# Embedded Systems

Introduzione ai Sistemi di Elaborazione
Hardware/Software Codesign

2004

## Outline

- Introduzione ai sistemi di elaborazione
- Hardware/Software Codesign
  - Generalità
  - Architetture di riferimento
  - Modelli di specifica
  - Partizionamento
  - Esplorazione dello spazio di progetto
  - Sintesi delle interfacce
  - Stima

## Introduzione

- Sistema digitale
  - Sistema eterogeneo complesso costituito da una piattaforma hardware (microprocessori, componenti hardware programmabili (es. FPGA) o dedicati (es. ASIC)), livelli applicativi software, interfacce, trasduttori e attuatori.
- Classificazione dei sistemi digitali

| | |
|---|---|
| **Dominio di Applicazione**<br>Computer<br>Telecom<br>Automotive | **Tipo di sistema**<br>Sistemi *General-Purpose*<br>Sistemi *Embedded* |
| ***Programmabilità***<br>Sezione software<br>A livello di applicazione e a livello di istruzione<br>Sezione hardware | **Implementazione**<br>Tecnologia<br>Stile di progetto<br>Livello di integrazione |



## Introduzione

- Tipo di sistema e dominio di applicazione
  - Il tipo di sistema, il dominio di applicazione e la dimensione del sistema sono strettamente correlati.
  - In generale, i sistemi possono essere:
    - *General-purpose*.
      - Personal computer, Workstation e Mainframe
    - *Embedded* (dedicati) e sistemi di controllo.
      - Applicazioni per l'*Automotive* e l'aviazione
      - Controllo di applicazioni domestiche
      - Controllo di impianti industriali
      - Robot

## Introduzione

- *Programmabilità*
  - Sistemi *General-purpose*:
    - L'utente finale e l'operatore hanno accesso a tutte le componenti software del sistema.
  - Sistemi *Embedded* e di controllo:
    - Il produttore programma il sistema; l'utente finale può intervenire su una sezione molto limitata della componente software.
  - Sezione software
    - A livello di applicazione
    - A livello di istruzione (sia inter-istruzione sia intra-istruzione)
      - Programmi
      - Architettura dell'insieme delle Istruzioni (ISA)
  - Sezione hardware
    - Riconfigurabilità (tecnologia *field-programmable*)



## Introduzione

- Implementazione
  - Tecnologia
    - CMOS, BiCMOS, GaAs, ...
  - Stile di progetto
    - Full-custom
    - Semi-custom
    - PLD e FPGA
  - Modalità operative
    - Circuiti digitali sincroni
    - Circuiti digitali asincroni
    - Circuiti analogico/digitali

## Introduzione

- Problemi di progetto dei sistemi digitali (esempi)
  - Problemi di progetto di Sistemi *General-Purpose*
    - Progetto della architettura del processore e del corrispondente compilatore
      - Insieme delle istruzioni (ISA)
      - pipeline
        - » Unità hardware di controllo della pipeline
    - Dimensionamento e controllo della cache
      - Scelta dei parametri e messa a punto
    - Supporto per il progetto di sistemi multiprocessore
    - Inoltre
      - I compilatori devono essere progettati contemporaneamente allo sviluppo dell'architettura hardware.
      - Sono richieste applicazioni software per verificare le prestazioni dell'hardware (Co-simulazione hardware e software).



## Introduzione

- Problemi di progetto di Sistemi *Embedded*
  - Sistemi dedicati per la computazione ed il controllo
    - Reattivi
      - » Il sistema risponde agli stimoli prodotti dall'ambiente
    - Real-time
      - » Vincoli temporali (hard e soft) sulla evoluzione dei task
  - Progetto di sezioni hardware e software dedicate che interagiscono fra loro
    - Hardware
      - » PLD-FPGA-ASIC, ASIP, DSP…
    - Software
      - » Applicativi
      - » Sistemi operativi *Special-purpose*
      - » Driver per dispositivi periferici.

## Introduzione

- In generale, gli obiettivi nella realizzazione dei sistemi di elaborazione nel rispetto dei vincoli sono:
  - Massimizzare le prestazioni del sistema
    - Velocità, consumo di potenza (energia), affidabilità…
  - Ridurre i costi
    - Ridurre le parti da realizzare e la loro dimensione
    - Estendere il tempo di vita dei componenti hardware utilizzando elementi riprogrammabili
    - Facilitare il *debugging* e il *testing* a livello di sistema

