La struttura dei sistemi embedded: unità di calcolo, memorie, bus

Braione Pietro, revisione Domenico G. Sorrenti

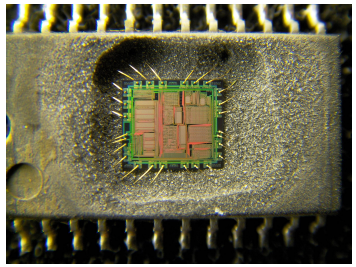
Sistemi Embedded

Obiettivi

- Analizzare la composizione di un sistema embedded
- Nelle diverse varianti tecnologiche

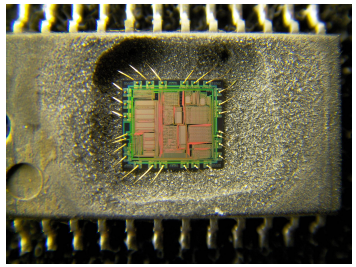
Livelli di scala

Livelli di scala

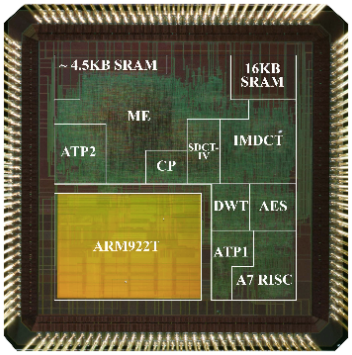


Integrated Circuit
(IC)

Livelli di scala

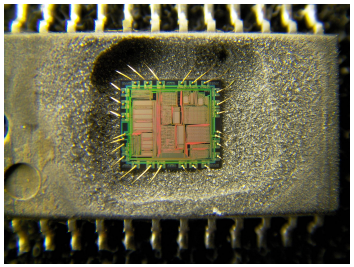


Integrated Circuit (IC)

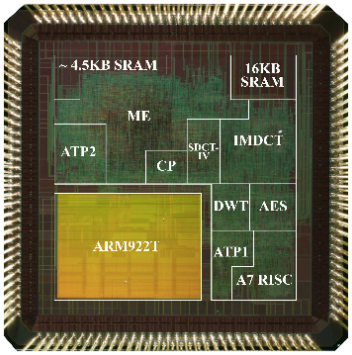


System On Chip (SOC)

Livelli di scala



Integrated Circuit (IC)



System On Chip (SOC)

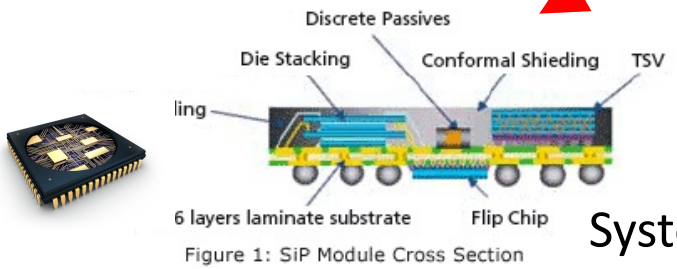
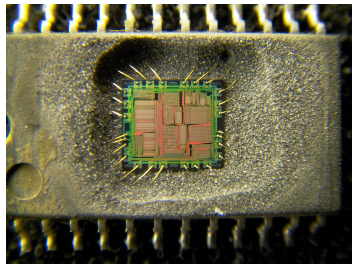


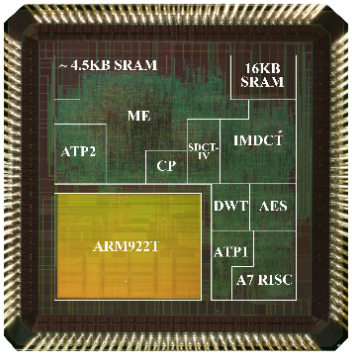
Figure 1: SiP Module Cross Section

System In Package (SIP)

Livelli di scala



Integrated Circuit
(IC)



System On Chip
(SOC)

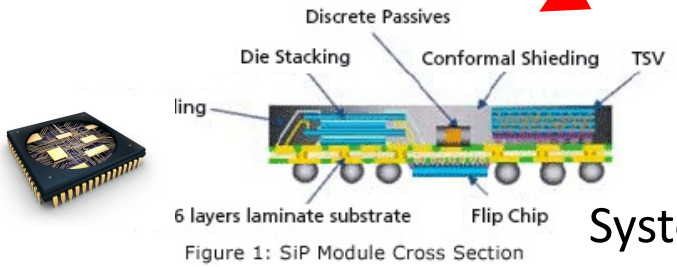
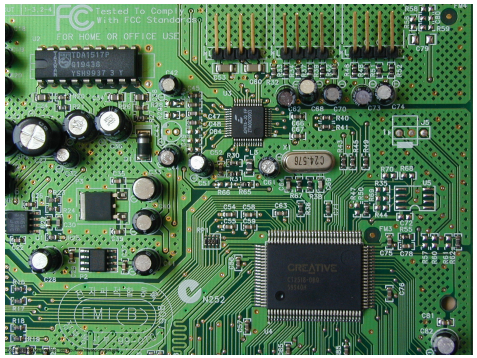


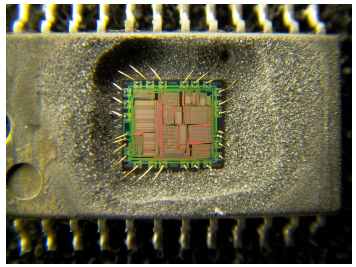
Figure 1: SiP Module Cross Section

System In Package
(SIP)

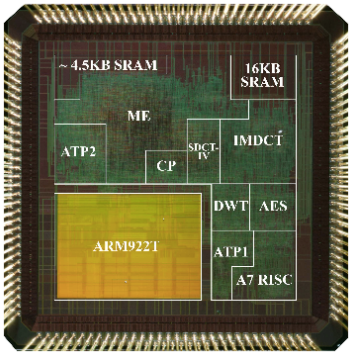


Printed circuit board
(PCB)

