Metodologie di Progettazione Hardware/Software

Introduzione

Embedded systems overview

- Computing systems are everywhere
- Most of us think of “desktop” computers
  - PC’s
  - Laptops
  - Mainframes 
  - Servers
- But there’s another type of computing system
  - Far more common...

Embedded systems overview

- Embedded computing systems
  - Computing systems embedded within electronic devices
  - Hard to define. Nearly any computing system other than a desktop computer
  - Billions of units produced yearly, versus millions of desktop units
  - Perhaps 50 per automobile

Embedded Systems

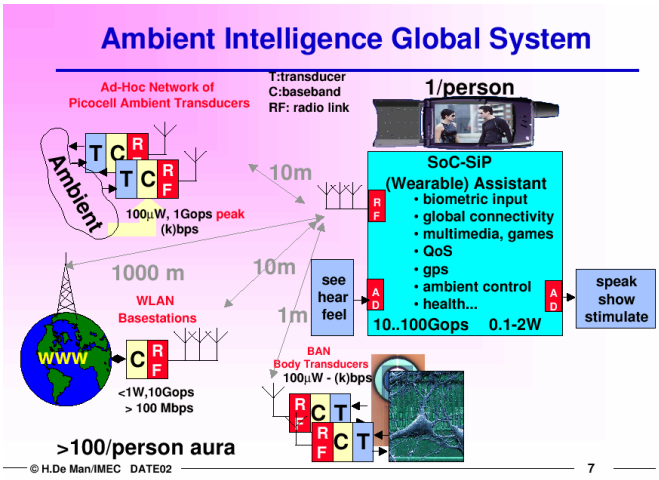
- **Embedded Systems (ES)** = sistemi di elaborazione delle informazioni dedicati contenuti all'interno di un prodotto più grande

Application areas (1)

- Automotive electronics
- Aircraft electronics
- Trains
- Telecommunication
- Military applications

Application areas (2)

- Consumer electronics



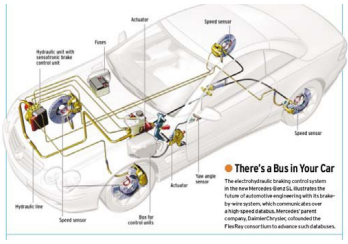
A “short list” of embedded systems

Anti-lock brakes
Auto-focus cameras
Automatic teller machines
Automatic toll systems
Automatic transmission
Avionic systems
Battery chargers
Camcorders
Cell phones
Cell-phone base stations
Cordless phones
Cruise control
Curbside check-in systems
Digital cameras
Disk drives
Electronic card readers
Electronic instruments
Electronic toys/games
Factory control
Fax machines
Fingerprint identifiers
Home security systems
Life-support systems
Medical testing systems

Modems
MPEG decoders
Network cards
Network switches/routers
On-board navigation
Pagers
Photocopiers
Point-of-sale systems
Portable video games
Printers
Satellite phones
Scanners
Smart ovens/dishwashers
Speech recognizers
Stereo systems
Teleconferencing systems
Televisions
Temperature controllers
Theft tracking systems
TV set-top boxes
VCR’s, DVD players
Video game consoles
Video phones
Washers and dryers

Cars

- Multiple processors
  - ✓ Up to 100
  - ✓ Networked together
- Multiple networks



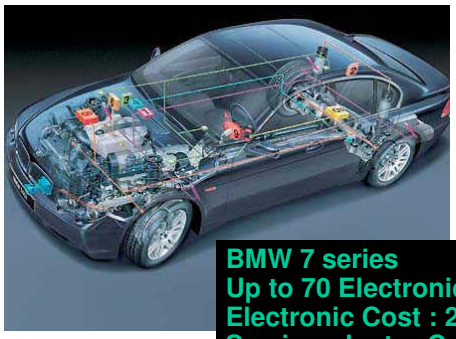
–Functions:

- ABS: Anti-lock braking systems
- ESP: Electronic stability control
- Airbags
- Efficient automatic gearboxes
- Theft prevention with smart keys
- Blind-angle alert systems
- ... etc ...

Cars

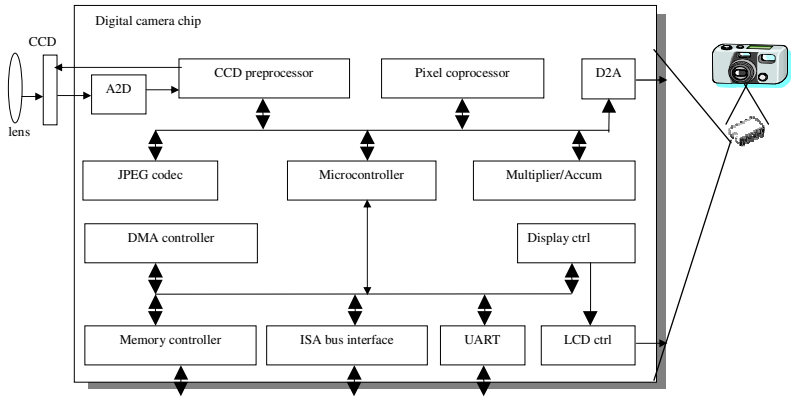
- Large diversity in processor types:
  - 8-bit – door locks, lights, etc.
  - 16-bit – most functions
  - 32-bit – engine control, airbags
- Form follows function
  - Processing where the action is
  - Sensors and actuators distributed all over the vehicle

Cars



BMW 7 series
Up to 70 Electronic Modules
Electronic Cost : 25% total Car cost
Semiconductor Content > 1000 $

Digital camera



- Single-functioned -- always a digital camera
- Tightly-constrained -- Low cost, low power, small, fast
- Reactive and real-time

Caratteristiche dei sistemi embedded

- Specializzazione
  - Eseguono un solo programma, ripetutamente
- Inclusione di elettronica ed altri dispositivi dedicati per:
  - Interagire con il mondo esterno
  - Svolgere alcune funzioni elaborative
- Sistemi reattivi e real-time
  - Devono continuamente reagire ai cambiamenti dell'ambiente in cui opera il sistema
  - Deve produrre certi risultati in real-time senza ritardo
- Con molti vincoli (spesso assai stringenti)
  - Basso costo, ridotto consumo di potenza, piccolo, veloce, ecc.

