4/10/22

Lezione E2

Tecnologie per sistemi embedded

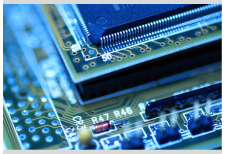
analizzo elettronica & informatica

Sistemi embedded e real-time

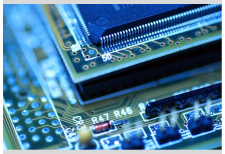
26 novembre 2020

Marco Cesati

Dipartimento di Ingegneria Civile e Ingegneria Informatica
Università degli Studi di Roma Tor Vergata



Di cosa parliamo in questa lezione?



In questa lezione parliamo di alcuni aspetti legati al processo di realizzazione di un sistema embedded

1. Metodologie di sviluppo
2. Varie tecnologie hardware
3. Gli ASIC
4. I chip programmabili
5. I microprocessori

Lo sviluppo dei sistemi embedded

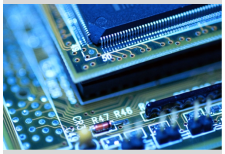
Per realizzare un sistema embedded è necessario considerare tre aspetti fondamentali:

- Architetture
- Applicazioni
- Metodologie

vale anche per i sofware, dove ciascun aspetto può essere approfondito in modo abbastanza indipendente

Nei sistemi informatici "general-purpose" è possibile approfondire ciascuno di questi aspetti in modo relativamente indipendente dagli altri

Al contrario, nei sistemi embedded architettura, applicazione e metodologia di sviluppo sono fortemente interdipendenti



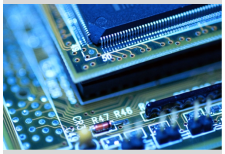
Obiettivi dello sviluppo di sistemi embedded

Alla base dello sviluppo di un sistema embedded c'è sempre la necessità di realizzare una determinata applicazione

(capacità calcolo, dati, prestazioni...)

- L'applicazione determina i requisiti funzionali e non-funzionali del sistema
- Esempi di requisiti funzionali:
  - Capacità di elaborazione, memorizzazione, comunicazione
  - Prestazioni (media e caso peggiore, throughput e latenza, di picco e a regime, . . .)
  - Consumo di energia (durata della batteria) e consumo di potenza (dissipazione di calore)
  - Affidabilità e robustezza (tempo di vita)
  - Sicurezza del prodotto
- Esempi di requisiti non-funzionali del sistema:
  - Costi di fabbricazione, costi di progetto, costi di manutenzione dell'hardware e del software
  - Durata della fase di progettazione
  - Numero di pezzi da produrre

Tecnologie per sistemi embedded
Marco Cesati



Schema della lezione
Metodologie di sviluppo
Application Specific Integrated Circuit
Programmable Logic Device
Microprocessori

Metodologie di sviluppo

↳ "generale"

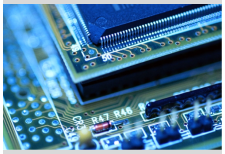
Perché è indispensabile definire una metodologia di sviluppo?

Lo sviluppo di un sistema embedded richiede:

- creatività per scoprire nuove soluzioni e adattare vecchie idee
- ripetibilità per evitare di ricominciare sempre da zero il processo di sviluppo di un nuovo prodotto
- rapidità per rispettare i vincoli sulla durata della fase di progetto
- prevedibilità dei costi di progettazione, fabbricazione e manutenzione

Definire una buona metodologia di sviluppo è essenziale per soddisfare questi requisiti (tranne quello della creatività!)

La **metodologia di sviluppo** non è (solo) una astrazione od una teoria da seguire: deve essere definita in termini di **strumenti** e **risorse** disponibili!



Le difficoltà dello sviluppo

- Il processo di sviluppo comprende molte fasi diverse
- Per molte di queste fasi non esiste alcuno strumento di sintesi, quindi è necessario operare molte analisi e simulazioni
- È necessario costruire rapidamente simulatori della specifica applicazione embedded
- Non è possibile simulare con elevati livelli di accuratezza il sistema, a causa dei vincoli sulla durata della fase di progetto e dei limiti di costo dello sviluppo
- Sono necessari strumenti sofisticati per progettare architetture eterogenee con diversi tipi di processori, memorie, dispositivi di comunicazione, ...

