Progetto concorrente di hardware e software

GIOVANNI DE MICHELI
Stanford University, Stanford, CA USA

Il progetto concorrente (*co-progetto* o *co-design*) delle unità fisiche e logiche dei sistemi digitali è un tema di grande attualità. Il co-progetto ha come obiettivo soluzioni che sfruttino al meglio le relazioni tra scelte realizzative hardware e programmazione software. Questo campo è molto vasto, a causa delle diverse applicazioni dei sistemi digitali e delle varie tecniche di realizzazione. In questa rassegna, si considerano alcune problematiche di co-progetto e lo stato dell'arte dei corrispondenti strumenti di progetto.

1. Introduzione

LA MAGGIOR PARTE DEI SISTEMI DIGITALI è programmabile e consiste in unità fisiche (hardware) e logiche (software). Il progetto concorrente di hardware e software (co-progetto o anche *hardware/software co-design*) è una metodologia che mira a raggiungere obiettivi specifici (quali prestazioni, costo, affidabilità, versatilità ecc.) sfruttando la sinergia fra hardware e software [1]. Siccome esistono sistemi digitali con differenti architetture e applicazioni, ne conseguono diversi problemi di co-progetto.

Molti di questi problemi, noti da tempo, sono stati risolti dai progettisti con soluzioni *ad hoc*. Sfortunatamente le dimensioni dei sistemi digitali, misurabili in termini di porte logiche (*gate*) o di istruzioni macchina, sono talmente grandi che il co-progetto manuale è sconsigliabile per la possibilità di errori, la difficoltà di verifica e il tempo necessario al suo compimento. Il progresso tecnologico porta sicuramente all'aumento di tali dimensioni, e quindi a una maggiore necessità di sostituire il progetto manuale (anche ad alto livello) con metodi basati il più possibile su tecniche di sintesi e verifica automatizzate.

Il crescente interesse verso metodologie di co-progetto è oggigiorno dovuto all'introduzione sul mercato di alcuni strumenti di progetto assistito da calcolatore (*computer-aided design - CAD tools*) (per esempio co-simulatori) e al possibile arrivo in tempi brevi di altri strumenti (ad esempio programmi di co-sintesi). A causa della feroce competitività nel mercato dei sistemi digitali, è probabile che gli strumenti di co-progetto giocheranno un ruolo chiave nello sviluppo di detti sistemi e che quelli di maggior successo e valore (misurato in prestazioni/costo) saranno interamente (o quasi) progettati con l'ausilio di strumenti CAD.

Le previsioni di crescita annuale composta del volume di vendite di strumenti di progetto a livello sistema (co-progetto incluso) sono attorno al 34%, secondo Dataquest, per il quinquennio 1993-1998 (Figura 1). Questo dato trova corrispondenza nella crescita delle vendite di componenti elettronici, e in particolare di microcontrollori e circuiti DSP (*digital signal processor*) per l'elaborazione di segnali digitali (Figura 2).

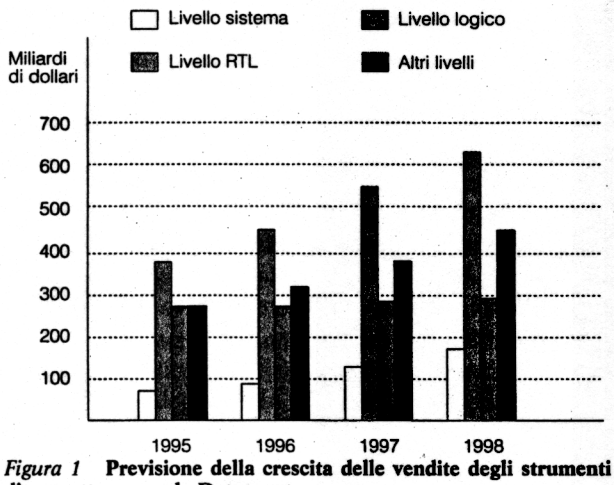


Figura 1 **Previsione della crescita delle vendite degli strumenti di progetto, secondo Dataquest.**