Caratteristiche dei sistemi embedded

Un sistema embedded è progettato per eseguire una sola o comunque poche specifiche applicazioni:

- Le applicazioni da svolgere sono note a priori, prima che il processo di progettazione inizi
- Spesso è opportuno garantire al sistema la flessibilità necessaria per i futuri aggiornamenti o per un eventuale riutilizzo del componente. Normalmente si raggiunge questo scopo rendendo il sistema riprogrammabile

Caratteristiche dei sistemi embedded

- Dovendo interagire con il mondo esterno i S.E. includono dispositivi quali
  - Sensori
  - Attuatori
- I sistemi embedded sono sistemi ibridi (digitale + analogico)
  - Sono presenti convertitori A/D e D/A
- L'interazione con l'utente avviene con mezzi spesso semplificati:
  - Display di dimensione ridotta
  - Dispositivi di input limitati
  - Dispositivi di I/O specializzati rispetto alle competenze o al modo di operare dell'utente

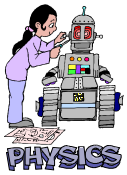
Caratteristiche dei sistemi embedded

- Alcune funzioni possono essere svolte in maniera più efficiente facendo ricorso a dispositivi hardware dedicati quali DSP, celle IP, ecc.
- Applicazioni DSP tipiche:
  - segnali generici: filtraggio, DFT, FFT, ecc.
  - voce: encoding, decoding, equalizzazione, ecc.
  - modem: modulazione, demodulazione
- Altro hardware dedicato:
  - Compressione video, crittografia, …

Caratteristiche dei sistemi embedded

- Molti ES devono rispettare vincoli **real-time**
- Un sistema real-time system deve reagire agli stimoli provenienti dall'oggetto controllato entro un intervallo di tempo che dipende dall'ambiente.
- Per i sistemi real-time, risposte giuste arrivate tardi sono errate.
- **„Un vincolo real-time constraint è detto hard, se il non rispetto del vincolo potrebbe avere conseguenze catastrofiche "** [Kopetz, 1997].
- Tutti gli altri time-constraints sono detti **soft**.

Caratteristiche dei sistemi embedded



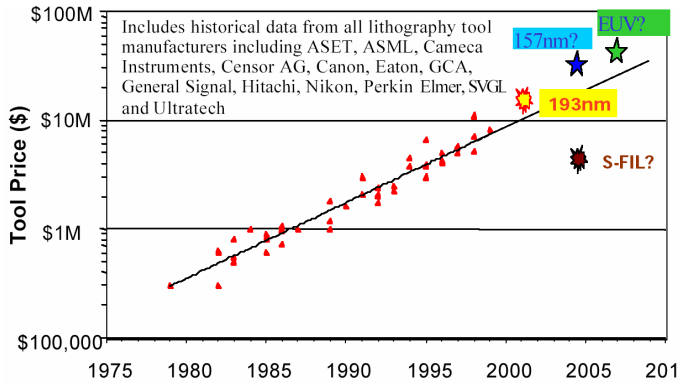
- Tipicamente sono **reactive systems**:
  "**A reactive system is one which is in continual interaction with is environment and executes at a pace determined by that environment**" [Bergé, 1995]
  Il comportamento dipende dagli ingressi e dallo stato corrente.
  ☞ la macchina a stati è il modello più appropriato.

Principali requisiti dei sistemi embedded

- I requisiti di progetto imposti ad un Sistema Embedded possono essere:
  - Requisiti funzionali
  - Requisiti temporali
  - Requisiti di affidabilità
  - Consumo
  - Prestazioni
  - Costo
- Spesso i requisiti risultano essere in contrasto tra loro

Challenges for implementation in hardware

- Lack of flexibility (changing standards).
- Mask cost for specialized HW becomes very expensive



☞Trend towards implementation in Software

Importance of Embedded Software and Embedded Processors

"... the New York Times has estimated that the average American comes into contact with about 60 micro-processors every day...." [Camposano, 1996]

Latest top-level BMWs contain over 100 micro-processors [Personal communication]

Most of the functionality will be implemented in software

Challenges for implementation in software

If embedded systems will be implemented mostly in software, then why don't we just use what software engineers have come up with?

- Exponential increase in software complexity
- In some areas code size is doubling every 9 months [ST Microelectronics, Medea Workshop, Fall 2003]
- *... > 70% of the development cost for complex systems such as automotive electronics and communication systems are due to software development*

Challenges for Embedded Software

- Come catturare il comportamento richiesto di sistemi complessi?
- Come tradurre in modo efficiente le specifiche in una implementazione?
- Deve essere presa in considerazione dagli ingegneri del software la potenza dissipata?
- Come verificare che i vincoli real-time sono rispettati?
- Quale linguaggio di programmazione fornisce caratteristiche real-time?