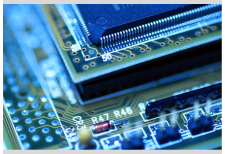
Tradizionalmente le metodologie di sviluppo utilizzate per il software sono differenti da quelle utilizzate per l'hardware

Che tipo di metodologia adottare per i sistemi embedded?

L'ideale è una metodologia ibrida con il meglio dei due mondi!

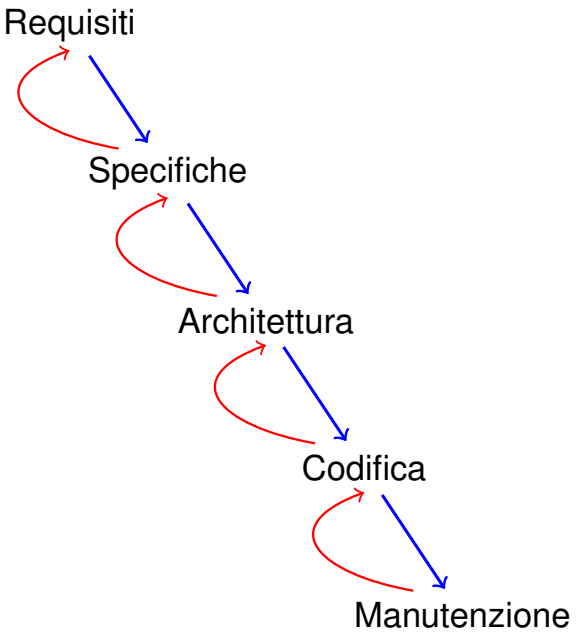
sì, ma come lo faccio!?

Tecnologie per sistemi embedded
Marco Cesati

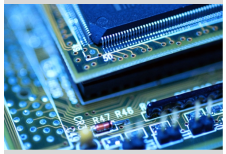


Schema della lezione
Metodologie di sviluppo
Application Specific Integrated Circuit
Programmable Logic Device
Microprocessori

Sviluppo del software “a cascata”

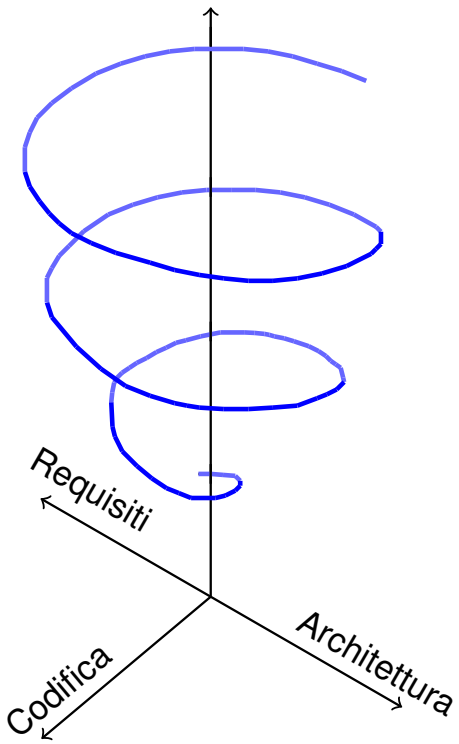


- Le fasi di progettazione sono rigidamente gerarchiche
- Per progettare un livello è necessario definire completamente il livello superiore
- È molto costoso apportare modifiche ad un livello superiore mentre si sta definendo un livello inferiore

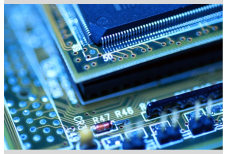


Sviluppo del software “a spirale”

Il giro "n" integra i precedenti "n-1" giri, "NON TORNO" INDIETRO



- Lo sviluppo è un processo iterativo che alterna tra le diverse fasi
- In ogni ciclo costituito da tutte le fasi si ottiene una versione del prodotto più evoluta
- In ogni ciclo si sfrutta l’esperienza accumulata nei cicli precedenti
- È più facile introdurre modifiche e raffinamenti in ciascuna fase



Sviluppo dell'hardware a sintesi digitale

Le metodologie di sviluppo dell'hardware utilizzano tecniche sofisticate assenti nei processi di sviluppo del software

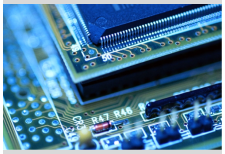
Ad esempio:

- Algoritmi di sintesi dei circuiti basati su tecniche di ricerca
- Algoritmi per la stima delle metriche (consumo di energia, influenza elettromagnetica, area occupata,...)
- Algoritmi predittivi per la valutazione dei modelli che operano su definizioni incomplete del progetto