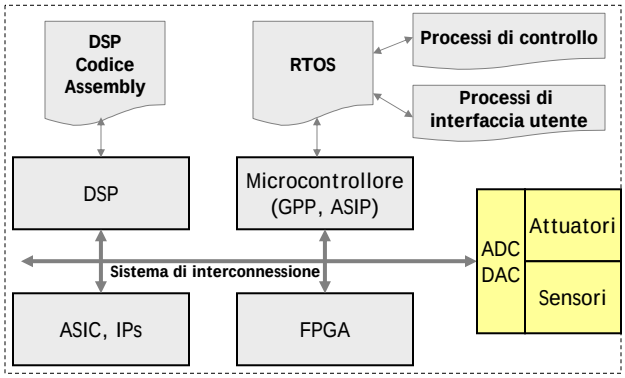


## Introduzione

- Tendenze nella realizzazione dei sistemi
  - Aumentare la complessità delle applicazioni anche per prodotti standard con ampio volume di produzione
    - Ereditare delle funzionalità facilitando la crescita del prodotto
      - Estendere in modo semplice le capacità del sistema
    - Sottostare a requisiti di flessibilità
    - Miscelare differenti tecnologie e tipi di processori
  - Realizzare sistemi sempre più complessi
    - Utilizzare differenti tecnologie, tipi di processori e stili di progetto
    - Produrre sistemi su di un singolo chip (*System-on-a-chip*) combinando componenti provenienti da differenti sorgenti (IP market)
  - Sottostare ad un ampio insieme di vincoli e direttive
  - Ridurre e sovrapporre cicli di progetto
    - Ridurre il *Time-to-market* abbreviando le fasi di progetto e simulazione

## Introduzione

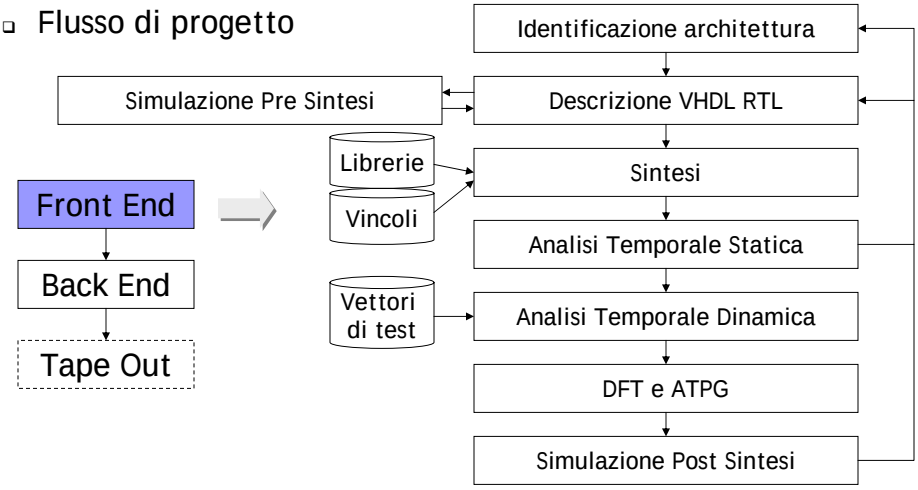
- *System-on-a-chip*
  - Possibile implementazione



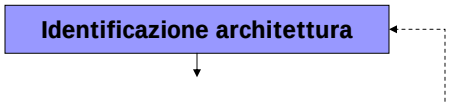



## Introduzione

- Flusso di progetto



## Introduzione




- L'identificazione della architettura è una attività creativa basata sulla esperienza e sensibilità del progettista.
  - Il progettista deve decidere l'architettura obiettivo che soddisfa le specifiche desiderate.
- Nel caso di progetti complessi le architetture possono essere identificate svolgendo simulazioni estensive considerando differenti modelli di architettura.



## Introduzione

- Nella realizzazione dei sistemi digitali complessi:
  - gli obiettivi possono essere perseguiti identificando delle soluzioni che siano un buon bilanciamento tra le componenti hardware e software;
  - gli obiettivi possono essere raggiunti disponendo di:
    - Opportuni formalismi di descrizione del sistema che supportino differenti aspetti e siano indipendenti dal dominio implementativo (hardware e software)
    - Nuovi strumenti di progetto automatico (semi-automatico) che consentano di identificare e sfruttare la sinergia tra componenti hardware e software attraverso una loro realizzazione concorrente.

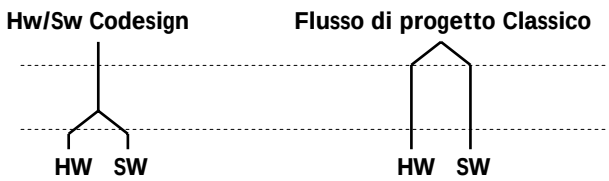
# Hardware/Software Co-design

Generalità

2004

## Hardware/Software Co-design: generalità

- L'*Hardware/software Codesign* è una metodologia di progetto per sistemi costituiti da componenti sia hardware sia software:
  1. Analisi della specifica di sistema per l'identificazione delle componenti hardware e software;
  2. Valutazione delle alternative di progetto.