Challenges for Embedded Software

- It is not sufficient to consider ES just as a special case of software engineering
- EE knowledge must be available, Walls between EE and CS must be torn down

Design metrics

Progettazione – ottimizzazione di diverse design metrics

- Obiettivo di progetto:
  - Realizzare una implementazione con la funzionalità richiesta
- Sfide nella progettazione:
  - Ottimizzare contemporaneamente diverse design metrics
- Design metric
  - Una caratteristica misurabile dell'implementazione del sistema

Implementazione: un microprocessore con relativo programma, una connesione di gates o una loro combinazione

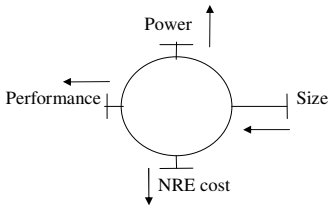
Progettazione – ottimizzazione delle design metrics

- Metriche comuni
  - **NRE cost** (Non-Recurring Engineering cost): il costo di progettazione del sistema, da conteggiare una sola volta
  - **Unit cost**: il costo economico per la produzione di ogni copia del sistema, escluso il costo NRE
  - **Size**: lo spazio fisico richiesto dal sistema
  - **Performance**: il tempo di esecuzione o il throughput del sistema
  - **Power**: la potenza consumata dal sistema
  - **Flexibility**: la capacità di modificare la funzionalità del sistema senza richiedere un elevato valore di NRE cost

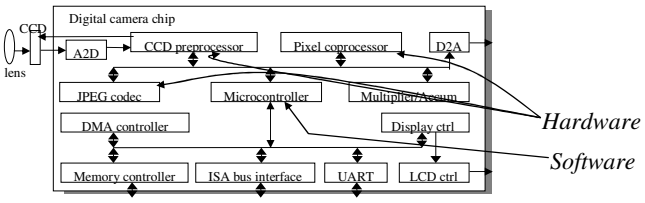
Progettazione – ottimizzazione delle design metrics

- Metriche comuni (continua)
  - **Time-to-prototype**: il tempo per realizzare una versione prototipale funzionante del sistema
  - **Time-to-market**: il tempo richiesto per lo sviluppo e la vendita ai clienti del sistema
  - **Reliability *R(t)* =** probabilità che il sistema stia lavorando correttamente all'istante t assunto che stava lavorando all'istante *t*=0
  - **Maintainability *M(d)* =** probabilità che il sistema continui a lavorare correttamente *d* unità di tempo dopo un errore.
  - **Availability *A*(*t*)**: probabilità che il sistema stia lavorando al tempo *t*
  - **Safety**: nessun danno deve essere provocato
  - **Security**: comunicazione confidenziale e autenticata

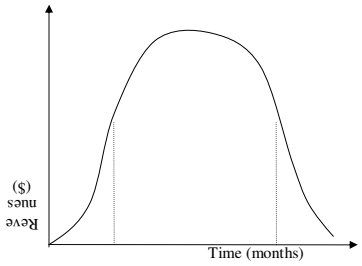
Competizione tra metriche – migliorare una metrica può peggiorare le altre



- Esperienza sia con il **software** sia con **hardware** è necessaria per ottimizzare le metriche di progetto
  - Non solo esperto di hardware o esperto di software, come di norma avviene
  - Un designer deve avere confidenza con diverse tecnologie per potere scegliere quella migliore per una data applicazione e certi vincoli di progetto

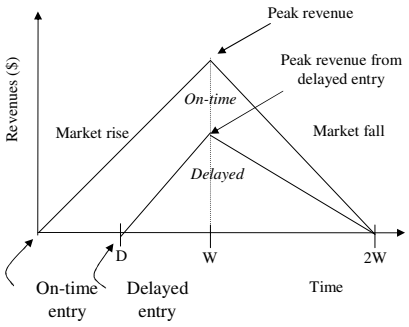


Time-to-market



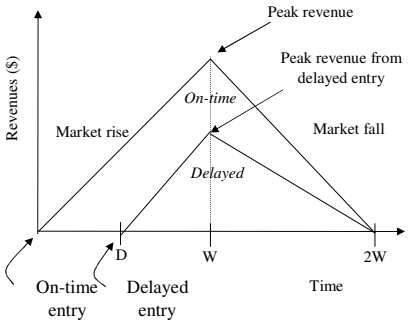
- Il tempo richiesto per lo sviluppo e la commercializzazione di un prodotto
- Market window
  - Periodo in cui potrebbe avere le vendite maggiori
- Il time-to-market medio richiesto è di circa 7-8 mesi
- I ritardi possono essere molto costosi

Perdite dovute a un ingresso ritardato nel mercato



- Modello semplificato del reddito
  - Tempo di vita di un prodotto = 2W, picco a W
  - L'ingresso nel mercato definisce un triangolo che rappresenta la penetrazione del mercato
  - L'area del triangolo rappresenta il reddito
- Perdità
  - La differenza tra le le aree dei due triangoli, quello dell'immissione in tempo e quello dell'immssione ritardata

Perdite dovute a un ingresso ritardato nel mercato



- Area = 1/2 * base * altezza
  - In tempo = 1/2 * 2W * W
  - Ritardato = 1/2 * (W-D+W)*(W-D)
- Percentuale di reddito perso = $(D(3W-D)/2W^2)*100\%$
  - Tempo di vita= 2W=52 settimane, ritardo D=4 settimane
  - (4*(3*26 –4)/2*26^2) = 22%
  - Tempo di vita 2W=52 settimane, ritardo D=10 settimane
  - (10*(3*26 –10)/2*26^2) = 50%

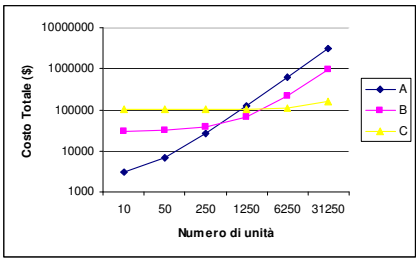
NRE e unit cost

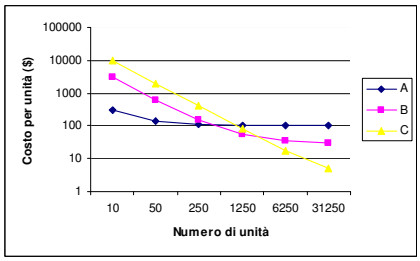
- Costi:
  - Unit cost: il costo economico per la produzione di ogni copia del sistema, escluso NRE cost
  - NRE cost (Non-Recurring Engineering cost): il costo di progettazione del sistema, da conteggiare una sola volta
  - *Costo totale = NRE cost + unit cost * numero di unità*
  - *Costo per prodotto = costo totale / numero di unità*
    = (NRE cost / numero di unità) + unit cost
- Esempio
  - NRE=$2000, unit cost=$100
  - Per 10 unità
    - Costo totale = $2000 + 10*$100 = $3000
    - Costo per prodotto = $2000/10 + $100 = $300