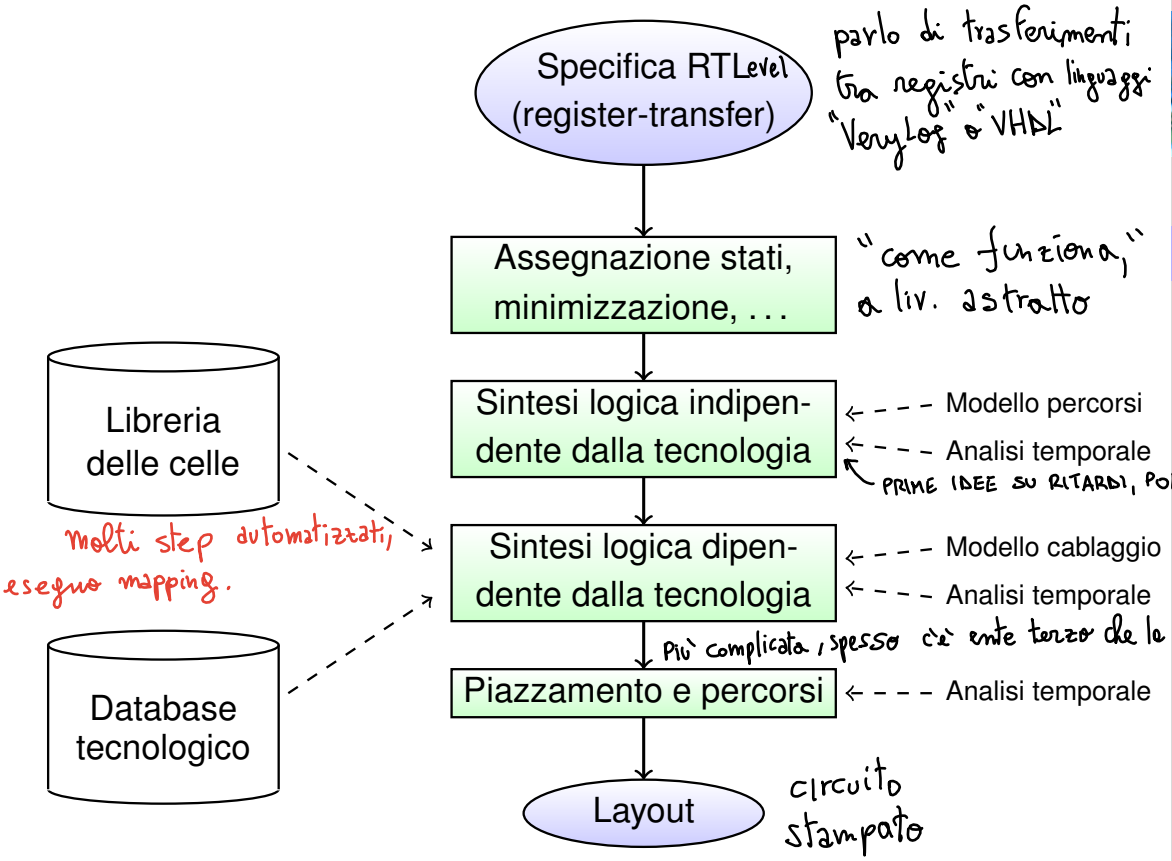
Perché tecniche equivalenti non sono largamente usate nella produzione del software? nel SW ho anche requisiti non funzionali!

- Nel progetto dell'hardware le metriche di valutazione del prodotto sono obiettive e facili da quantificare
- Ciascuna metrica rappresenta quasi sempre un vincolo progettuale molto forte e deve quindi essere ottimizzata
- I cicli di sviluppo dell'hardware sono in genere più brevi

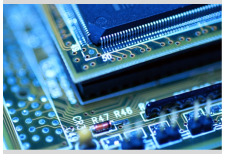
una CPU deve avere α nm, dim < y ..., non posso dire lo stesso del SW!



Sviluppo dell'hardware a sintesi digitale (2)



non ho ancora fatto nulla!



Co-design di sistemi embedded

funzionalità realizzabile in HW o SW

Quale è il passaggio fondamentale nello sviluppo di un qualunque sistema embedded?