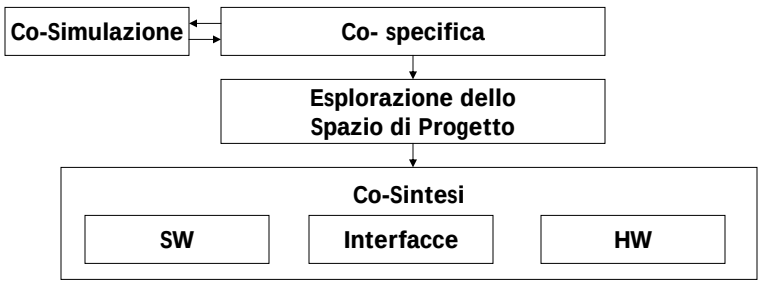
## Hardware/Software Co-design: generalità

- Obiettivi dell'*Hardware/software Codesign*
  - Ottimizzare il processo di progetto
    - Aumentare la produttività riducendo il tempo di progetto del sistema
      - Facilitare il riuso di componenti hardware e software
    - Fornire un ambiente integrato per la sintesi e la validazione delle sezioni hardware e software
  - Consentire la realizzazione di progetti qualitativamente migliori
    - Esplorare differenti alternative di progetto nello spazio definito dalla architettura



## Hardware/Software Co-design: generalità

- Passi del flusso di *Codesign*
  - co-specifica (co-modellazione)
  - co-verifica (co-simulazione)
  - Esplorazione dello spazio di progetto e co-sintesi
  - Sintesi delle interfacce



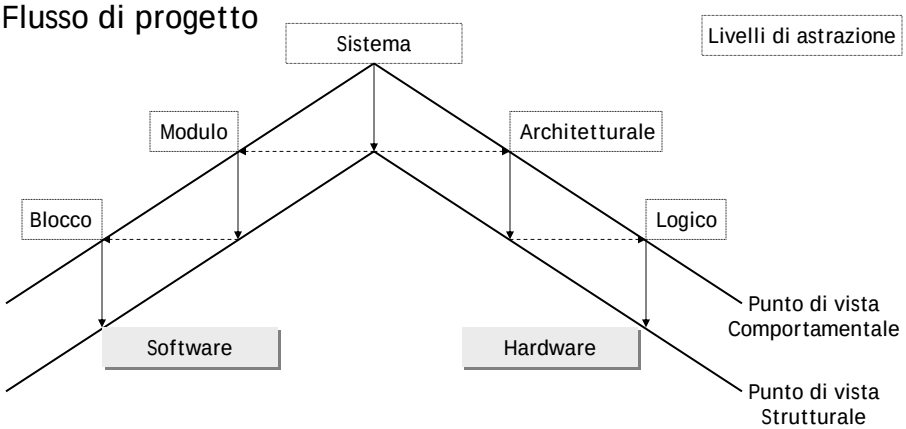
## Hardware/Software Co-design: generalità

- Osservazioni:
  - Progettazione concorrente di sezioni hardware e software di sistemi parzialmente specificati o con specifiche variabili.
  - Strette finestre di *time-to-market* richiedono un processo di progetto *first-time-right*
    - L'identificazione tempestiva di errori sistematici di progetto
    - Prerequisiti: **stimatori affidabili** e progettisti esperti
  - Aumento della produttività attraverso il riuso di componenti e funzioni
    - Richiesta una libreria di funzioni e componenti
    - Problema: migrazione di funzioni tra differenti tecnologie e tra hardware e software.