Il miglioramento continuo della tecnologia dei semiconduttori permette nuovi approcci alla progettazione digitale. L'uso di dispositivi con geometrie submicrometriche porta ad alti livelli d'integrazione e ad alti costi di fabbricazione. Diventa pertanto necessario ammortizzare il co-

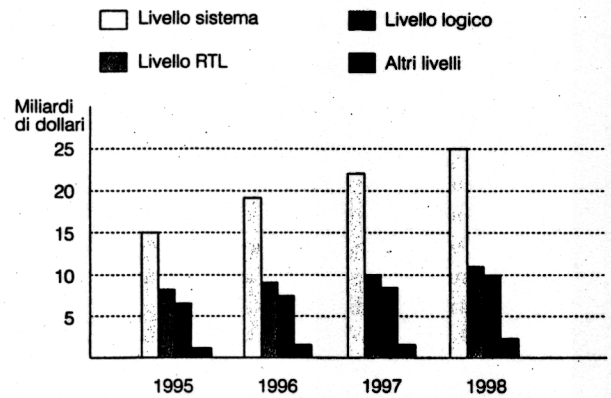


Figura 2 **Previsione della crescita delle vendite di componenti elettronici, secondo Dataquest.**

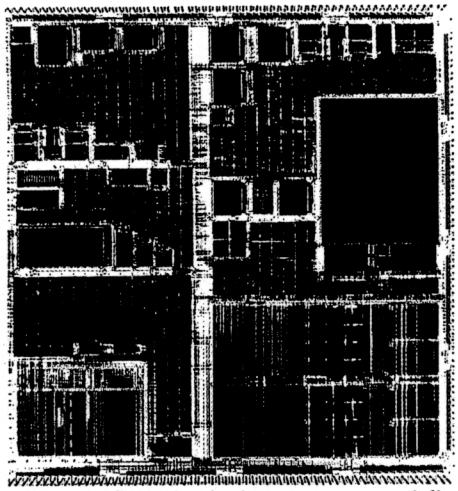


Figura 3 **Esempio di un circuito integrato con noccioli programmabili. In alto a destra il nocciolo VC, basato sul processore MIPS-X. In alto a sinistra il processore VP+ DSP, sopra la memoria.** (*Per cortesia di* Integrated Information Technology).

sto di un progetto hardware attraverso un grosso volume di produzione. A questo punto, il software può essere visto come una maniera di differenziare prodotti basati sullo stesso hardware. Per esempio, esistono oggi parecchi circuiti integrati che contengo uno o più noccioli (*core*), macrocelle che realizzano microprocessori o DSP e le cui funzioni vengono dedicate a un'applicazione specifica mediante appositi programmi in memoria locale (Figura 3 [7]). L'uso di noccioli specifici permette di riciclare vari livelli di software, ad esempio sistemi operativi o software integrato. In generale, il riciclaggio (*re-use*) di hardware e software, è spesso un fattore decisivo per il contenimento dei costi.

2. Aree d'applicazione dei sistemi digitali e corrispondenti problemi di co-progetto

Sebbene la nozione di sistema digitale sia alquanto generica, è possibile fare un semplice raggruppamento in tre classi, con lo scopo di definire meglio i relativi problemi di co-progetto. In linea di massima, si può distinguere tra:

- sistemi di calcolo generico,
- sistemi di calcolo e controllo dedicati,
- emulatori e prototipi.

A ciascuna di queste aree corrisponde una serie di problemi di co-progetto, differenti negli obiettivi e nei metodi realizzativi.