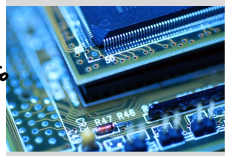
es: PROIETTORE → SPEGNI SE SI SURRISCALDA → SW → sensore invia dati a programma, microcontrollore SA, FLESSIBILE ma aumento il carico!
→ HW → metto sensore, microcontrollore NON SA, "SICURO" ma NON TARABILE

Partizionamento hardware/software

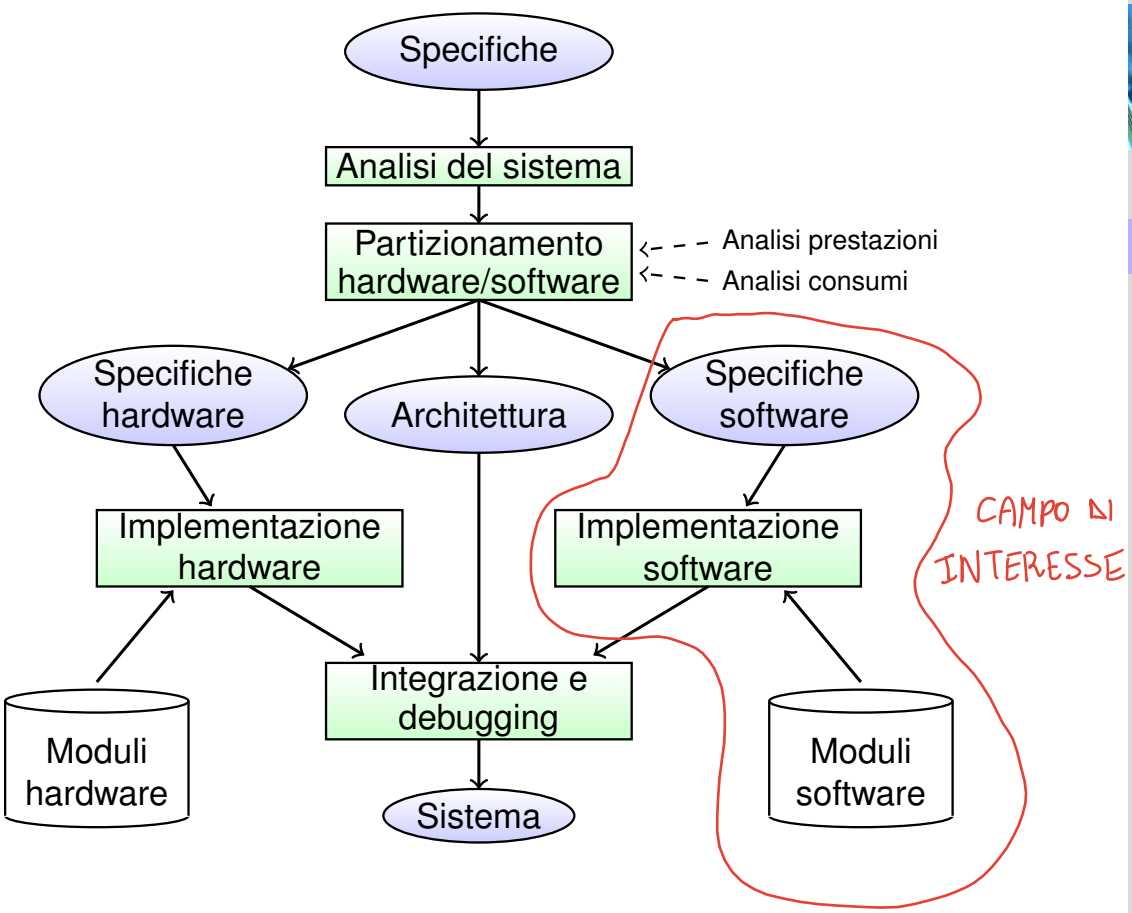
Decisione critica: stabilire quale funzionalità del prodotto dovrà essere realizzata direttamente in hardware e quale dovrà essere realizzata dal software di un dispositivo programmabile

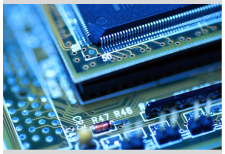
La metodologia di sviluppo *co-design* (da "concurrent design"):

- Analizza i requisiti e determina un opportuno partizionamento hardware/software
- Definisce l'architettura generale del sistema embedded, ed in particolare l'interfacciamento tra hardware e software
- Determina le specifiche dell'hardware e del software
- Consente di realizzare le componenti hardware e software separatamente e contemporaneamente