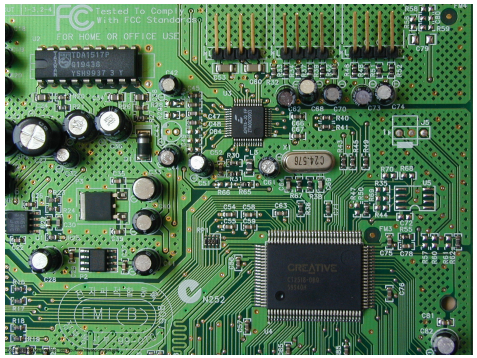
Livelli di scala



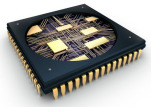
Integrated Circuit
(IC)



System On Chip
(SOC)



Printed circuit board
(PCB)



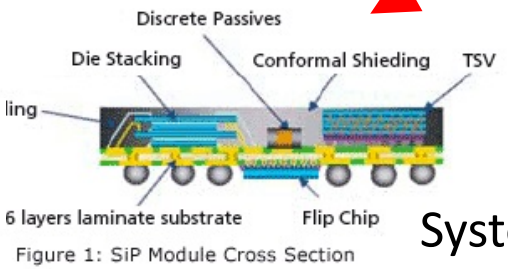
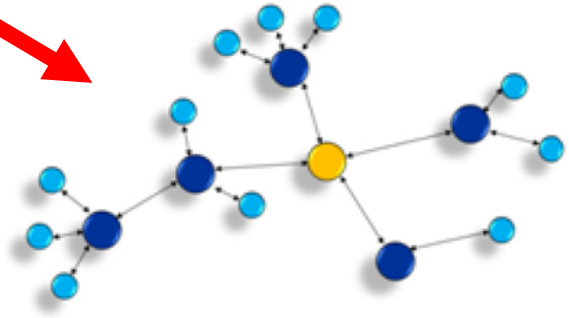


Figure 1: SiP Module Cross Section

System In Package
(SIP)



Network
(tightly – loosely coupled)

Microcontrollori

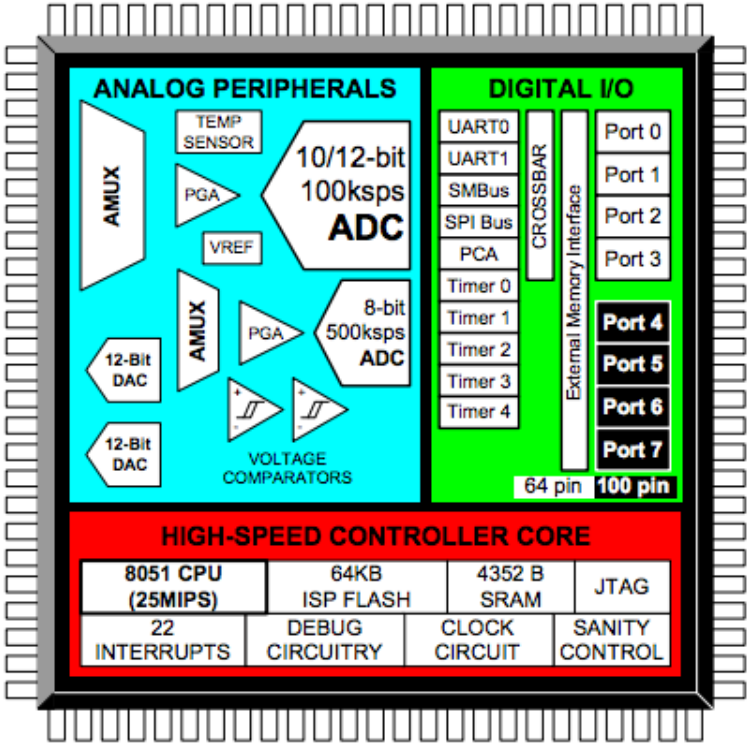
- Microcontrollore = microprocessore + memoria + periferiche su un chip singolo
- Diversi tipi di memoria (SRAM, EEPROM, FLASH) di dimensioni limitate (pochi kB – pochi MB)
- Diversi tipi di periferiche on-chip (I/O, timer, convertitori A/D e D/A, PWM, controller memoria esterna, interfacce di rete, etc.)
- Relativamente bassa velocità di calcolo (KHz – MHz, oggi meno vero)
- Bassi consumi e costi (0,25 – O(10US$) / unità)
- Possono essere a 4, 8, 16, 32 ed anche a 64 bit

Microcontrollori: esempi (1)

- EPSON S1C60N16
  - 4-bit MCU per applicazioni a bassissimi consumi (interruttori, orologi…)
  - CPU: 32 kHz – 1 MHz
  - ROM: 4 kWords (12 bit), RAM: 256 Words (4 bit)
  - Periferiche: timer, driver per LCD, comunicazione seriale, I/O digitale 8 bit
- ATMega 328P
  - 8-bit MCU usato sulla scheda Arduino UNO
  - Core: AVR, 16 MHz
  - SRAM: 2 kB, EEPROM: 1 kB, FLASH: 32 kB
  - Periferiche: I/O digitale, PWM, ADC, SPI, I2C, timer

Microcontrollori: esempi (2)