2.1. Sistemi di calcolo generali e microprocessori

I sistemi di calcolo generali includono i calcolatori tradizionali di tutte le dimensioni (dai portatili ai super-calcolatori) che sono programmabili dall'utente. In questo caso, sono disponibili compilatori per vari linguaggi e pertanto la macchina può eseguire funzioni generiche secondo i programmi sviluppati o acquistati dall'utente.

Le funzioni principali di ciascun calcolatore sono espletate da uno o più microprocessori ISP (*instruction-set processor*). L'obiettivo principale di progetto di un ISP è quello di raggiungere le massime prestazioni ottenibili con la tecnologia di fabbricazione corrente, e ciò necessita il progetto concorrente di hardware (processore) e software (compilatore) perché le prestazioni sono misurate dalla velocità d'esecuzione di programmi applicativi [1,10].

Il progetto di un compilatore deve cominciare appena l'insieme di istruzioni è definito e in parallelo col progetto circuitale. Infatti, l'uso del compilatore è necessario per valutare se la scelta delle istruzioni e l'organizzazione del processore permettono di raggiungere le prestazioni richieste.

L'organizzazione dei microprocessori più recenti è basata sull'esecuzione parallela di vari flussi di istruzioni, l'uso di *deep pipeline* e di vari livelli di memoria locale (*cache*) [10]. Tipici problemi di co-progetto sono i seguenti:

- la scelta (fatta dal compilatore) delle istruzioni da eseguire in parallelo, basata sulla rapidità d'esecuzione dei blocchi funzionali hardware;
- il dimensionamento delle *cache* e la selezione dell'algoritmo di validazione dei dati, che deve adattarsi alle dimensioni e alla velocità della memoria. Il controllo del pipeline mediante strutture hardware e/o tecniche di riordinamento delle istruzioni [11].

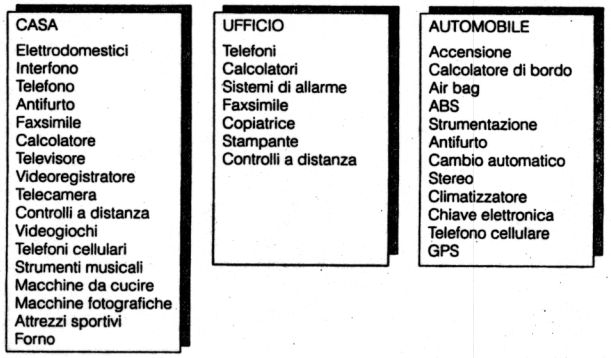
Strumenti di co-progetto per l'ottimizzazione e il controllo di cache e pipeline sono ancora allo stadio di ricerca, mentre co-simulatori vengono usati correntemente per validare le scelte dei progettisti.

2.2. Sistemi dedicati

I sistemi dedicati (*embedded system*) sono progettati per eseguire funzioni specifiche. L'utente non può programmare il sistema, o lo può fare solo in parte. Il sistema viene già fornito con programmi dedicati (embedded software). Questi sistemi possono svolgere funzione di controllo (ad esempio controllo di processi industriali), di elaborazione dell'informazione (ad esempio reti di telecomunicazioni), o entrambi. Il mercato industriale delle applicazioni digitali dedicate è in fortissima espansione. È sufficiente pensare a settori quali l'elettronica di bordo per automobili, aeroplani e imbarcazioni, il controllo dei robot per l'industria manifatturiera, l'elettronica civile (*consumer electronics*), i sistemi di telecomunicazioni e quelli per la gestione e difesa del territorio [1,6,8,14] (Figura 4).

I sistemi dedicati usano microprocessori ISP, processori DSP per l'eleborazione dei segnali, microcontrollori e/o circuiti dedicati con vari gradi di programmabilità (vedi sezione 3). I sistemi di controllo si interfacciano all'ambiente esterno mediante sensori e attuatori. A volte alcuni segnali d'ingresso/uscita sono analogici e pertanto il circuito stesso provvede alla conversione analogico/digitale e viceversa (Figura 5).