Co-design di sistemi embedded (2)





Sviluppo di sistemi embedded basati su piattaforma

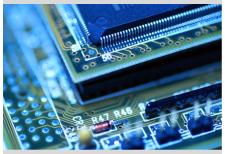
Spesso i sistemi embedded vengono sviluppati definendo od utilizzando una *piattaforma* di riferimento

dentro hanno "tutto", mini-PC

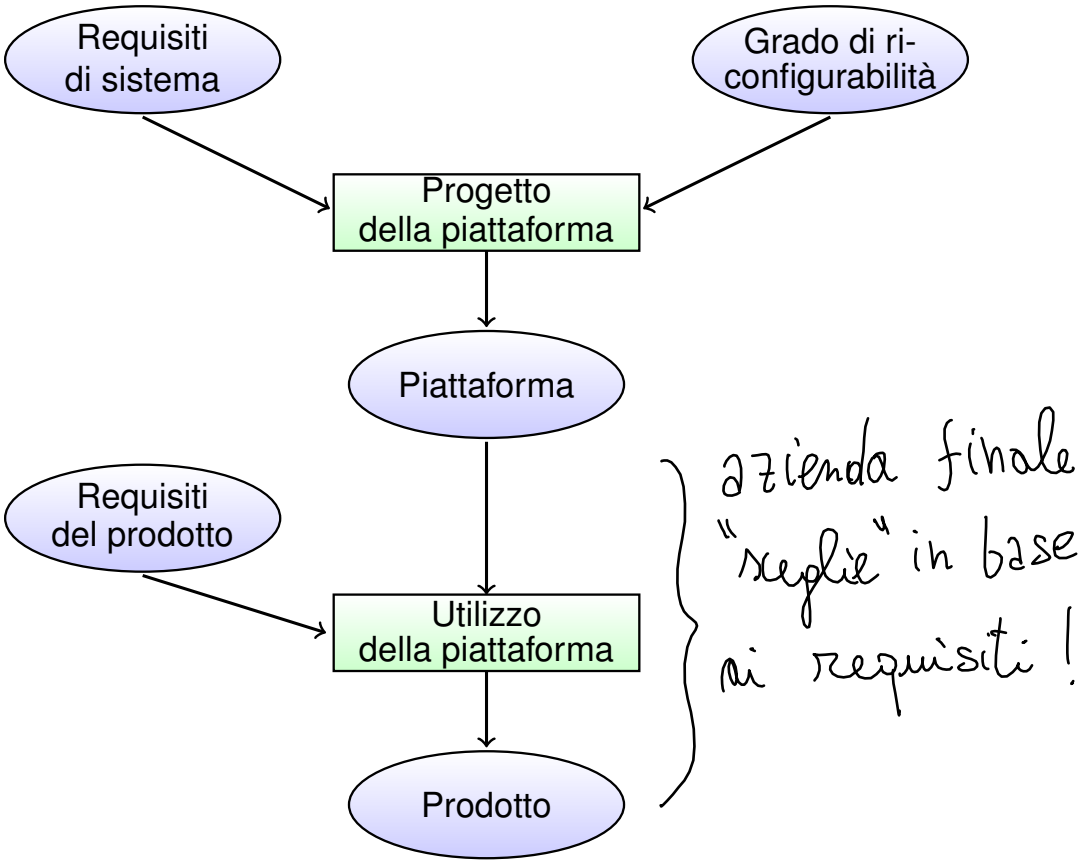
- Metodologia molto comune per i *Systems-on-Chips* (*SoC*)
- La piattaforma è un dispositivo o una architettura di base che può essere facilmente modificata per realizzare nuove funzionalità o modificare quelle esistenti (meglio di niente!)
- Molto utile quando il sistema embedded deve realizzare funzioni definite e regolate da standard (esempi: lettore DVD, access point IEEE 802.11, stampante PostScript)

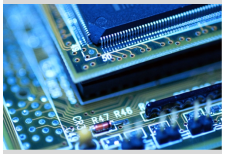
Il processo di sviluppo è costituito da due fasi:

1. Si definisce la piattaforma in termini di funzionalità di base, prestazioni e riconfigurabilità
2. Si utilizza la piattaforma così definita per progettare un prodotto specifico basato su di essa



Sviluppo di sistemi embedded basati su piattaforma (2)





Tecnologie hardware per sistemi embedded

Perché nasce?