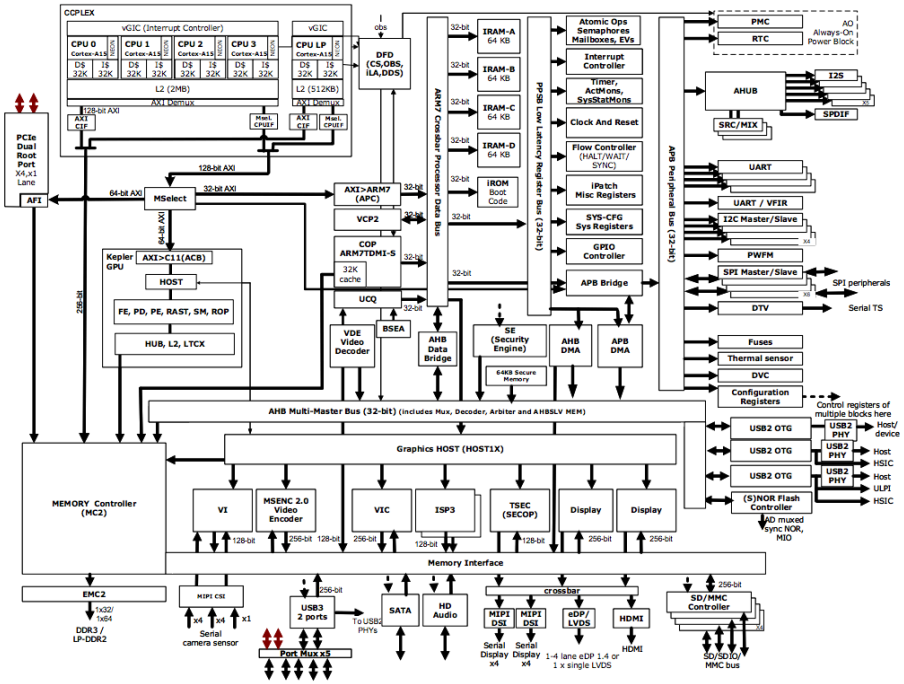
- Silicon Labs C8051F020
  - Il MCU usato nel laboratorio di questo insegnamento negli scorsi anni
  - Core: 8051 (8 bit), 25 MHz
  - SRAM: 4 kB, FLASH: 64 kB
  - Periferiche: timer, digital I/O (8 + 4 bit), I2C, SPI, UART, DAC, ADC, memoria esterna, sensore temperatura



Microcontrollori: esempi (3)

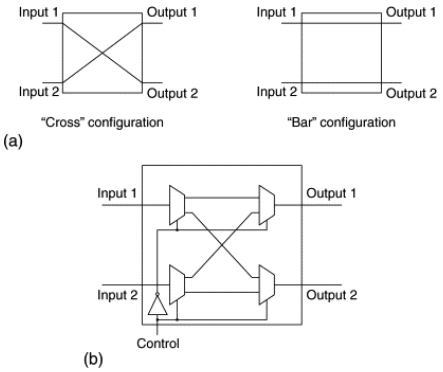
- ST33
  - Famiglia MCU per smartcards
  - Core: 32-bit ARM SC300 (derivato dal Cortex M3), 22.5 MHz
  - RAM: fino 30 kB, FLASH: fino 1.2 MB
  - Comunicazione ISO7816, SPI; coprocessori per CRC, RSA, ECC, DES…; random number generator
- STM32 F0:
  - Entry-level ARM 32 bit
  - Core: ARM Cortex M0, max 48MHz
  - SRAM: max 32 kB, FLASH: max 256 kB
  - Diverse periferiche (timer, I2C, SPI, CAN, USB, UART, USART…)
- STM32 H7:
  - Massima potenza teorica del core M7 a 32 bit
  - Core: ARM Cortex M7, 400 MHz, con istruzioni DSP e FP
  - RAM: 1 MB, FLASH: 2 MB
  - Periferiche avanzate (timer, I2C, SPI, CAN, USB, UART, USART, LCD TFT, Ethernet, S/PDIF, HDMI, camera…

Un SoC avanzato: NVIDIA Tegra K1



- Quad-core ARM processor
- GPU con 192 CUDA cores
- Audio/video decoder
- Video encoder
- Display interface
- Controller memoria con indirizzamento virtuale
- Più controller/interfacce per USB, SATA, PCIe, I2C, SPI…

NoC: motivazioni



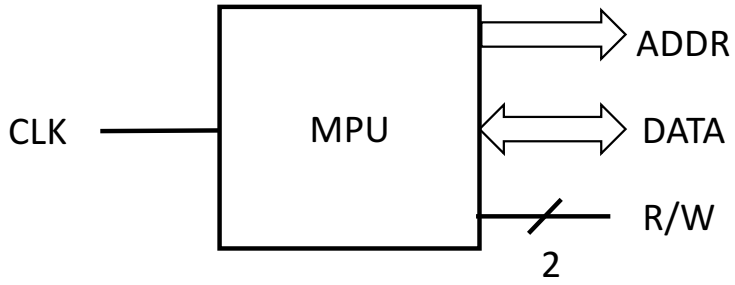
- Oggi è possibile mettere su un chip un numero molto elevato di unità (multi-cores)
- Inoltre esiste un mercato aperto per tali unità (IP cores)
- Interconnettere le unità con bus, crossbar o connessioni punto-punto o altro diventa sempre meno facile:
  - Bassa scalabilità
  - Comunicazione inaffidabile
  - Standardizzazione
- Un NoC è un sistema di interconnessione che sfrutta i criteri della comunicazione di rete:
  - Separazione strato datalink/rete/trasporto
  - Uso di routers
  - Controllo di errore/ritrasmissione
- Vantaggi:
  - Comunicazione a pacchetti
  - Scalabilità
  - Parallelismo
- Vendors: Sonics (https://sonicsinc.com/network-on-chip-noc/), Arteris (http://www.arteris.com/technology), NetSpeed systems (http://netspeedsystems.com/)

Unità di elaborazione

- General-purpose
  - Microprocessori (MPU) e microcontrollori (MCU)
  - Logiche programmabili (FPGA,…)
- Special-purpose
  - Digital Signal Processors (DSP)
  - Graphic Processing Units (GPU)
  - Application-specific integrated circuits (ASIC)

Microprocessore

- ADDR: linee indirizzo, indicano che dati vuole leggere/scrivere
- DATA: linee dati, servono per comunicare i dati
- R/W: indicano se l'MPU vuole leggere, scrivere o nessuno dei due
- CLK: segnale di clock, per sincronizzazione



Tipologie microprocessori

- CISC vs RISC
- Architettura Von Neumann vs architettura Harvard

CISC vs RISC

- ISA (Instruction Set Architecture) = l'insieme di istruzioni di un processore
- Due famiglie di ISA: CISC e RISC
- CISC = Complex Instruction Set Computer:
  - Istruzioni che eseguono operazioni complesse, anche con trasferimenti multipli da/a memoria (es. OTIR di Z80)
  - Lunghezza istruzione può variare (istruzioni più comuni più corte)
  - Pochi registri
- RISC = Reduced Instruction Set Computer:
  - Distingue istruzioni load/store (trasferimento da/a memoria) da istruzioni di manipolazione che operano solo sui registri
  - Lunghezza istruzione fissa
  - Molti registri

Perché CISC?