Molti sistemi dedicati sono usati per applicazioni in tempo reale. Di conseguenza le funzioni del sistema possono essere sottoposte a vincoli temporali, e sono quindi spesso presenti, all'interno di questi sistemi, dei circuiti temporizzatori. La sicurezza e l'affidabilità dei sistemi di controllo sono fattori molto importanti. Pertanto la validazione dei parametri di progetto e della sua realizzazione è un compito necessario e delicato. Il problema della validazione è complesso, a causa dei vincoli temporali da soddisfare e della presenza nel progetto di componenti hardware e software.



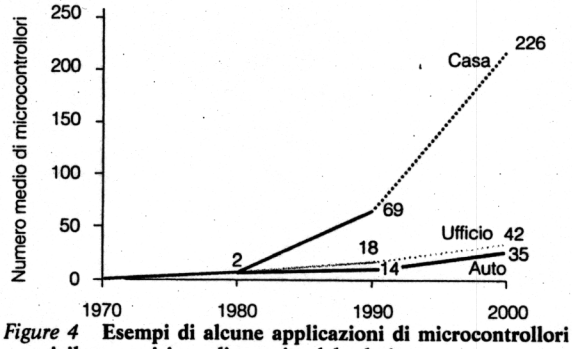


Figure 4 **Esempi di alcune applicazioni di microcontrollori per uso civile e previsione di crescita del relativo mercato.** (Secondo *Electronic News*)

La validazione di sistemi dedicati può essere effettuata mediante simulazione concorrente (co-simulazione) delle componenti hardware e software o mediante metodi di verifica formale. Tali metodi verificano la congruenza di modelli del progetto (ad esempio specifiche e realizzazione) e/o dimostrano alcune proprietà (ad esempio, mancanza di condizioni di blocco o *deadlock*) [3].
Un altro importante problema di co-progetto è la sintesi di sistemi hardware/software a partire da specifiche ad alto livello. Il processo di sintesi, assistito da calcolatore, ha come scopo la generazione delle descrizioni dettagliate (porte logiche o maschere) del circuito hardware, dei programmi (in linguaggi ad alto livello, assemblatore e/o macchina) che realizzano le componenti software e delle interfacce tra i vari blocchi funzionali. Strumenti e metodologie per la progettazione concorrente di sistemi dedicati sono descritti nell'articolo di Gaetano Borriello in questo stesso fascicolo.