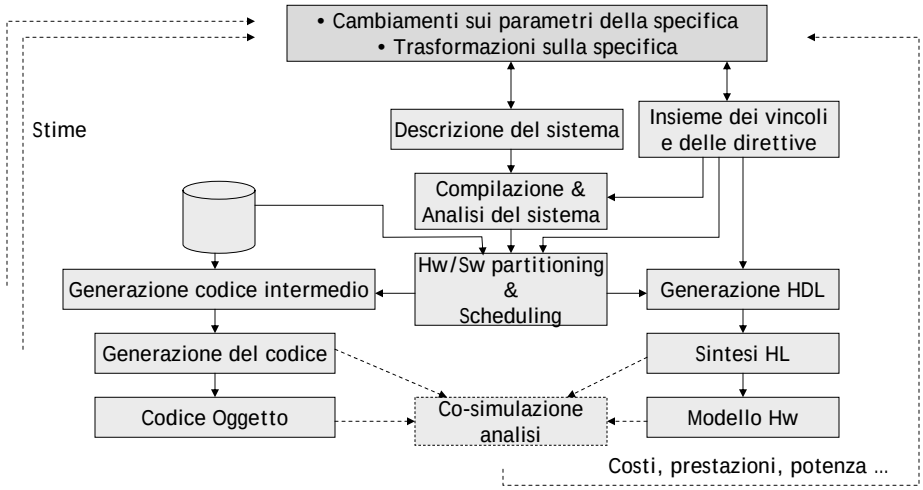


## Hardware/Software Co-design: generalità

- Flusso di progetto

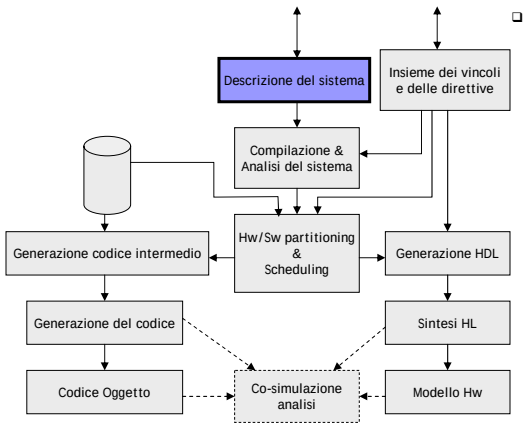


## Hardware/Software Co-design: generalità



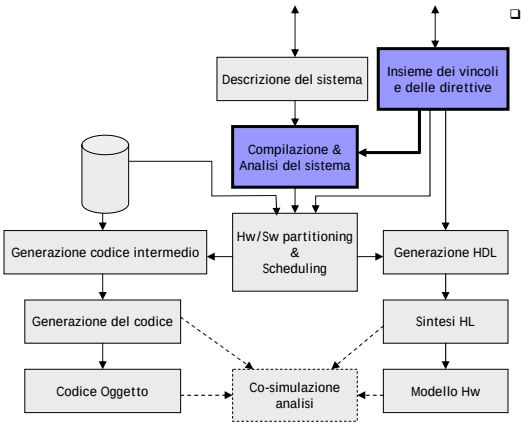



## Hardware/Software Co-design: generalità




- Co-specifica
  - Descrizione del sistema con un linguaggio di specifica che non è specializzato ne verso la sezione software ne verso la sezione hardware.
  - La co-specifica *eseguibile* è utilizzata come base per la validazione a livello di sistema
  - Problemi:
    - Integrazione nel modello astratto di funzioni riusabili e componenti
    - Inclusione nella specifica di vincoli non-funzionali

# Hardware/Software Co-design: generalità

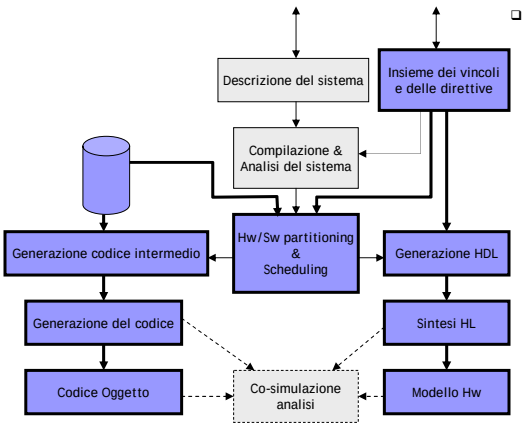



- Co-simulazione
  - Simulazione contemporanea delle sezioni hardware e software
  - La co-simulazione è utilizzata come supporto al processo di integrazione
    - Consente il controllo permanente della consistenza tra hardware e software
  - Problemi:
    - La validazione della integrazione è tanto più costosa quanto più aumenta il livello di dettaglio
      - Riduzione della accuratezza solo per migliorare le prestazioni nella simulazione



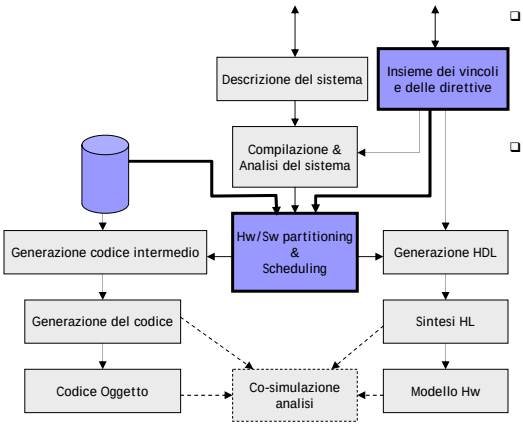


# Hardware/Software Co-design: generalità




- Co-sintesi e ottimizzazione
  - È una fase di sintesi molto complessa costituita da
    - *Hardware/software partitioning* e selezione dei componenti
    - *Hardware/software scheduling*
    - Sintesi della componente software e generazione del codice
    - Sintesi della componente hardware
    - Sintesi delle interfacce e degli elementi di comunicazione
  - Applicazioni
    - Esplorazione dello spazio di progetto
    - Ottimizzazione a livello di sistema

# Hardware/Software Co-design: generalità




- Hw/Sw Partitioning
  - Partizionare la funzionalità del sistema in componenti software e in componenti hardware
- Scheduling
  - Definire i periodi in cui le funzioni devono essere svolte tenendo conto delle precedenze, dei vincoli di tempo, della mutua esclusività tra i processi, della sincronizzazione e comunicazione e della allocazione alle risorse (load balancing nel caso software)





# Hardware/Software Co-design: generalità

- Hw/Sw Partitioning
  - Similarità con il problema della allocazione (distribuzione dei carichi) nella sintesi di alto livello (sistemi operativi real-time)
  - Algoritmi di partizionamento
    - Ottimizzazione stocastica (Simulating Annealing, Algoritmi Genetici) e ricerca locale
    - List scheduling, ILP, ottimizzazione stocastica con ottimizzazione pareto, logica fuzzy, tabu search, …
  - Gli approcci differiscono non solo per la modalità di ricerca ma anche per la granularità del partizionamento e per la architettura obiettivo (variabili controllabili).
    - La granularità degli elementi su cui si focalizza l'attenzione può essere anche variabile.
  - Spesso, il partizionamento è combinato con lo scheduling