- L’architettura CISC è motivata dalla situazione fino alla fine degli anni settanta:
  - Programmazione in assembly, quindi necessità di istruzioni complesse per facilitare il compito
  - Memorie costose, quindi istruzioni a lunghezza variabile per ridurre l’occupazione del codice in memoria
  - Velocità processore e memoria paragonabile, quindi accessi multipli in memoria non problematici
- Ma verso la fine degli anni settanta la situazione cambia

La filosofia RISC

- Nuove tendenze a partire dalla fine degli anni settanta:
  - Diffusione compilatori e linguaggi ad alto livello: meno necessità di istruzioni assembly complesse
  - Aumenta la velocità del processore rispetto a quella della memoria: necessità di ridurre i trasferimenti da/a memoria
  - Minore costo della memoria: minore necessità di risparmiare spazio con istruzioni a lunghezza variabile
- La filosofia RISC quindi:
  - Usa una lunghezza di istruzione omogenea (per semplificare la struttura ed aumentare il clock del processore)
  - Limita i modi di indirizzamento (architettura load/store)
  - Aumenta il numero di registri per ridurre la necessità di usare la memoria
- Semplificazione dell'architettura significa anche:
  - Più spazio per registri e cache sul chip
  - Più spazio sul chip, anche per controllo del datapath in pipelining

Esempi di processori CISC e RISC

- CISC
  - x86 e x86-64 (nella maggior parte di PC e server, ma oggi “dentro” sono RISC)
  - 8051 (nelle schede usate in laboratorio negli anni passati)
- RISC
  - AVR (nelle schede Arduino)
  - ARM (ma le istruzioni Thumb fanno in parte eccezione)
  - MIPS
  - POWER

CISC vs RISC: Esempio

X86-64:

```
mov eax, DWORD PTR [rbp-4]
imul eax, DWORD PTR [rbp-4]
```


Moltiplicazione registro - memoria

ARM:

```
ldr r3, [r7, #4]
ldr r2, [r7, #4]
mul r3, r2, r3
```


Load + moltiplicazione registro - registro

Architetture di Von Neumann e Harvard

- Nell'architettura di Von Neumann il programma e i dati sono contenuti nella stessa memoria (c'è un unico spazio di indirizzamento)
- Nell'architettura Harvard vengono usate memorie distinte per programmi e dati e bus distinti per accedervi (spazi di indirizzamento distinti)
- L'architettura Harvard:
  - consente un maggior parallelismo con pipelining
  - consente una larghezze di bus diverse per dati e istruzioni
  - fornisce una maggior sicurezza (i dati non possono essere eseguiti)

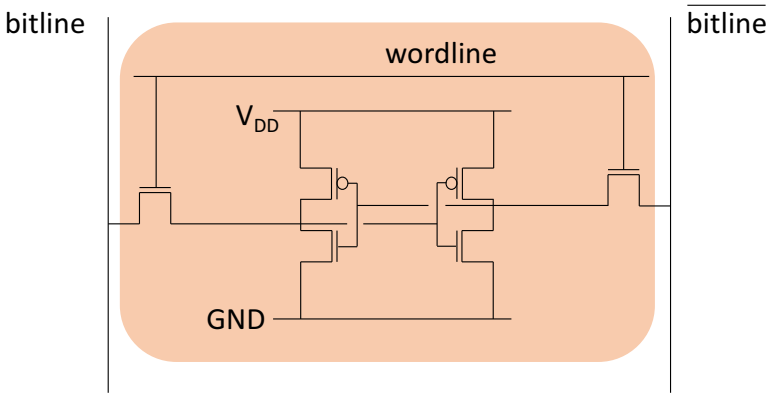
Memorie

- RWM (Read-Write Memory): il contenuto della memoria è modificabile ed è mantenuto finché questa è alimentata (sono volatili)
- NVRWM (Non-Volatile RWM): il contenuto della memoria è modificabile ed è mantenuto anche quando questa non è alimentata
- ROM (Read-Only Memory): il contenuto della memoria è cablato e non modificabile e si mantiene anche quando questa non è alimentata

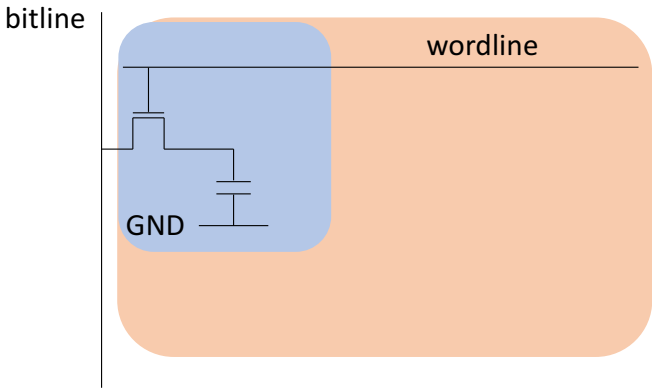
Caratteristiche memorie volatili

- SRAM (Static RAM): utilizza circuiti bistabili (flip-flop)
  - Elevatissima velocità
  - Alto consumo energetico
  - Bassa densità
  - Costi elevati
- DRAM (Dynamic RAM): utilizza carica in condensatore (necessita refresh)
  - Elevata velocità
  - Medio consumo energetico
  - Alta densità
  - Costi bassi