Ammortizzare l' NRE cost sulle unità comporta un costo aggiuntivo per unità di $200

NRE e unit cost

- Confronto delle tecnologie sulla base dei costi – la migliore dipende dalle quantità prodotte
  - Tecnologia A: NRE=$2,000, unit=$100
  - Tecnologia B: NRE=$30,000, unit=$30
  - Tecnologia C: NRE=$100,000, unit=$2





- Ma bisogna anche considerare il time-to-market

Le performance

- Misure del sistema abbondantemente utilizzate
  - Clock frequency, instructions per second – non sono delle buone misure
  - Esempio: Digital camera – un utente vuole conoscere quanto velocemente un immagine viene processata, e non la frequenza del clock o il numero di istruzioni al secondo
- Latenza (response time)
  - Il tempo tra l'inizio e la fine di un task
  - Esempio: la digital camera processa un immagine in 0,25 secondi
- Throughput
  - Tasks per second. Esempio. La digital camera processa 4 immagini al secondo

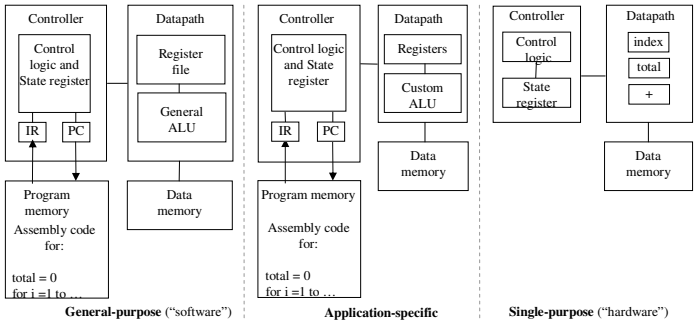
Design technologies

Tre tecnologie chiave nei sistemi embedded

- Tecnologia
  - Un modo di realizzare un task, usando processi, metodi o conoscenza
- Tecnologie per I sistemi embedded
  - Processor technology
  - IC technology
  - Design technology

Processor technology

- L'architettura dell'unità di computazione utilizzata per implementare la funzionalità richiesta al sistema
- I processori non devono essere necessariamente programmabili
  - "Processore" non vuol dire "general-purpose processor"



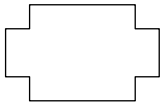
Processor technology

- I processori variano in funzione del problema

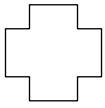
Desired functionality

```
total = 0;
for (i = 0; i< N; i++)
  total += M[i];
```

General-purpose processor



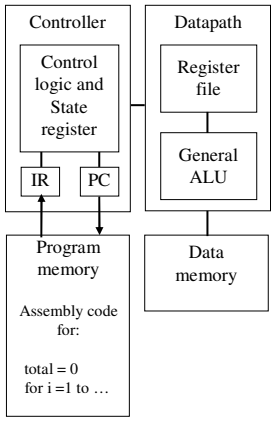
Application-specific processor



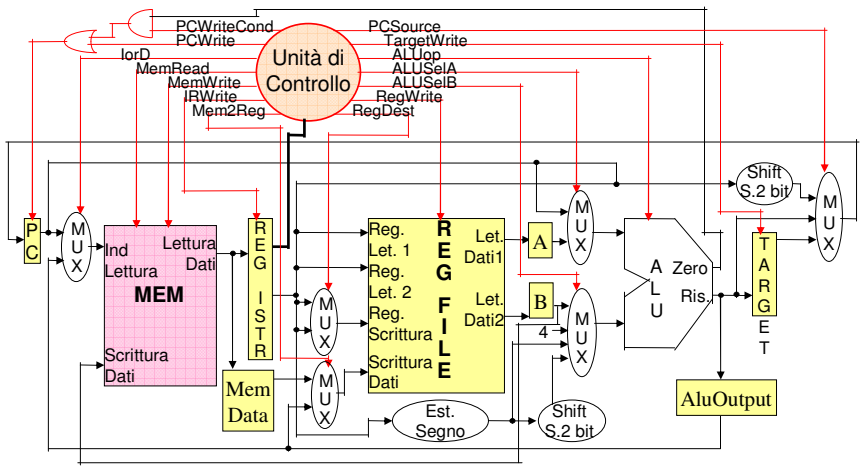
Single-purpose processor

General-purpose processors

- Dispositivi programmabili usati per un ampio spettro di applicazioni
  - Noti come “microprocessori”
- Caratteristiche
  - Program memory
  - General Datapath con ampio register file e ALU di tipo generale
- Vantaggi
  - Basso time-to-market e NRE costs
  - Alta flessibilità
- Svantaggi
  - Alti costi per unità per grandi volumi
  - Le prestazioni potrebbero essere ridotte

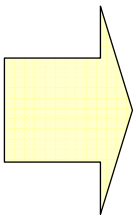


General-purpose processors il DLX sequenziale



General-purpose processors

```
total = 0;
for (i = 0; i< N; i++)
  total += M[i];
```