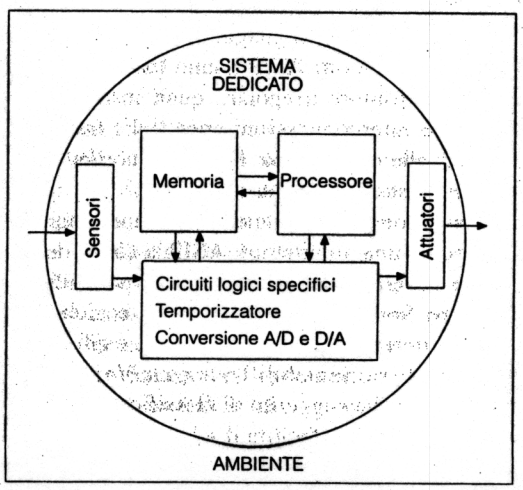


Figura 5 **Schema delle parti essenziali di un controllore dedicato.**

2.3. Emulatori e prototipi

L'arrivo e l'affermazione sul mercato di tecnologie FPGA (*field-programmable gate array*) hanno reso meno netta la distinzione tra hardware e software perché ora un circuito integrato può essere configurato dopo la sua produzione e vendita, e in particolare, una volta inserito in un sistema digitale. In questo articolo, si considera la tecnologia FPGA [12] ove le interconnessioni logiche sono programmate mediante valori mantenuti in una memoria locale. Pertanto, tali circuiti permettono il progetto di sistemi riconfigurabili anche durante l'esecuzione delle proprie funzioni.
Emulatori e prototipi sono basati sulla tecnologia FPGA [1]. L'unità fisica (hardware) è configurata mediante sistemi di sintesi [5], che traducono specifiche ad alto livello nelle interconnessioni tra le porte logiche. È importante notare che questo processo è ripetibile. Emulatori hardware sono usati per velocizzare l'esecuzione di programmi software. Pertanto, questi emulatori possono essere considerati come co-processori. A differenza di co-processori dedicati, quali quelli numerici, un emulatore hardware può essere configurato in maniera da permettere l'esecuzione in hardware di diversi programmi software.
Uno dei primi esempi di emulatori è la PAM (*progammable active memory*) che consiste in una piastra di FPGA e memorie, interfacciata ad un processore. Sono stati costruiti due modelli di PAM, chiamati PeRLe-0 e PeRLe-1, che differiscono sia nel numero e tipo di componenti FPGA usati, che nella frequenza di funzionamento [13]. La piastra PeRLe-1 è mostrata nella figura 6.
L'uso dell'emulatore è il seguente: il programma software da eseguire viene analizzato e le parti critiche sono estratte e sintetizzate in hardware (la sintesi consiste nel determinare la configurazione delle interconnessioni programmabili degli FPGA). Le parti non critiche del programma vengono invece compilate. L'esecuzione del programma avviene parallelamente sulla piastra programmata e sul processore. Risultati sperimentali mostrano una riduzione del tempo totale di calcolo di uno o due ordini di grandezza.
Un'altra applicazione delle tecniche di emulazione hardware mediante FPGA è la simulazione di sistemi digitali [3]. Queste simulazioni sono spesso molto lunghe da eseguire anche su calcolatori veloci, e inoltre parecchie simulazioni sono necessarie per ciascun ciclo di progetto.

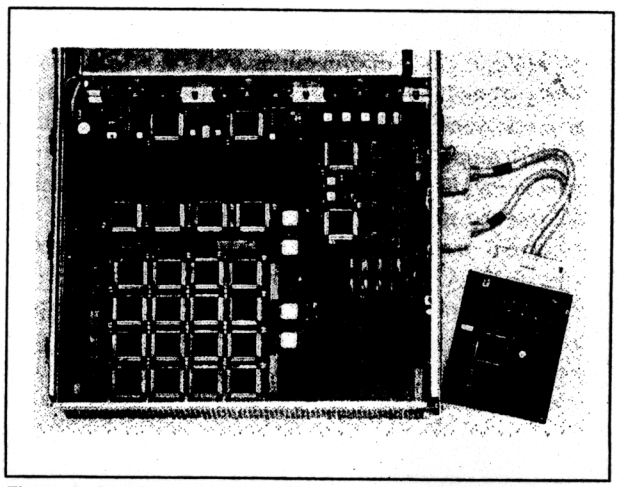


Figura 6 **La piastra PeRLe-1.** (*Per cortesia di* P. Bertin).

L'emulazione riduce notevolmente i tempi d'esecuzione [3]. È interessante osservare come in questo caso uno strumento che usa una tecnica hardware/software è usato a sua volta per il progetto di sistemi hardware/software.
Emulatori hardware sono infine usati come prototipi di sistemi. I prototipi vengono utilizzati dai progettisti come fasi intermedie nella realizzazione di un sistema. L'esecuzione di programmi software sul prototipo viene usata come metodo di validazione del progetto prima di procedere alla sua realizzazione. Pertanto i prototipi servono ad evidenziare problemi in fase di progettazione e a ridurre il rischio di ripetere progetto e realizzazione nel caso di errori. L'uso di emulatori per validare complessi sistemi digitali è vantaggioso perché permette di controllare l'esecuzione di programmi software sulla piattaforma prototipo, quando è ancora possibile cambiare sia hardware che software. Una volta che la configurazione hardware desiderata è stata finalizzata, può essere tradotta nelle specifiche di fabbricazione (maschere) mediante sistemi di sintesi, che accettano come ingressi gli stessi modelli hardware usati dal sistema di emulazione. Questo permette un efficiente ciclo di progetto, che riduce i tempi e le possibilità di errore.