## Hardware/Software Co-design: generalità

- Hardware/software scheduling
  - Lo scheduling è una operazione che identifica una relazione tra tempo e processo in esecuzione
  - Scheduling senza prelazione
    - Al termine del servizio di ogni interruzione il controllo viene nuovamente ceduto al processo interrotto.
      - Ridotto sovraccarico dovuto al cambio di contesto
      - Adatto per *real-time lasco* poiché le prestazioni di tempo reale sono lasciate al progettista.
    - Preferito per granularità fine e per applicazioni *data dominated*



## Hardware/Software Co-design: generalità

- Hardware/software scheduling
  - Scheduling con prelazione e scheduling dinamico
    - Al termine del servizio di ogni interruzione il controllo può non essere restituito al processo interrotto.
    - Adatto per sistemi *real time*.
    - Scheduling dinamico
      - La relazione d'ordine d'esecuzione dei processi (priorità dei processi) cambia *run time*.
    - Preferito in sistemi con componenti reattivi
      - Componenti che devono rispondere agli stimoli prodotti dall'ambiente
  - Approcci misti
    - Gruppi di processi all'interno dei quali lo scheduling è statico.

## Hardware/Software Co-design: generalità

- Sintesi delle interfacce
  - Generazione di interfacce tra componenti eterogenei
  - Elementi da considerare
    - Protocolli di comunicazione
      - (non)buffer e (non)bloccante
    - Modalità di comunicazione
      - Interrupt, DMA, porte dedicate, memoria condivisa, …
    - Vincoli
      - Processore, FPGA, Canali di comunicazione, banda passante…
    - Dispositivi dinamicamente riconfigurabili



## Hardware/Software Codesign

Architetture di riferimento

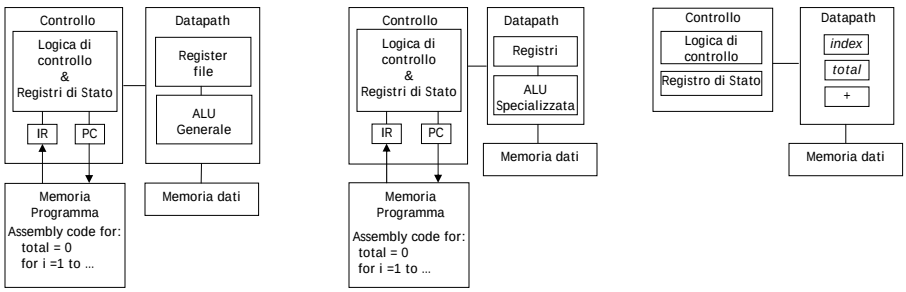
# Introduzione

- L'architettura è determinata da:
  - Aspetti tecnologici
    - Processore
    - Circuiti Integrati
  - Vincoli di progetto
    - Ambito applicativo
      - L'obiettivo principale è lo sviluppo di sistemi soggetti a molti vincoli non funzionali che hanno una forte influenza sia sugli obiettivi di progetto sia sulle architetture
        - » Stretti margini di costo, vincoli temporali stringenti (Real Time), vincoli energetici e di potenza, sicurezza (EMI – Avionics e Automotive), peso e dimensione …
    - Flessibilità
      - La specializzazione non deve compromettere la possibilità di riadattare il dispositivo a nuove specifiche (riuso, cambiamenti dell'ultimo momento…)
  - Requisiti della applicazione



# Tecnologia del Processore

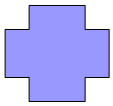
- Architettura del motore computazionale utilizzato per implementare la desiderata funzionalità del sistema
- Il processore non è necessariamente programmabile
  - *Processore* non è sinonimo di *General-Purpose Processor* (GPP)




General-purpose (software) Application-specific Single-purpose (hardware)

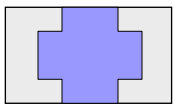
## Tecnologia del Processore

- I processori cambiano nella loro specializzazione in relazione al problema

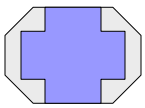


```
...
total = 0
for i = 1 to N  loop
   total += M[i]
end loop
...
```

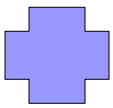
Funzionalità desiderata



General-purpose processor
GPP



Application-specific processor
ASIP

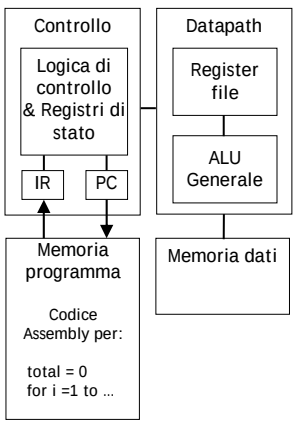


Single-purpose processor
ASIC, FPGA...