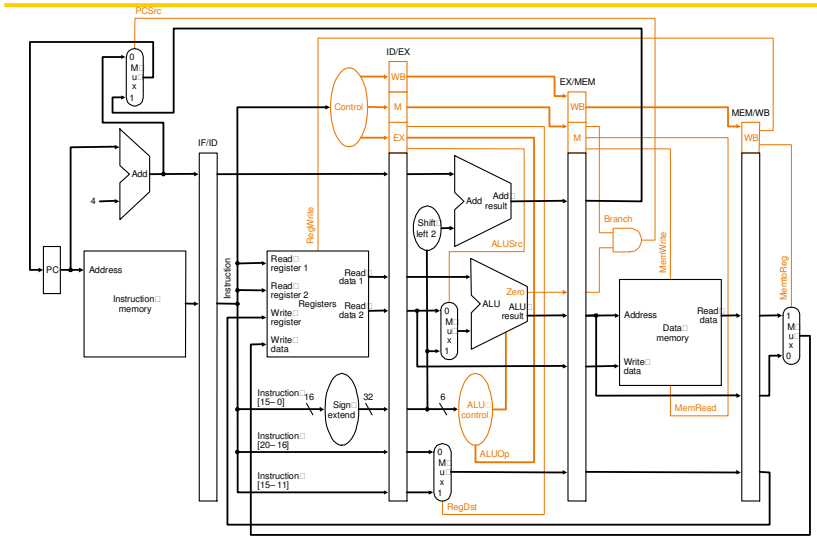
```
        addi r1, r0, 0
        addi r3, r0, 0
Loop:   lw r4,M(r1)
        addi r1,r1,4
        slti r2, r1, 40
        add r3,r3,r4
        bnez r2, loop
```


Pipelining: Increasing Instruction *Throughput*

T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11
IFetch 0	IDec 0	IExe 0	IMem 0	IWrB 0	IFetch 1	IDec 1	IExe 1	IMem 1	IWrB 1	IFetch 2	IDec 3

T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11
IFetch 0	IDec 0	IExe 0	IMem 0	IWrB 0							
	IFetch 1	IDec 1	IExe 1	IMem 1	IWrB 1						
		IFetch 2	IDec 2	IExe 2	IMem 2	IWrB 2					
			IFetch 3	IDec 3	IExe 3	IMem 3	IWrB 3				
				IFetch 4	IDec 4	IExe 4	IMem 4	IWrB 4			
					IFetch 5	IDec 5	IExe 5	IMem 5	IWrB 5		
						IFetch 6	IDec 6	IExe 6	IMem 6	IWrB 6	
							IFetch 7	IDec 7	IExe 7	IMem 7	IWrB 7

General-purpose processors
il DLX pipeline

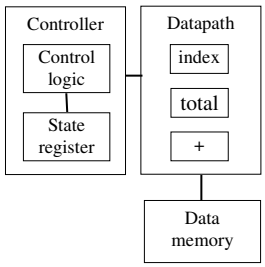


Superscalar and VLIW Architectures

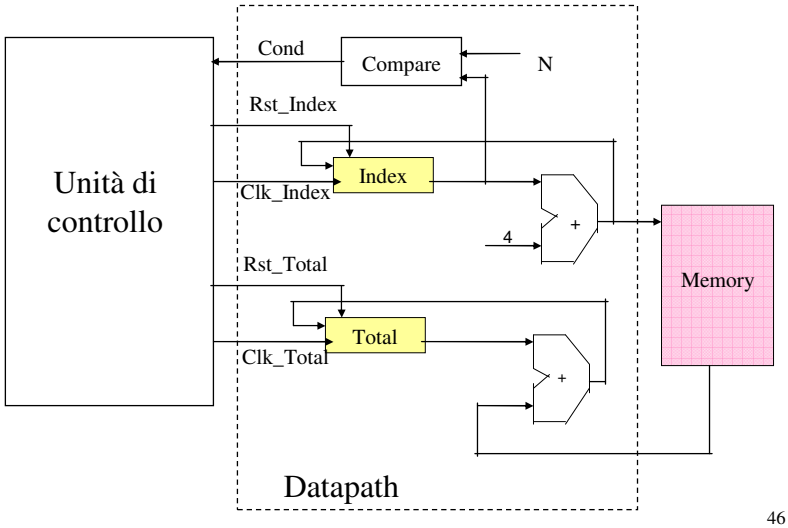
- Performance can be improved by:
  - Faster clock (but there's a limit)
  - Pipelining: slice up instruction into stages, overlap stages
  - *Multiple ALUs* to support more than one instruction stream
    - Superscalar
      - Scalar: non-vector operations
      - Fetches instructions in batches, executes as many as possible
        - May require extensive hardware to detect independent instructions
    - VLIW: each word in memory has multiple independent instructions
      - Relies on the compiler to detect and schedule instructions
      - Currently growing in popularity

Single-purpose processors

- Sistema digitale progettato per eseguire un solo programma
  - Es. coprocessori, acceleratori o periferiche
- Caratteristiche
  - Contengono i soli componenti che servono per eseguire un solo programma
  - Non hanno la program memory
- Vantaggi
  - Veloce, Low power, Piccola dimensione
- Svantaggi
  - Assenza di flessibilità, alto time-to-market, alto costo NRE

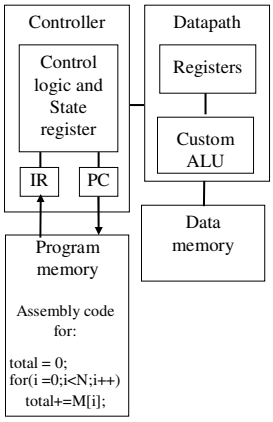


Single-purpose processors



Application-specific processors

- Processori programmabili ottimizzati per una particolare classe di applicazioni
  - Compromesso tra general-purpose e single-purpose processors
- Caratteristiche
  - Program memory, Datapath ottimizzato, Unità funzionali speciali
- Vantaggi
  - Buona flessibilità, buone performance, size e power
- Svantaggi
  - Alto costo NRE (processore e compilatore)