3. Programmabilità dei circuiti e impatto sul software

I sistemi digitali si possono programmare a livello applicativo, istruzione o hardware. Nel primo caso, che è il più restrittivo, si utilizzano le istruzioni del programma applicativo visibili dall'utente. Nel secondo, la programmazione a livello istruzione è ottenuta compilando programmi software, e il grado di programmabilità è definito dall'architettura del processore. La programmazione a livello hardware consiste infine nella riconfigurazione dell'unità fisica. Un esempio classico è la microprogrammazione, cioè la configurazione di un'unità di controllo mediante un microprogramma. Oggi la microprogrammazione è usata per i DSP, ma non per gli ISP basati su architetture con insiemi di istruzioni ridotte RISC (*reduced instruction-set computer*) [10]. Infine, con la riconfigurazione hardware ottenuta mediante tecnologie FPGA, si possono specializzare tutte le componenti di un circuito al fine di eseguire efficientemente applicazioni particolari. Questo caso rappresenta il livello più avanzato di programmabilità.
In generale, il grado di programmabilità di un sistema ne aumenta il campo d'applicazione, ma non necessariamente le prestazioni, se non nel caso di applicazioni specifiche. Nel campo dei microprocessori, le prestazioni più elevate si raggiungono oggi con architetture super-scalari RISC [10] che sono programmate a livello istruzione. Per applicazioni particolari, i circuiti integrati per applicazioni specifiche ASIC (*application-specific integrated circuit*) con programmabilità limitata, danno spesso le prestazioni più elevate con il minimo consumo di potenza. Ciascun ASIC richiede però un progetto particolare, che può essere costoso, se non è prodotto in largo volume. In aggiunta, un ASIC è utilizzabile solo per soddisfare le specifiche per cui è stato progettato, e le specifiche, a volte, evolvono col tempo. Ovviamente esistono soluzioni intermedie tra i processori e gli ASIC, ad esempio ASIC programmabili o contenenti "noccioli programmabili".
In alcuni settori, quali quello dell'elaborazione dell'informazione per telecomunicazioni, si usano processori programmabili a livello istruzione con architettura ASIP (*application-specific instruction set processor*) dedicata a un'applicazione specifica [1,2]. La ragione d'essere degli ASIP deriva dal fatto che essi sono spesso utilizzati all'interno di sistemi dedicati che eseguono programmi con particolari profili di istruzioni. L'insieme delle istruzioni e la struttura degli ASIP sono scelti in maniera da privilegiare le istruzioni più frequenti. A questo scopo si utilizzano di solito particolari insiemi di registri e di connessioni tra i registri stessi, i *bus* e le altre unità funzionali, scelti in modo da privilegiare il tipo di calcolo previsto.
Gli ASIP possono essere considerati come soluzioni intermedie tra ISP e ASIC: gli ASIP sono più versatili degli ASIC, ma meno degli ISP. D'altro canto, le prestazioni di un ASIP misurate su funzioni dedicate possono essere più alte di quelle ottenibili con un ISP, ma inferiori a quelle di un ASIC. La potenza dissipata da un ASIP è di solito inferiore a quella di un ISP, ma è superiore a quella di un ASIC. Nel caso che un ASIP abbia molteplici applicazioni, il costo della progettazione e della fabbricazione può essere ammortizzato su un volume più grande di quello di un ASIC. In aggiunta, cambiamenti di specifiche di progetto possono essere gestiti più facilmente nel caso di un ASIP, quando è possibile farlo riprogrammando il software. Sfortunatamente gli ASIP richiedono però un compilatore specifico e il costo del loro sviluppo si deve aggiungere al costo del progetto hardware.
La compilazione di programmi per gli ASIP ha requisiti diversi da quella per gli ISP usati nei normali calcolatori. In primo luogo, gli ASIP eseguono programmi sviluppati e compilati dal produttore e non dall'utente. Pertanto il tempo di compilazione ha importanza relativa e speciali tecniche di ottimizzazione del programma macchina possono essere applicate senza preoccuparsi di lunghi tempi di compilazione. In secondo luogo, i sistemi dedicati devono spesso soddisfare vincoli d'esecuzione in tempo reale. Pertanto il programma macchina non solo deve essere compatto, ma deve anche garantire specifici tempi di esecuzione. Al momento, molti programmatori usano ancora linguaggi assemblatori, per essere sicuri di soddisfare i vincoli temporali. Un obiettivo principale dello sviluppo di compilatori per applicazioni specifiche è quello di permettere l' uso di linguaggi ad alto livello come il C e il C++. Infine, molti ASIP hanno insiemi di istruzioni particolari e strutture irregolari, quali insiemi di registri eterogenei e interconnessioni specifiche tra le unità di calcolo e quelle di memoria. Pertanto i compilatori devono gestire queste peculiarità.
Ovviamente non è pensabile di progettare compilatori diversi per ciascuna architettura ASIP, a causa del numero di ASIP e del tempo e lavoro necessari per sviluppare un compilatore. Sono state proposte varie tecniche per adattare compilatori a differenti architetture e tali compilatori sono chiamati riorientabili (*retargetable compiler*) [2]. Peter Marwedel ha suggerito di classificare questi compilatori a seconda della facilità d'adattamento alle varie architetture [2]. Un compilatore portabile (*portable*) è un compilatore che può essere adattato riscrivendo una sua