## General-purpose processor - GPP

- Dispositivi programmabili utilizzati in una ampia gamma di applicazioni
  - Noti con il termine di microprocessori
- Caratteristiche
  - Memoria programma e dati
    - Congiunte (Von Neuman) o separate (Harvard e Super Harvard)
  - Datapath non specializzato, ampio insieme di registri ad uso generale e ALU non specializzate
- Benefici e svantaggi
  - Time-to-market e costi NRE bassi
  - Flessibilità elevata
  - Tempo di esecuzione difficilmente predicibile
  - Potenza dissipata elevata



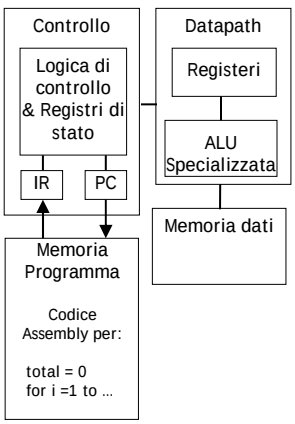
## General-purpose processor - GPP

- Microcontrollori (MCU)
  - CISC, elevata densità di codice e risorse computazionali ridotte
- Processori RISC
  - Ampio Register-file e insieme di istruzioni ortogonali
- DSP
  - Architettura specifica per l'elaborazione di segnali digitali
- Processori per il Multimedia
  - VLIW, alte prestazioni, bassa densità di codice e elevata potenza dissipata.



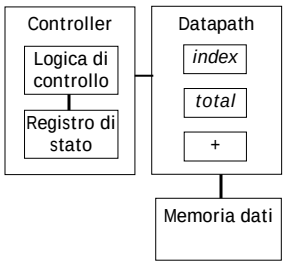
## Application-specific processors - ASIP

- Processori programmabili ottimizzati per una classe particolare di applicazioni che hanno caratteristiche comuni
  - Rappresentano un compromesso tra i processori GPP e i processori specializzati su di una singola applicazione
- Caratteristiche
  - Memoria programma
  - Datapath ottimizzato e specializzato (ampiezza e numero dei registri), ridotto insieme di registri e unità funzionali specializzate
- Benefici
  - Flessibilità buona
  - Prestazioni, dimensioni e potenza buone



## Single-purpose processors - SPP

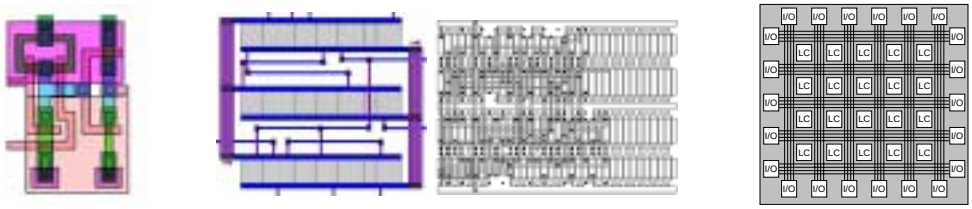
- Circuiti digitali progettati per eseguire esattamente un programma
  - Coprocessori, acceleratori, periferiche…
- Caratteristiche
  - Nessuna memoria programma
  - Contengono solo i componenti necessari per eseguire un singolo programma
- Benefici
  - Flessibilità nulla o molto ridotta
  - Prestazioni, dimensioni e potenza ottime






## Tecnologia dei circuiti integrati

- Modo con cui una implementazione digitale è tradotta in un circuito integrato (IC- Integrated circuit o chip)
- Tre tipi di tecnologie per circuiti integrati
  - Full-custom ASIC
  - Semi-custom ASIC
    - Gate array, *sea of gates* e standard cell
  - PLD (Programmable Logic Device)



## Full-custom ASIC

- Struttura ottimizzata per una particolare funzionalità digitale
  - Dimensione e posizione dei transistor
  - Connessioni dedicate ed ottimizzate
- Benefici
  - Prestazioni ottime, dimensioni ridotte, bassa potenza
- Svantaggi
  - Costi non ricorrenti molto alti (esempio 300k$) e time-to-market molto elevato



## Semi-custom ASIC

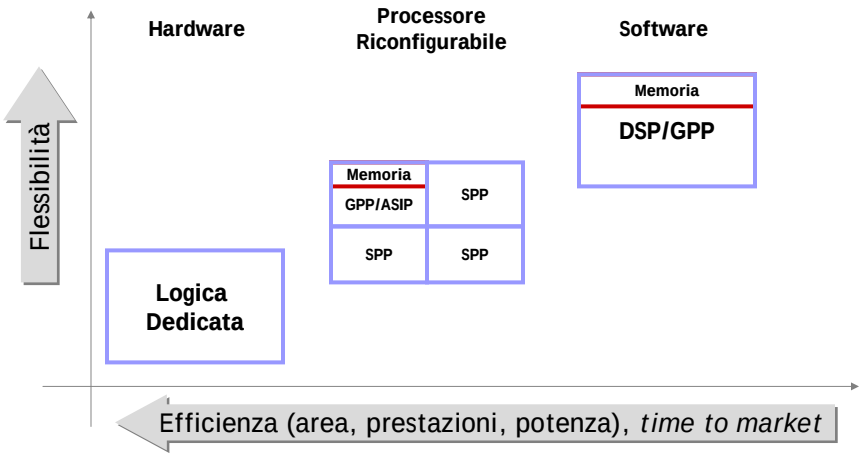
- Struttura generale (struttura regolare di transitor pre-implementati) che viene adattata per una particolare funzionalità digitale
  - Dimensione e posizione dei transistor fissate
  - Connessioni dedicate ed ottimizzate
  - Rispetto ai *Gate Array*, le strutture *sea-of-gates* non hanno dei canali di routing predefiniti consentendo implementazioni più compatte.
- Benefici
  - Prestazioni buone, dimensioni medie, media potenza
  - Rispetto alle realizzazioni *full-custom* si hanno sia ridotti tempi di fabbricazione e progetto, sia minori costi NRE.
    - Rispetto alle realizzazioni *full-custom*, il chip è pre-fabbricato e il processo si limita a definire le sole metallizzazioni (60% di maschere in meno) e, poiché la posizione dei transistor è predefinita, il tempo necessario per progettare il layout di una cella è ridotto drasticamente
- Svantaggi
  - Costi non ricorrenti medi e time-to-market elevato