Esempi: Microcontrollori, DSP

A Common ASIP: Microcontroller

- For embedded control applications
  - Reading sensors, setting actuators
  - Mostly dealing with events (bits): data is present, but not in huge amounts
  - e.g., VCR, disk drive, digital camera (assuming SPP for image compression), washing machine, microwave oven
- Microcontroller features
  - On-chip peripherals
    - Timers, analog-digital converters, serial communication, etc.
    - Tightly integrated for programmer, typically part of register space
  - On-chip program and data memory
  - Direct programmer access to many of the chip's pins
  - Specialized instructions for bit-manipulation and other low-level

Digital Signal Processors (DSP)

- For signal processing applications
  - Large amounts of digitized data, often streaming
  - Data transformations must be applied fast
  - e.g., cell-phone voice filter, digital TV, music synthesizer
- DSP features
  - Several instruction execution units
  - Multiple-accumulate single-cycle instruction, other instrs.
  - Efficient vector operations – e.g., add two arrays
    - Vector ALUs, loop buffers, etc.

Microcontrollore: Famiglia ST6

- Microcontrollore a 8 bit

  - **Memories**
    Up to 4 Kbytes of program memory OTP/ROM
    Up to 64 bytes of RAM
  - **I/O Ports**
    Up to 20 I/O lines
    Multifunctional, bi-directional I/O pins
    Up to 4 high current capability I/O line
  - **Clock, Reset and Power Supply**
    Power supply operating range: 3.0V to 6V
    Maximum external frequency: 8 MHz
    Oscillator Safeguard (OSG) and Backup oscillator (LFAO)
    Low Voltage Detector (LVD)
    2 power saving modes: WAIT and STOP
  - **Interrupts**
    4 interrupt vectors plus NMI and RESET
    Software programmable for each I/O
- **I/O Ports**
  Up to 20 I/O lines
  Multifunctional, bi-directional I/O pins
  Up to 4 high current capability I/O line
- **Peripherals**
  Watchdog timer
  8-bit timer
  ADC
- **Instruction Set**
  8-bit accumulator-based architecture
  40 instructions
  9 addressing modes

Microcontrollore: STR7(ARM7TDMI® core)

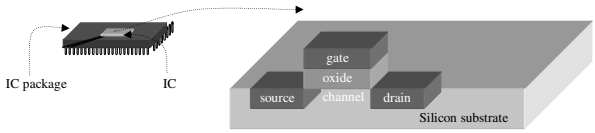
- STR710F Flash Microcontrollers from STMicroelectronics combine the industry standard ARM7TDMI® RISC microprocessor with embedded Flash and powerful peripheral functions including, USB and CAN.

La famiglia ARM7

- **L'ARM7 è un processore RISC a 16/32 bit,** prodotto, a partire dal 1993, dalla "Advanced Risc Machines Ltd."
- **È disponibile in una ampia gamma di versioni,** dedicate principalmente ad applicazioni di tipo **"embedded".**
- **Tra queste, ci sono numerosi mC prodotti da** almeno 10 diversi costruttori tra i quali STM, Motorola, NEC, Atmel, Philips, Sharp, TI.
- **Si tratta quindi di un vero e proprio standard** industriale.

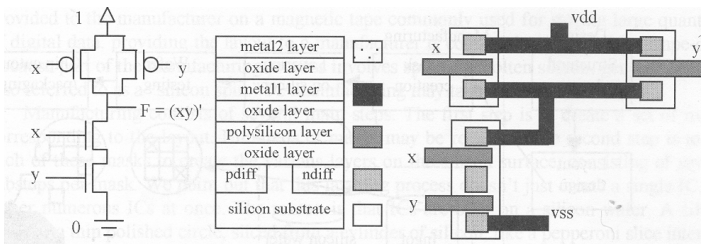
Integrated circuit (IC) technology

- Il modo in cui una implementazione digitale (gate-level) è mappata su un IC
  - IC: Integrated circuit, o “chip”
  - IC sono costituiti da diversi layer
    - I layer più bassi formano i transistor
    - Quelli intermedi formano i componenti logici
    - Quelli più alti collegano i componenti con i wires
  - Le tecnologie differiscono nel livello di specializzazione rispetto a un design



IC technology

Porta NAND



IC technology

- Tre tipi di IC technologies
  - Full-custom/VLSI
  - Semi-custom ASIC (gate array and standard cell)
  - PLD (Programmable Logic Device)

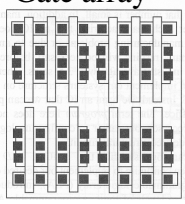
Full-custom/VLSI

- Tutti i layer sono ottimizzati per una particolare implementazione del sistema embedded
  - Placing transistors
  - Sizing transistors
  - Routing wires
- Benefici
  - Eccellenti performance, piccola dimensione, ridotta potenza
- Svantaggi
  - Elevato NRE, lungo time-to-market

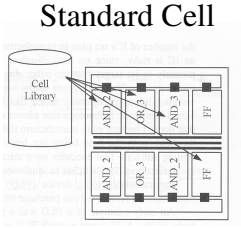
Semi-custom

- I layer più bassi sono in tutto o in parte realizzati
  - I designers devono occuparsi dei livelli più alti
- Benefici
  - Buone performance, buona dimensione,minore NRE rispetto a una implementazione full-custom
- Svantaggi
  - Richiede ancora settimane o mesi per lo sviluppo