parte. Anche se questo compito non è indifferente, è di certo preferibile allo sviluppo ex novo di un compilatore. Un compilatore-compilatore (*compiler-compiler*) è un programma che genera un compilatore a partire da una descrizione dell'architettura. Pertanto generare un nuovo compilatore richiede di specificare l'architettura nel formalismo richiesto e di eseguire il compilatore-compilatore. Infine, un compilatore indipendente dall'architettura (*machine independent*) è in grado di generare il programma macchina per diverse architetture.

Di solito i compilatori hanno un ingresso dipendente dal linguaggio di programmazione, un generatore/ottimizzatore di una forma intermedia, e un generatore di programma macchina (*code generator*). La compilazione riorientabile differisce da quella usuale nella fase di generazione del programma macchina. In questa fase, si svolgono tre compiti principali e interdipendenti:

- la selezione delle istruzioni,
- l'allocazione dei registri;
- l'ordinamento (*scheduling*) delle istruzioni.

Questi tre compiti sono interdipendenti. La selezione delle istruzioni deve sfruttare indirizzi particolari di operandi e risultati. L'allocazione dei registri deve tenere in conto la loro struttura eterogenea e ridurre il ricorso a memorie non locali. L'ordinamento delle istruzioni deve soddisfare i vincoli temporali. Molti ASIP hanno due o più flussi paralleli di istruzioni e sono microprogrammati. Quindi un compito particolare del compilatore è quello di allineare l'esecuzione delle microistruzioni, codificarle e ottenere un microprogramma macchina compatto.

I programmi (in linguaggi ad alto livello) da compilare sono a volte generati da programmi di sintesi. La sintesi del software è necessaria quando le funzioni software sono specificate con formalismi diversi dai linguaggi di programmazione. Un esempio è dato dall'uso di macchine a stati finiti per rappresentare sistemi hardware/software, con lo scopo di applicare tecniche di verifica formale. La sintesi del software corrisponde allora alla traduzione e ottimizzazione dei modelli a stati finiti in linguaggi quali il C [3]. Un altro esempio di sintesi del software è relativo all'uso di strumenti di partizionamento automatico di specifiche di sistema in componenti hardware e software [4,9]. I modelli delle componenti software prodotte dal partizionatore vengono tradotti automaticamente in programmi compilabili.