## PLD

- Dispositivi hardware che mettono a disposizione componenti logici più o meno complessi che possono essere connessi tra loro a seconda delle esigenze di progetto
  - Dispongono di:
    - Componenti logici
      - Porte logiche, Flip-flop, Buffer…
    - Linee di connessione
- Benefici
  - Bassi costi NRE, Basso time-to-market
- Svantaggi
  - Prestazioni ridotte, dimensioni elevate e potenza elevata



## Vincoli di Progetto e Architettura



## Vincoli di Progetto e Architettura

- La differenza principale tra una architettura basata su processori General-Purpose e le altre possibili è il livello di specializzazione.
- Scelta della architettura richiede di:
  - individuare "quanto è speciale" una data applicazione.
  - Non compromettere la flessibilità
    - I sistemi con campo di applicazione specifico dovrebbero coprire una classe di applicazioni
    - La flessibilità è indispensabile per tenere in conto di cambiamenti dell'ultimo momento e della possibilità di correggere errori
    - Una eccessiva flessibilità potrebbe essere controproducente in termini di costo, prestazione e potenza
- L'analisi di sistema richiede
  - Identificare le proprietà della applicazione che possono essere usate per specializzare il sistema
  - Quantificare gli effetti di ogni singola specializzazione



## Architettura e Processo di Co-design

- L'architettura impatta sul processo di Co-design
  - Influenza sulla fase di analisi per l'identificazione delle caratteristiche specifiche dalla architettura obiettivo e sulla fase di Co-simulazione
    - Necessaria una modellazione accurata sia della comunicazione, sia della esecuzione sia delle temporizzazioni
    - Necessario determinare l'accuratezza dei risultati ottenuti nella fase di co-verifica
      - Stimatori efficienti ed accurati per: prestazione (concorrenza, latenza), costi (bus, processori, memorie), potenza dissipata, grado di affidabilità del sistema (self-checking, fault-tolerance)
  - Influenza sulla fase di Co-sintesi
    - Multi-processo e multi-thread, con e senza prelazione, molti modelli di comunicazione, spazio di progetto vincolato…

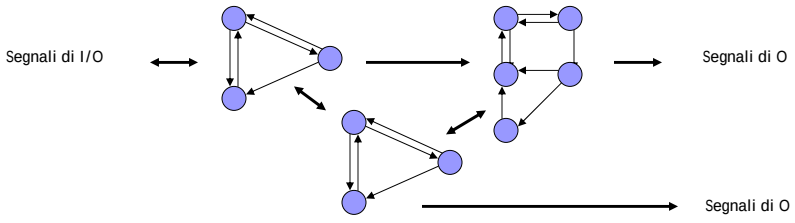
## Requisiti della applicazione

- Le architetture sono strattamente correlate al tipo di sistema che domina l'applicazione.
  - Sistemi dominati dal controllo
  - Sistemi dominati dai dati
  - Sistemi misti
    - Telefono cellulare, Controllore di motore
- Le entità di processo hanno una specifica *affinità* con una classe di applicazione
  - I/O, profondità elevata del flusso di istruzioni, flusso di dati di dimensioni uniformi da 8,16,32,64 bit: GPP, ASIP
  - Utilizzo di array circolari, operazioni di *somma di prodotti*: DSP
  - Profondità ridotta del flusso di istruzioni, flusso di dati di dimensioni disuniformi e/o di pochi bit (anche 1): ASIC, FGPA



## Sistemi Dominati dal Controllo

- Sistemi reattivi con un comportamento guidato dagli eventi
- Il modello di computazione, utilizzato per descrivere il sistema, è realizzato mediante macchine a stati finiti (FSM) e/o reti di petri (PN).
- *Grande unità di controllo con un data path piccolo*



## Sistemi Dominati dal Controllo

- Problemi di Co-design
  - Esecuzione concorrente di FSM che reagiscono a eventi asincroni
  - Flusso di controllo dipendente dai dati di ingresso
  - Correttezza, sincronizzazione tra FSM, scheduling degli eventi
- Problemi che derivano dalla architettura obiettivo
  - Scheduling dei processi e sincronizzazione tra processi
  - Prelazione e cambio di contesto
  - Tempi di latenza brevi
  - Dimensione del codice, soprattutto nel caso di prodotto di FSM.