Gate array



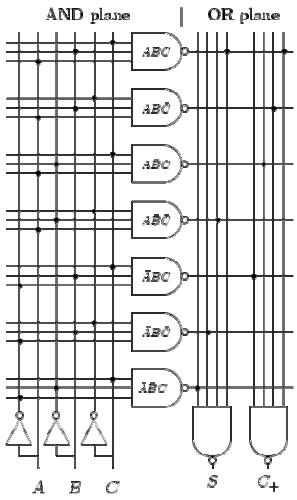
Standard Cell



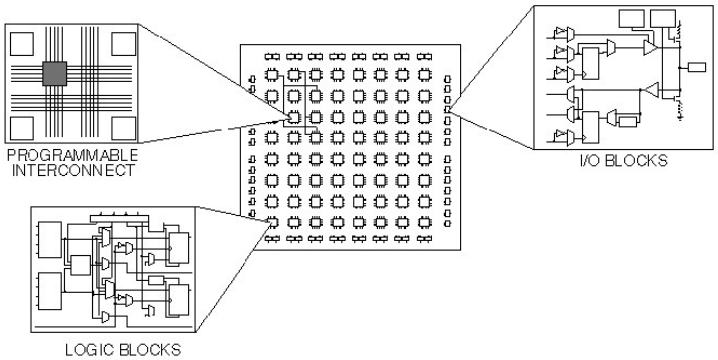
PLD (Programmable Logic Device)

- Tutti i layers già esistono
  - I designers possono comprare un IC
  - Le connessioni sull' IC sono o create o distrutte per implementare la funzionalità richiesta
  - Due tipi:
    - Programmable Logic Array (PLA)
    - Field-Programmable Gate Array (FPGA)
- Benefici
  - Basso NRE, la dispobibilità dell'IC è quasi immediata
- Svantaggi
  - Più grande, più costoso, consuma molto, più lento

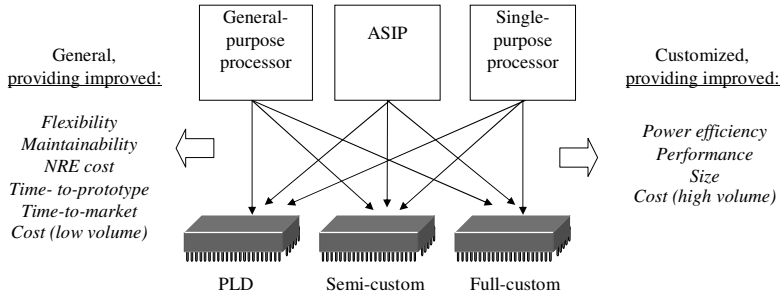
Programmable Logic Array (PLA)



Xilinx FPGA



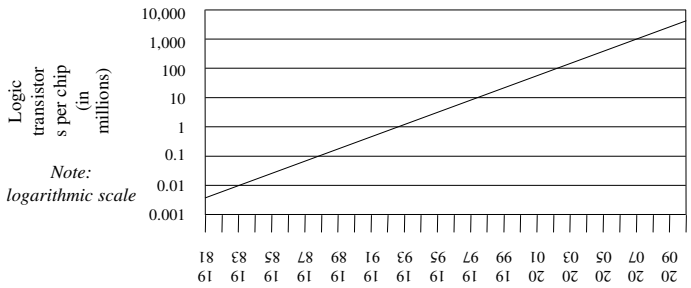
Indipendenza tra processore e IC technologies



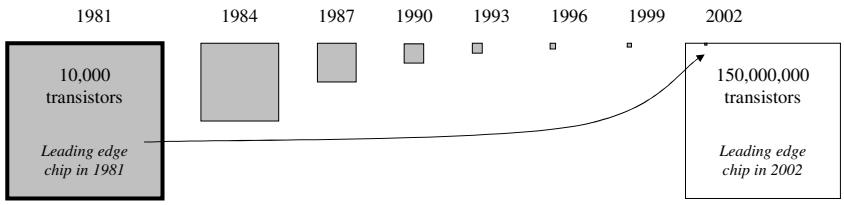
Legge di Moore

- Il più importante trend nei embedded systems
  - Predetto in 1965 dal cofondatore della Intel, Gordon Moore

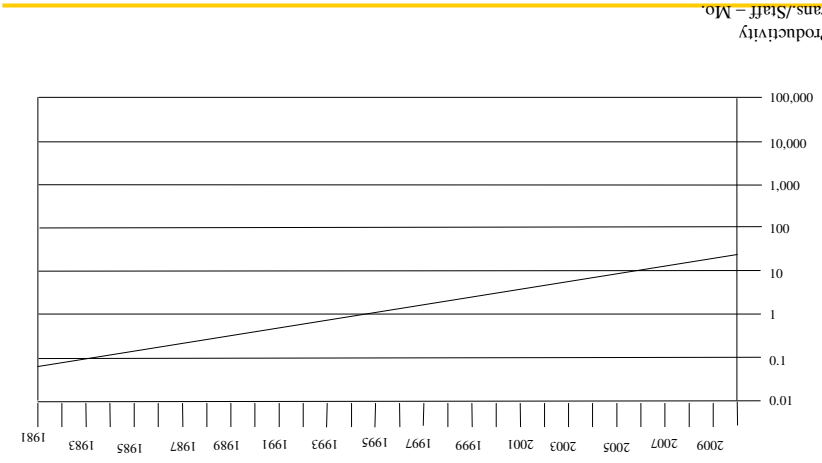
La capacità dei transistor raddoppierà ogni 18 mesi per le prossime decadi