SRAM e DRAM



Cella RAM statica



Cella RAM dinamica

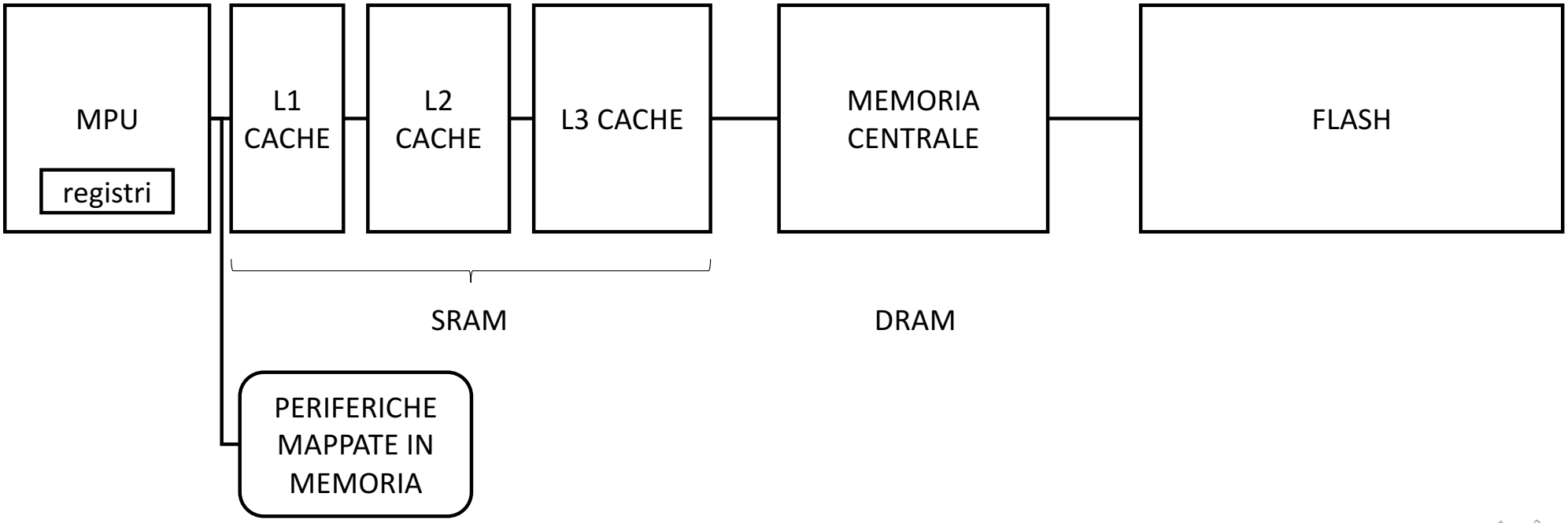
Caratteristiche memorie non volatili

- ROM (Read-Only Memory): mascherate o programmabili (PROM)
  - Alta densità
  - Basso costo
  - Affidabile
- EPROM (Electrically Programmable ROM): il contenuto della memoria è programmabile elettricamente e cancellabile con raggi UV
- EEPROM (Electrically Erasable Programmable ROM): il contenuto può essere cancellato elettricamente a livello di byte
  - Bassa densità
  - Alto costo
  - Mediamente affidabile
- FLASH: EEPROM cancellabile a blocchi, tempi di cancellazione lunghi
  - Alte densità
  - Basso costo
  - Mediamente affidabile

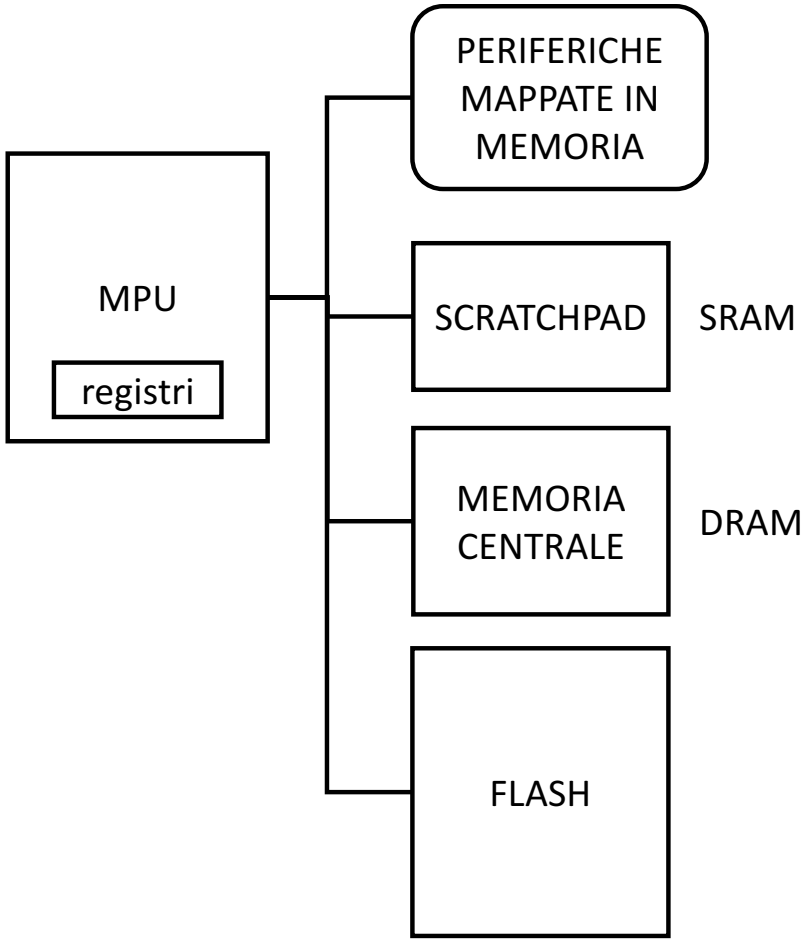
Gerarchie di memoria

- Le memorie più veloci sono più costose e a minore densità
- Risulta pertanto utile organizzare le memorie in maniera gerarchica:
  - Più “vicina” al processore la memoria più veloce e più piccola
  - Se non trovo un dato in un livello gerarchico di memoria, accedo al livello gerarchico successivo
- Sfrutta le proprietà di località del software
- Svantaggi: il tempo di accesso diviene variabile (in situazioni non real-time questo non sarebbe un problema)