Alla base dei sistemi embedded è la tecnologia integrata

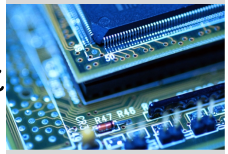
- corsa alla miniaturizzazione, piccolo ~ meno ritardi nella propagazione, limite fisico: CALORE (Prima era DISTANZA TRA COMPONENTI)
- inventata alla fine degli anni '50
- basata su un processo di fabbricazione *planare*
- ha avuto un progresso tecnologico molto veloce

In pratica, il progettista ha ampia possibilità di scelta tra:

- componenti COTS (Commercial, Off-the-Shelf) già pronti
- microprocessori, inclusi microcontrollori e processori specializzati (DSP, . . .)
- logiche programmabili
- ASIC (Application-Specific Integrated Circuit)

Ad-hoc per l'applicazione

Quale scegliere?



Tecnologie per ASIC

, fa 1 sola cosa, non riprogrammabile

Se un progettista deve affidare una certa funzione di un prodotto ad un circuito integrato apposito:

- può utilizzare un circuito integrato già prodotto e commercializzato (componente COTS)
- può sviluppare in proprio un circuito specializzato per quella specifica funzione (componente ASIC)

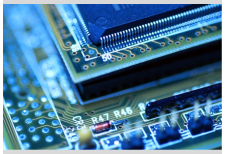
I costi di progettazione e sviluppo di un ASIC sono molto alti

Quando si giustifica la scelta di progettare un ASIC?

- si richiedono dimensioni ridotte, particolari esigenze
- si richiedono altissime prestazioni
- il mercato richiede volumi molto elevati
- nessun componente COTS è adeguato

È soprattutto l'enorme costo di sviluppo di nuovi circuiti integrati che ha portato l'industria a definire piattaforme di riferimento da cui derivare nuovi prodotti (Progresso)

Tecnologie per sistemi embedded
Marco Cesati



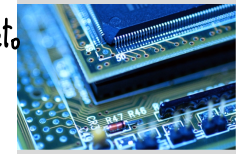
Schema della lezione
Metodologie di sviluppo
Application Specific Integrated Circuit
Programmable Logic Device
Microprocessori

Come progettare ASIC?

Tecnologia standard cell

Per semplificare i problemi di progettazione degli ASIC è stata introdotta una tecnologia nota come standard cell circuiti digitali completo → riutilizzabile, tipo LEGO.

- si alza il livello di astrazione con il quale si progettano i circuiti: dal transistor si passa alla cella
- una cella è un circuito digitale completo già progettato e ottimizzato da riutilizzare a piacimento
- le definizioni delle celle sono raccolte in librerie commerciali
- ciascuna libreria contiene migliaia di tipi di celle per realizzare altrettante funzioni (logiche di base, multiplexer, adder, flip-flop, registri, memorie, ...)
- tutte le celle, quando realizzate nel silicio del circuito integrato, hanno la stessa altezza
- esistono programmi che calcolano il piazzamento ottimale delle celle nel silicio per realizzare le interconnessioni richieste, voglio distanza minima!

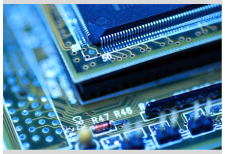


Tecnologia gate array (evoluzione)

Anche la tecnologia gate array è stata introdotta per semplificare la progettazione degli ASIC

- È simile alla tecnologia standard cell, nel senso che un venditore fornisce una libreria di circuiti digitali completi
- In standard cell il chip è costruito su un die di silicio puro (quindi le celle sono molto flessibili) die: piastrina su cui realizzo circuito
- In gate array il chip è parzialmente fabbricato: il silicio è già organizzato in gruppi di transistor (gate base) organizzati in righe regolari separate da canali di routing per le interconnessioni
- Una variante nota come sea of gates non possiede canali di routing: le interconnessioni sono affidate ad uno strato di metallo posto sopra i transistor

In passato, ogni anno venivano progettati decine di migliaia di nuovi ASIC; al giorno d'oggi invece il numero di nuovi progetti è fortemente ridotto. Il calo è dovuto alla nuova frontiera dei sistemi embedded: le **logiche programmabili**!