Illustazione grafica della legge di Moore

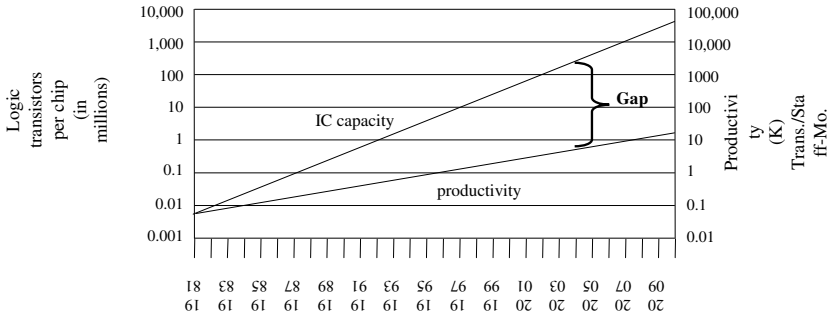


La produttività aumenta in modo esponenziale



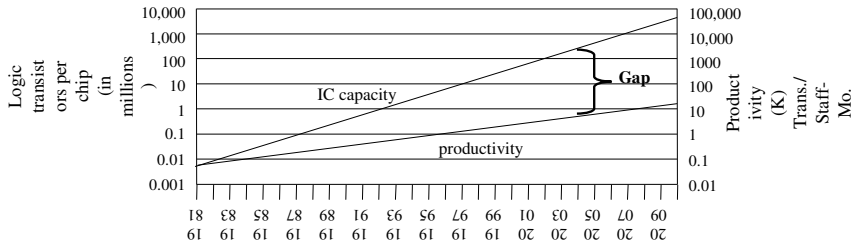
Design productivity gap

- Sebbene la capacità produttiva dei progettisti sia aumentata in modo impressionante, non ha tenuto il passo dell'incremento della capacità dei chip



Design productivity gap

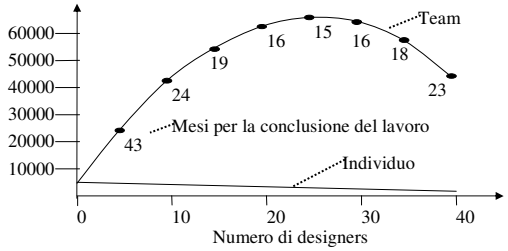
- 1981 Il chip più avanzato richiedeva 100 mesi uomo
  - 10,000 transistors / 100 transistors/mese
- 2002 Il chip più avanzato richiedeva 30,000 mesi uomo
  - 150,000,000 / 5000 transistors/mese



Il mese-uomo

- La situazione è peggiore di ciò che mostra il productivity gap
- In teoria l'aumento del numero di progettisti dovrebbe ridurre i tempi di completamento di un progetto. In realtà, la produttività del singolo progettista diminuisce a causa della complessità nella gestione e comunicazione nel gruppo di lavoro.
- In alcune casi, un incremento di persone potrebbe allungare il tempo di realizzazione di un progetto.

- 1M transistors, 1 designer=5000 trans/mese
- Ogni designer in più riduce di 100 trans/mese
- 2 designer producono 4900 trans/mese ciascuno



Gestione della crisi della produttività

- IP (Intellectual Property) Reuse
  - Assemblaggio di componenti pre-progettati
  - Componenti proprietari o provenienti dall'esterno
  - Soft e Hard IPs
- System-Level Design
  - Progetto e verifica a livello di sistema anziché a livello RTL o di gate

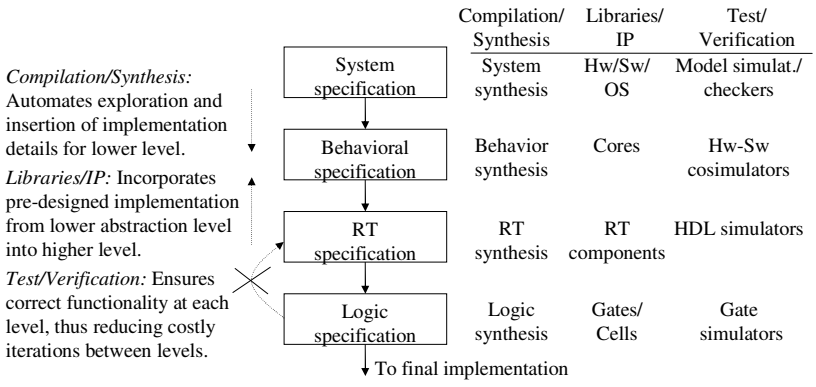
Design Technology

- E' il modo in cui si traduce la funzionalità richiesta del sistema in una sua implementazione
- Una procedura per progettare un sistema

- Il flusso di progettazione può essere parzialmente o totalmente automatizzato mediante un insieme di tools.
  - Software engineering tools,
  - Compilers,
  - Computer-Aided Design tools,

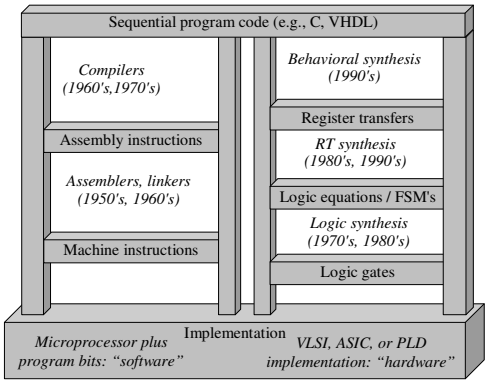
- Un'adeguata metodologia di progettazione aiuta la gestione del processo di design e migliora la qualità, le performance e i costi del progetto.

Design Technology



Il co-design

- In passato:
  - Hardware e software design technologies erano molto differenti
  - La recente maturazione della sintesi permette di avere una visione unificata di hardware e software
- Hardware/software "codesign"



La scelta tra hardware e software per una particolare funzione è semplicemente un tradeoff tra varie design metrics, come performance, power, size, NRE cost, e flessibilità; non c'è una differenza sostanziale tra cosa l'hardware o il software può implementare.

Riferimenti

– *"Embedded System Design: A Unified Hardware/Software Introduction"* , Frank Vahid, Tony Givargis, John Wiley & Sons Inc., ISBN:0-471-38678-2, 2002.
– *"Computers as Components: Principles of Embedded Computer Systems Design (With CD-ROM)"*, Wayne Wolf, Morgan Kaufmann Publishers, ISBN: 1-55860-541-X, 2001
-*"Embedded System Design"* by Peter Marwedel, Kluwer Academic Publishers, ISBN: 1-4020-7690-8, October 2003