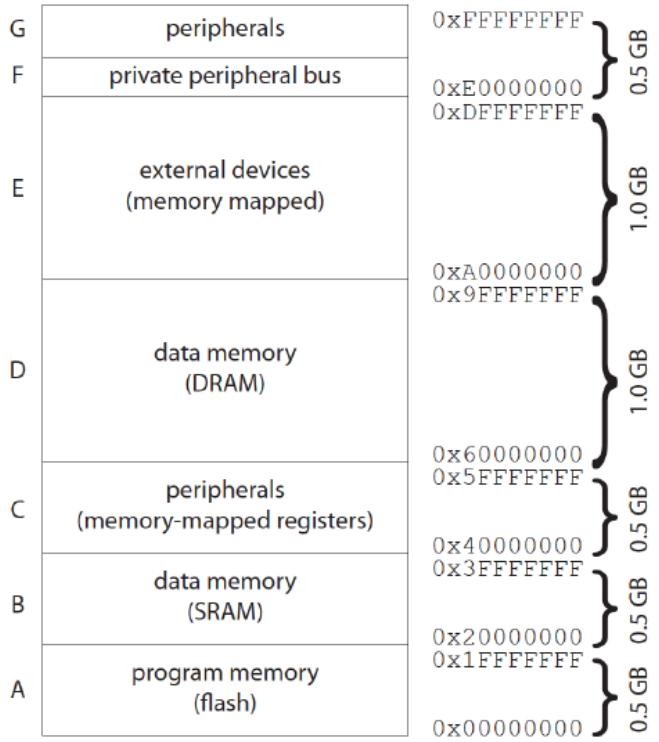
Gerarchie di memoria



Mappe memoria

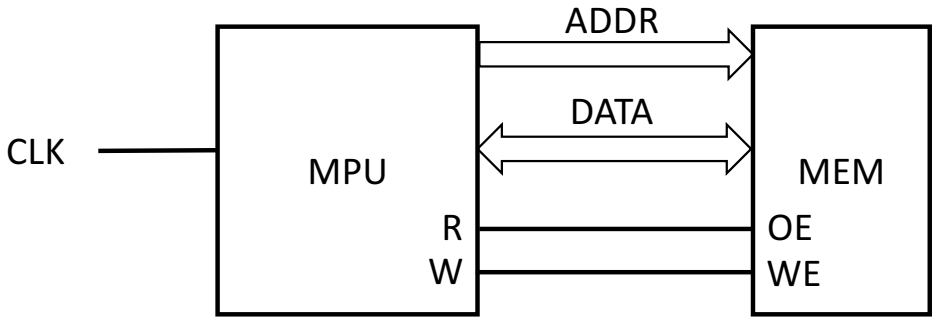


Mappe memoria



- Definisce la relazione tra indirizzi e memoria fisica / periferiche mappate in memoria
- Esempio: mappa memoria ARM Cortex M3
- Notare che la mappa della memoria non definisce quanta memoria fisica esiste!