Logiche programmabili

Le *logiche programmabili* (o *PLD*, Programmable Logic Device) sono chip che integrano risorse logiche e linee di interconnessione completamente fabbricate

I chip sono *programmabili* nel senso che:

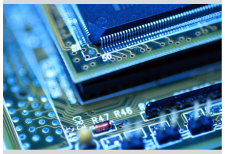
- ciascuna risorsa logica può essere configurata per svolgere una specifica funzione
- ciascuna linea di interconnessione può essere collegata o meno a varie risorse logiche

I PLD hanno differenti gradi di *programmabilità*:

- one-time programmable (OTP): la configurazione del chip è irreversibile ed è ottenuta applicando tensioni elettriche più alte di quelle della normale alimentazione
- riprogrammabili: la configurazione può essere effettuata più volte; le interconnessioni sono transistor pilotati dai bit di un circuito di memoria volatile (RAM statica) oppure persistente (EEPROM, Flash) (solo a circuito spento)
- riconfigurabili: la configurazione può essere effettuta più volte mentre il circuito è in funzione ed in modo selettivo (sono le più recenti)

(anche a circuito attivo, solo in alcune parti)

Io "programmo" un vero circuito elettronico, dicendo connessioni aperte e chiuse!



Complessità e organizzazione dei PLD

I PLD si distinguono anche in base alla complessità delle risorse logiche elementari (*celle*)

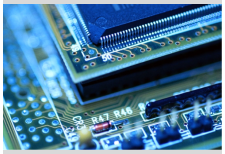
- Ad un estremo, una cella è costituita da una o due porte logiche, oppure da un multiplexer a 2 o 4 ingressi, oppure da un latch o flip-flop
- All'altro estremo, una cella contiene una o due circuiti in grado di realizzare qualunque funzione booleana a 4 ingressi, qualche porta logica, uno o due multiplexer e qualche flip-flop

Quali sono vantaggi delle celle semplici e quali quelli delle celle complesse?

Il vantaggio delle celle semplici è che il silicio è maggiormente sfruttato: il rischio di sotto-utilizzare qualche cella è ridotto

Il vantaggio delle celle complesse è che ne servono di meno, quindi è più facile realizzare le loro interconnessioni

Tecnologie per sistemi embedded
Marco Cesati



Schema della lezione
Metodologie di sviluppo
Application Specific Integrated Circuit
Programmable Logic Device
Microprocessori

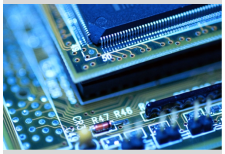
Tipologie di PLD

- PLA (Programmable Logic Array): costituito da zone di porte AND e OR e da aree di interconnessione configurabili (OTP)
- PAL (Programmable Array Logic): come i PLA, ma più semplici: la configurazione delle porte OR è prefissata (Solo AND)
- GAL (Generic Array Logic): include diversi PAL, con multiplexer per retro-azionare le uscite, e con flip-flop; inoltre può essere riprogrammato
- CPLD (Complex PLD): basato su celle complesse (GAL) ed un bus comune configurabile per mezzo di una memoria EEPROM o Flash
- FPGA (Field Programmable Gate Array): celle complesse e interconnessioni sono distribuite regolarmente nel chip; la configurazione è in memoria generalmente volatile

diffusione

spesso FPGA ma CPLD,

è un circuito, non un programma!



Utilizzo di microprocessori in sistemi embedded

Molti sistemi embedded sono realizzati sfruttando microprocessori più o meno evoluti e specializzati

↳ legge istr. e fa, è macchina programmabile

Quali sono i vantaggi dell'uso del microprocessore?

Vantaggio principale è la flessibilità: il software consente di avere un sistema molto più mantenibile ed estendibile

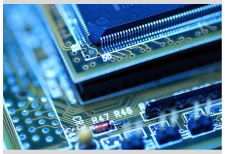
Altro vantaggio è che se l'applicazione da realizzare è complessa, il sistema basato su microprocessore è alla fine più semplice ed economico

Quali sono gli svantaggi nell'utilizzo dei microprocessori?

Lo svantaggio principale sono le prestazioni peggiori (rispetto all'implementazione in hardware della stessa funzionalità)

Tra le prestazioni si devono considerare anche il consumo di energia, la dissipazione di potenza, la memoria richiesta, il tempo di inizializzazione...