4. Conclusioni

Il progresso nel settore dei sistemi digitali dipenderà in parte dal livello di supporto degli strumenti di progettazione. In particolare, strumenti di co-progetto hardware/software saranno molto utili ad ottenere soluzioni che sfruttino al meglio la tecnologia dei semiconduttori e la disponibilità di componenti, noccioli e programmi.

Al momento esistono pochi strumenti di co-progetto disponibili sul mercato (ad esempio co-simulatori) ma altri sono oggetto di ricerca e sviluppo con grandissimo interesse. È altamente probabile che l'impatto dell'uso degli strumenti di co-progetto sui sistemi digitali sarà più profondo di quello degli strumenti di progetto dei circuiti integrati stessi.

BIBLIOGRAFIA

Diversi aspetti di co-progetto sono descritti da specialisti del settore nel volume [1] che riassume i temi di un corso sponsorizzato dalla NATO. Risultati recenti sulla compilazione si trovano in [2]. Alcuni aspetti di co-progetto sono descritti in numeri speciali di riviste [3,4].

[1] De Micheli G, Sami M G (ed.): *Hardware/Software Co-design.* Kluwer, 1996.

[2] Marwedel P, Goossens G (ed.): *Code Generators for Embedded Processors.* Kluwer, 1995.

[3] Hardware/Software Co-design (Special issue). *IEEE Micro*, agosto 1994, vol. 14, n. 4.

[4] Codesign: Tying the Hardware-Software Knot (Special issue). *IEEE Design & Test*, settembre 1993, vol. 10, n. 3.

[5] De Micheli G: *Synthesis and Optimization of Digital Circuits.* Mc Graw-Hill, 1994.

[6] De Micheli G, Sami M G, Sciuto D: Progetto di sistemi digitali dedicati. *Automazione e Strumentazione*, febbraio 1995, n. 2, p. 35-39.

[7] Feigel C: Processors Aim at Desktop Video. *Microprocessor Report*, febbraio 1994, vol. 8, n. 2.

[8] Gajski D, Vahid F, Narayan S, Gong J: *Specification and Design of Embedded Systems.* Prentice-Hall, 1994.

[9] Gupta R: *Co-synthesis of Hardware and Software for Digital Embedded Systems.* Kluwer, 1995.

[10] Hennessy J, Patterson D: *Computer Architecture: a Quantitative Approach.* Morgan Kaufmann, 1990.

[11] Huang I, Despain A: High-level Synthesis of Pipelined Instruction Set Processors and back-end Compilers. *Proceedings of DAC,* 1992, p. 135-140.

[12] Trimberger S: A Reprogrammable Gate Array and Applications. *IEEE Proceedings*, luglio 1993, vol. 81, n. 7, p. 1030-1041.

[13] Vuillemin J, Bertin P, Roncin D, Shand M, Touati H, Boucard P: Programmable Active Memories: Reconfigurable Systems Come of Age. *IEEE Transactions on VLSI*, marzo 1996, vol. 4, n. 1 p. 56-69.

[14] Wolf W: Hardware-Software Co-Design of Embedded Systems. *Proceedings of IEEE*, July 1994, vol. 82, n.7, p. 967-989.

GIOVANNI DE MICHELI è docente all'Università di Stanford in California, dove si occupa di progettazione di circuiti e sistemi integrati assistita da calcolatore e di tecnologie elettroniche di progetto. Autore di quattro libri ed altre pubblicazioni scientifiche, è Fellow della IEEE, presidente del comitato tecnico della Design Automation Conference (1996-97) e del comitato tecnico ed esecutivo della IEEE International Conference on Computer Design (1988-89).