere in grado di estrarre il codice ottenuto per ottenere diverse ASIC che saranno ottimizzate in maniera automatica per l'ambiente.

Lo scopo di questa tesi sarà comunque capire come sviluppare al meglio la progettazione di questo tipo di sistema, comprendendo il grado di libertà che si può raggiungere per la parte dinamica e capire quale tipo di riconfigurabilità è più conveniente usare, oltre al fatto di essere sicuri che sia conveniente sfruttare la metamorfosi per risparmiare

tempo e denaro durante la progettazione di un sistema embedded.

Riferimenti bibliografici

[1] Garrison W. Greenwood,Senior Member, IEEE and Andy M. Tyrrell,Senior Member, IEEE, "*Metamorphic Systems: A New Model for Adaptive System Design*", 2010

[2] Sekanina, L. ; Brno Univ. of Technol., Brno ; Martinek, T. ; Gajda, Z., *Extrinsic and Intrinsic Evolution of Multifunctional Combinational Modules*, 2006

[3] Altera, *FPGA Run-Time Reconfiguration: Two Approaches*, 2008

[4] Dirk Koch, *Partial Reconfiguration on FPGAs*, 2013