Tecnologie per sistemi embedded
Marco Cesati



Schema della lezione
Metodologie di sviluppo
Application Specific Integrated Circuit
Programmable Logic Device
Microprocessori

Tipologie di microprocessore per sistemi embedded

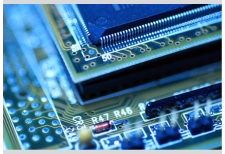
Si utilizzano sia microprocessori general purpose (GPP) che microprocessori dedicati (ASP, Application Specific Processor)

Ancora una volta la scelta è frutto di un compromesso tra flessibilità, tempi di sviluppo e prestazioni

Un'altra caratteristica molto importante è la forma in cui acquisire il microprocessore: COTS oppure IP

- I microprocessori COTS sono chip fisicamente indipendenti da integrare nel sistema
- I microprocessori IP (Intellectual Property) sono descrizioni a vario livello di astrazione dei circuiti:
  - Soft-macro: a livello Register-transfer in linguaggio VHDL, Verilog o equivalente, modifico come voglio!
  - Firm-macro: a livello di gate (netlist per specifici PLD o ASIC)
  - Hard-macro: a livello di layout per specifici PLD o ASIC

  → Compro "come è fatto", poi lo uso in FPGA!



Digital Signal Processor

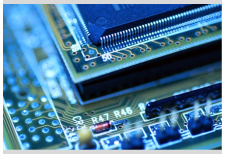
I DSP sono la classe principale di processori dedicati, e sono ottimizzati per l'elaborazione numerica

Alla base di quasi tutte le elaborazioni numeriche dei segnali si trova una sola operazione ricorrente: $z_{t+1} = z_t + x \cdot y$

La complessità di un dato algoritmo è espressa in numero di MAC (Multiply and Accumulate) e le prestazioni in MMAC/s (milioni di MAC al secondo)

Le architetture per DSP:

- Realizzano l'operazione MAC con una singola istruzione
- Ottimizzano l'esecuzione di cicli piccoli e con numero di iterazioni prefissato
- Hanno istruzioni per operare su più dati in parallelo (SIMD)
- Generalmente hanno insiemi di istruzioni VLIW (Very Large Instruction Word)
- Hanno gerarchie di memoria che distinguono tra codice e dati (architettura Harvard), o tra codice, dati e costanti (architettura SHARC, Super Harvard Architecture)



Network Processor

I network processor (NP, od anche packet processor) sono system-on-chip usati negli apparati di rete e telecomunicazione

Funzioni tipiche: la bufferizzazione dei pacchetti, l'elaborazione delle testate, la ricerca di indirizzi in tabelle, il calcolo di codici di controllo, la ritrasmissione di un pacchetto

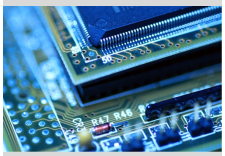
L'architettura di ogni NP è multi-processore: a ciascun canale è dedicato un processore specifico (CP, Channel Processor)

Alcuni processori RISC si occupano della supervisione e dello smistamento dei pacchetti tra canali diversi

Il collegamento tra i vari processori è dato da un insieme di bus ad altissima velocità ($\sim$ 100 Gbps)

La programmazione dei processori RISC è in genere in C

La programmazione dei CP è effettuata in *assembler* oppure con linguaggi basati su regole lessicali per *patter recognition*



Microcontrollori

non per general purpose; usato con sensori etc...

I microcontrollori (o MCU) sono microprocessori che dispongono di molte periferiche ed interfacce in un singolo chip

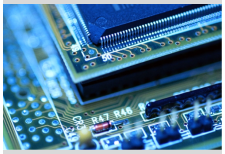
Sono particolarmente adatti per applicazioni che richiedono poca potenza di calcolo ma stretta interazione con l'hardware

- Unità aritmetico-logica (ALU) molto semplice
- Pochissimi registri, un registro "accumulatore"
- Un solo bus interno
- Memorie integrate (RAM, ROM, EEPROM) di pochi KB
- Timer hardware
- Piedini programmabili per I/O
- Convertitori analog./digitale (ADC) e digitale/analog. (DAC)
- Interfacce I^2C, SPI, JTAG, UART, PWM, ...

Quasi sempre non possiedono interfacce sofisticate verso la memoria esterna (bus e MMU): perché? Ram collegata al microcontrollore NO!

Soprattutto per tenere basso il numero di piedini (*pin*) del chip! complicherei il progetto!

Tecnologie per sistemi embedded
Marco Cesati



Schema della lezione
Metodologie di sviluppo
Application Specific Integrated Circuit
Programmable Logic Device
Microprocessori