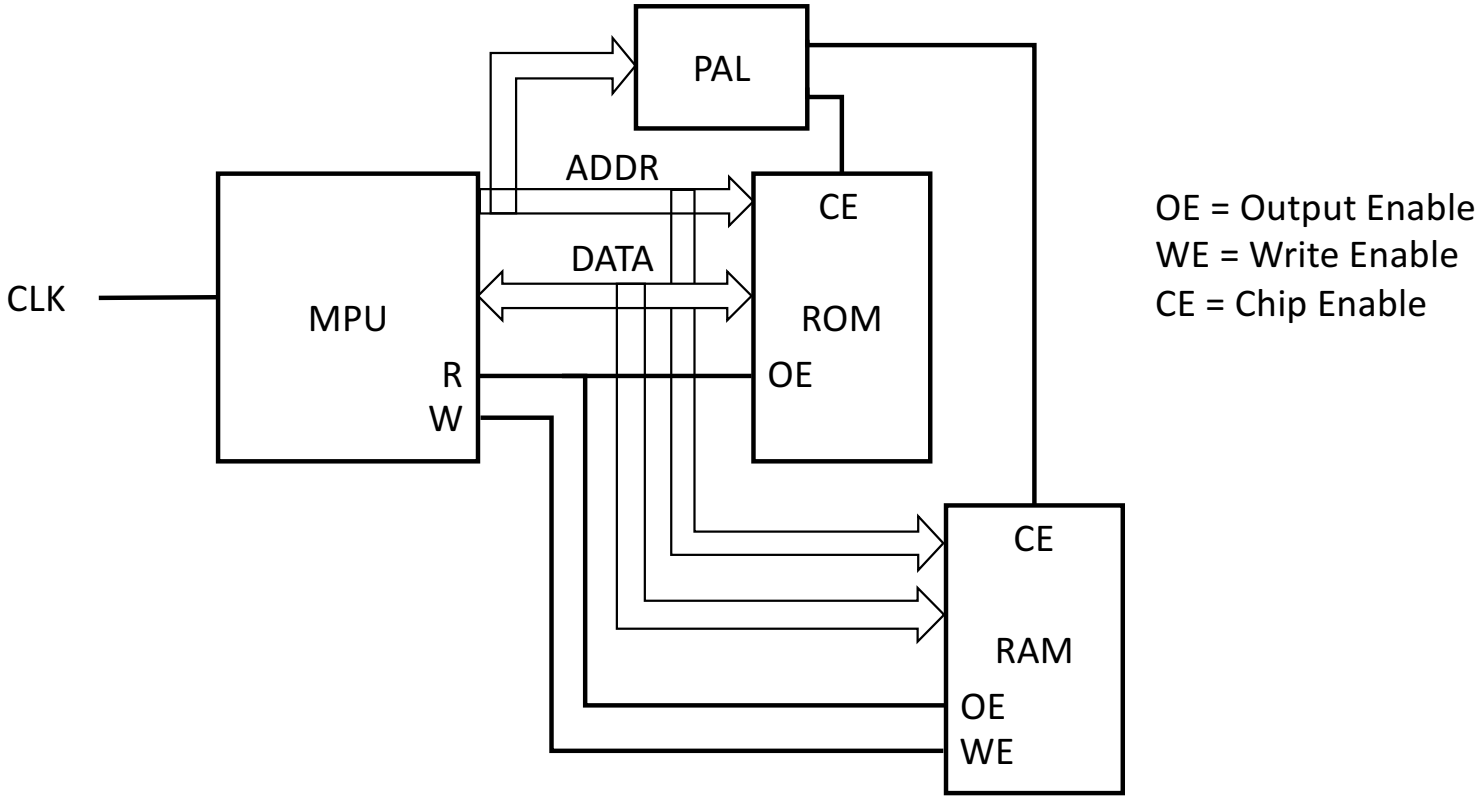
Configurazioni processore-memoria



OE = Output Enable
WE = Write Enable

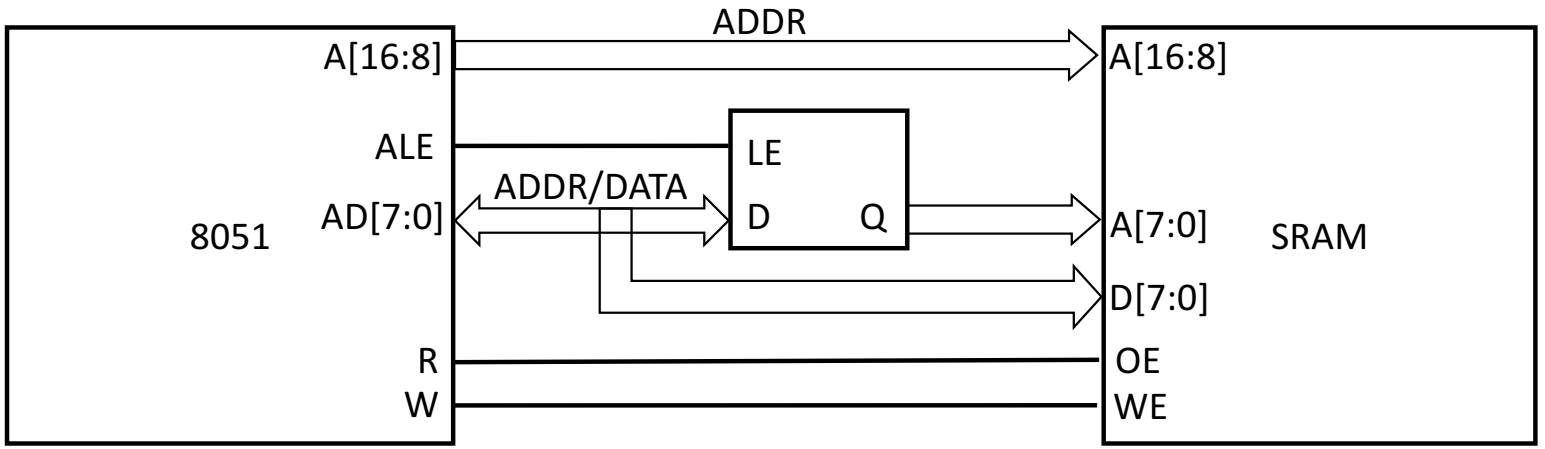
Configurazione con 1 chip di memoria

Configurazioni processore-memoria



Configurazione con più chip di memoria

Configurazione MCU 8051

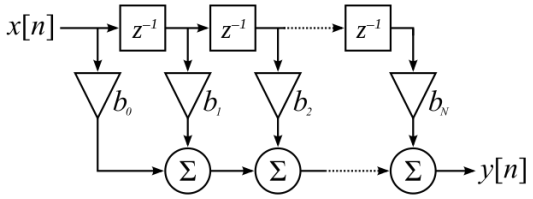


Special-purpose processors

- I processori special-purpose contengono istruzioni progettate per specifiche categorie di applicazioni
- Esempi:
  - Digital Signal Processors (DSP) = generica elaborazione di segnali digitali (es., filtraggio audio, immagini, video => MAC - Multiply And Cumulate)
  - Graphics Processing Units (GPU) = rendering grafico 3D
  - Cryptoprocessors = operazioni crittografiche
- Tipicamente CISC e Harvard architecture

Esempio: DSP (1)

- I DSP contengono istruzioni per effettuare elaborazioni di segnali digitali
- Importante categoria: filtri FIR (Finite Impulse Response)
- Ottenuti come somma pesata dell'input ritardato nel tempo (MAC)



$$y[n] = \sum_{i=0}^{N} b_i \cdot x[n-i]$$

Esempio: DSP (2)

- Per facilitare la definizione di filtri FIR il DSP della famiglia TMS320c54x (Texas Instruments) ha le istruzioni:
  - `rpt`: ripete l'instruzione successiva un certo numero di volte (zero-overhead loop)
  - `mac`: multiply-and-cumulate; ha tre argomenti e specifica l'operazione a:= a + x*y (moltiplica x e y, accumula il risultato in a)
- È così possibile realizzare un filtro FIR in esattamente N cicli clock:

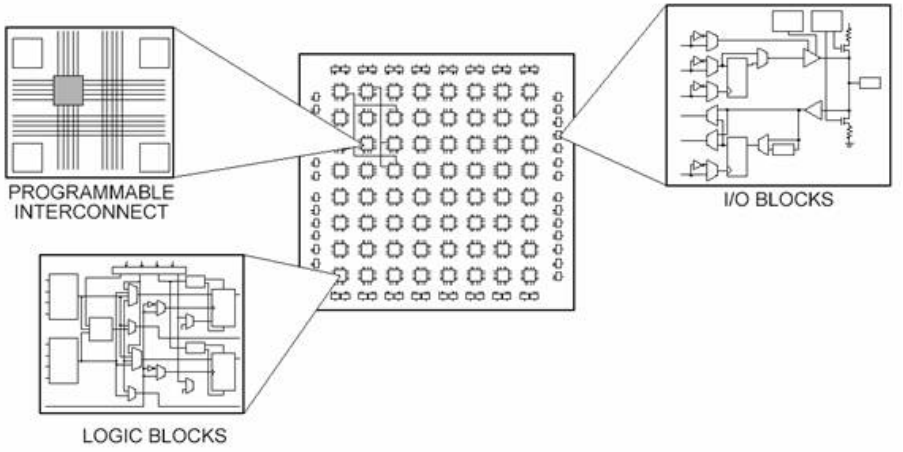
```
rpt N-1                ;ripete N volte l'istruzione successiva
mac *ar2+, *ar3+, a    ;moltiplica i valori puntati da ar2/3, accumula in a,
                       ; incrementa ar2/3
;se ar2/3 puntano ad una regione speciale di memoria on-chip, i due accessi
; in memoria vengono effettuati contemporaneamente
```

Logiche programmabili

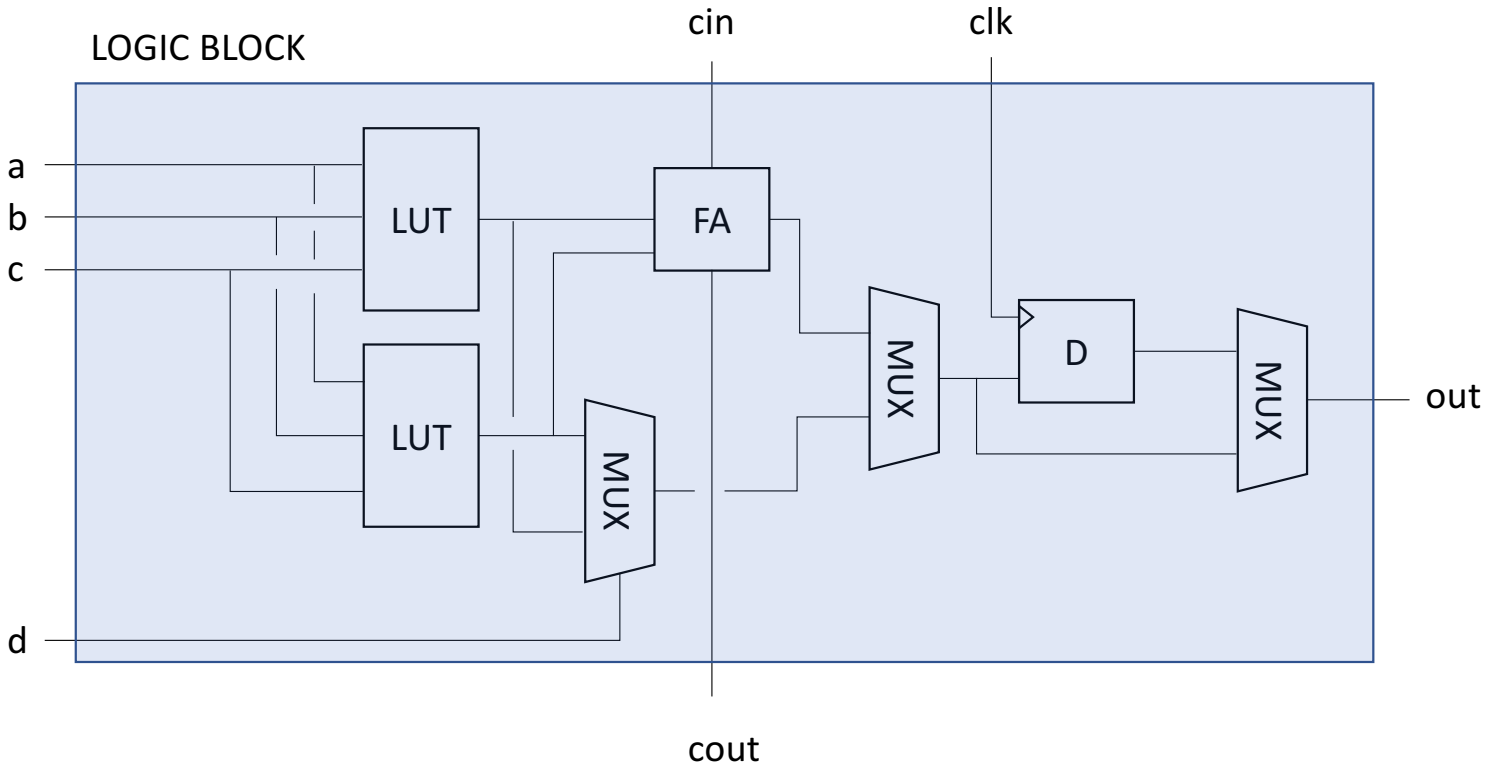
- Sono circuiti programmabili in hardware
- Composti da celle che realizzano varie funzioni
- Programmazione = stabilire interconnessioni tra celle e tra celle e pin di ingresso/uscita

Esempio: FPGA (1)

- Field Programmable Logic Arrays (FPGA) = logiche programmabili “sul campo”, ossia dopo che il chip è stato montato nel sistema
- Sono matrici di celle logiche (fino a diversi milioni) collegate da interconnessioni programmabili
- La programmazione di celle e interconnessioni avviene caricando la configurazione in una RAM



Esempio: FPGA (2)



Sviluppare per FPGA

- Linguaggi: Verilog, VHDL (ma anche linguaggi di programmazione come il C)
- Compilazione per produrre RTL netlists
- Sintesi per mappare la netlist sulle unità della FPGA
- Placing and routing per stabilire quali unità nella matrice bidimensionale utilizzare ed interconnettere
- Output: bitfile che viene caricato nella memoria della FPGA