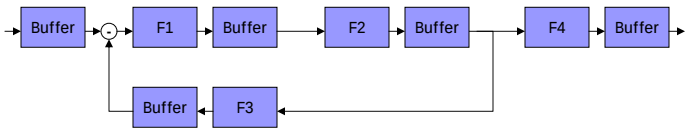


## Sistemi Dominati dal Controllo

- Opportunità per la specializzazione
  - Per lo più operazioni booleane o manipolazione di bit
  - Transizioni di stato con poche operazioni e stati contenuti in un insieme di dati ridotto
  - Data-path e unità funzionali specilizzate, architetture di memoria semplici, controllo centralizzato
- Architettura: sistemi basati su microcontrollori
  - Adatto per applicazioni dominate dal controllo e real-time
    - Ridotto tempo di commutazione del contesto
  - Basso consumo di potenza
  - Timer, interrupt controller e memoria su chip
  - Unità periferiche
    - UART, convertitore A/D, PWM

## Sistemi Dominati dai Dati

- Sistemi caratterizzati da un flusso di dati a frequenza costante (streaming oriented) con un comportamento generalmente periodico (cyclo static systems)
- Il modello di computazione, utilizzato per descrivere il sistema, è realizzato mediante diagrammi di flusso (esempio: DFL).
- *Piccola unità di controllo ed un ampio data path*






## Sistemi Dominati dai Dati

- Problemi di Co-design
  - Prestazioni elevate (throughput) a basso costo
  - Insieme ampio di dati (la memoria è un elemento rilevante)
  - Scheduling delle operazioni e dimensionamento dei buffer
- Problemi che derivano dalla architettura obiettivo
  - Il trasferimento dei dati e la memoria determinano i parametri di costo e prestazioni:
    - Dimensione: sono tipicamente richieste memorie di ampie dimensioni
    - Prestazione: l'accesso alla memoria è il collo di bottiglia
    - Energia: memoria e bus sono fattori di consumo energetico

## Sistemi Dominati dai Dati

- Opportunità per la specializzazione
  - Il flusso di controllo è parzialmente indipendente dai dati
  - Ingressi periodici e a periodo fissato
  - Il flusso dati generalmente sincrono
  - Architettura di memoria e unità di indirizzamento specializzati, supporto ai loop.
- Architettura: sistemi basati su DSP
  - Ottimizzato per applicazioni dominate dai dati
  - Adatto anche per flussi di controllo semplici
  - Unità hardware parallele
  - Insieme di istruzioni specializzato (MAC)



## Sistemi Misti

- La maggior parte dei sistemi non ha una specifica caratteristica
  - Esempi: controllori per motori, applicazioni per la telefonia, cellulari…
- Architettura: Sistema eterogeneo costituito da parti specializzate che comunicano (DSP, Microcontrollore, periferiche, dispositivi ad-hoc, …)

## Architetture: Miti e realtà (hardware)

- Mito: Attualemente il mercato dei sistemi embedded è dominato dai processori a 32 bit
  - I processori da 8, 16 bit sono pressoché estinti
  - Le architetture X86 sono tra le più utilizzate
  - La maggior parte dei processori a 32 bit è utilizzata nei PC e nelle *work-station.*
- Realtà: più del 80% dei processori utilizzati è a 4-16 bit
  - La maggior parte dei processori utilizzati è a 4-16 bit
  - Solo il 30% dei sistemi embedded utilizza architetture x86
  - I processori a 32 bit utilizzati per PC e *workstation* coprono solo il 40% del mercato dei processori a 32 bit.



## Architetture: Miti e realtà (hardware)

## Architetture: Miti e realtà (hardware)

- Mito: La tendenza attuale è verso una rezionalizzazione del numero delle architetture basate su processori
  - Poche architetture dominanano fette di mercato sempre più ampie.
- Realtà: La disponibilità di API (Application Programming Interface) consente la proliferazione di architetture nuove ed innovative.
  - API: qualunque insieme di routine, generalmente disponibili per essere utilizzate dai programmatori, che rendono il codice protabile su altre piattaforme; realizzano una macchina astratta che consente di ignorare i dettagli implementativi del sistema sottostante.



## Architetture: Miti e realtà (hardware)

- Mito: Gli ASIP (Application-Specific Instruction-set Processor) rappresentano un male necessario ma di breve durata
- Realtà: Tutti i dispositivi di elevata diffusione utilizzano uno o più ASIP.
  - La maggior parte degli apparati GSM.
  - La maggior parte del decoder MPEG.
  - Molti video-games.
  - La maggior parte dei dispositivi per la manipolazione di immagini 3D (se non sono puramente hardware).

## Architetture: Miti e realtà (software)

- Mito: Attualmente, la maggior parte delle applicazioni software per sistemi embedded è scritto in C.
- Realtà: Solo il 20% è scritto in C; il resto è scritto direttamente in assembler.
- Mito: I compilatori attuali producono un codice con un sovraccarico accettabile
- Realtà: Il codice prodotto dai compilatori è spesso inaccettabile
  - Codici DSP: dal 200% al 500%
  - Codici per applicazioni di controllo: dal 50% al 100%
  - Codici per PC e workstation: meno del 50% (il problema è comunque irrilevante)