Lezione E15

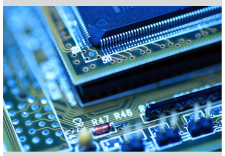
Dispositivi logici programmabili

Sistemi embedded e real-time

24 gennaio 2013

Marco Cesati

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



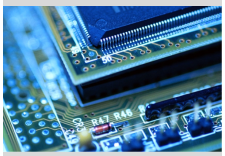
Di cosa parliamo in questa lezione?

In questa lezione si descrivono i dispositivi logici programmabili

1. Dispositivi logici programmabili
2. Struttura degli FPGA
3. VHDL

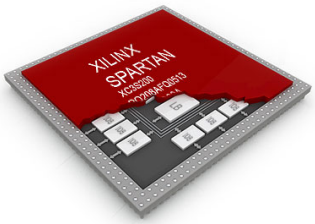
Dispositivi logici programmabili

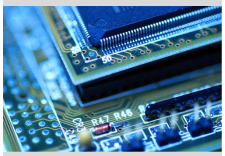
Marco Cesati



Elementi di calcolo per i sistemi embedded

- Vari tipi di elementi di calcolo per sistemi embedded:
  - ASIC
  - Microcontrollori
  - Microprocessori
  - Dispositivi logici programmabili
- Le logiche programmabili sono spesso convenienti
  - economiche
  - efficienti
  - riconfigurabili / aggiornabili





Dispositivi logici programmabili

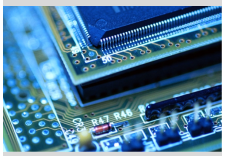
Le *logiche programmabili* (o *PLD*, Programmable Logic Device) integrano risorse logiche e linee di interconnessione

Sono chip *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

Esistono differenti gradi di *programmabilità*:

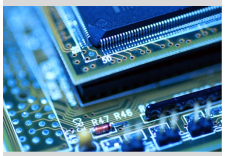
- one-time programmable (OTP): la configurazione del chip è irreversibile ed è ottenuta applicando tensioni elettriche più alte di quelle della normale alimentazione
- riprogrammabili: la configurazione può essere effettuata più volte; le interconnessioni sono transistor pilotati dai bit di un circuito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



Complessità e organizzazione dei PLD

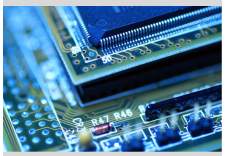
I PLD si distinguono anche in base alla complessità delle risorse logiche elementari (*celle*)

- 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
  - Vantaggio: Il silicio è maggiormente sfruttato, perché il rischio di sotto-utilizzare una cella è ridotto
- 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
  - Vantaggio: per realizzare la logica dell'applicazione occorrono meno celle complesse, perciò le interconnessioni sono più corte e facili da realizzare

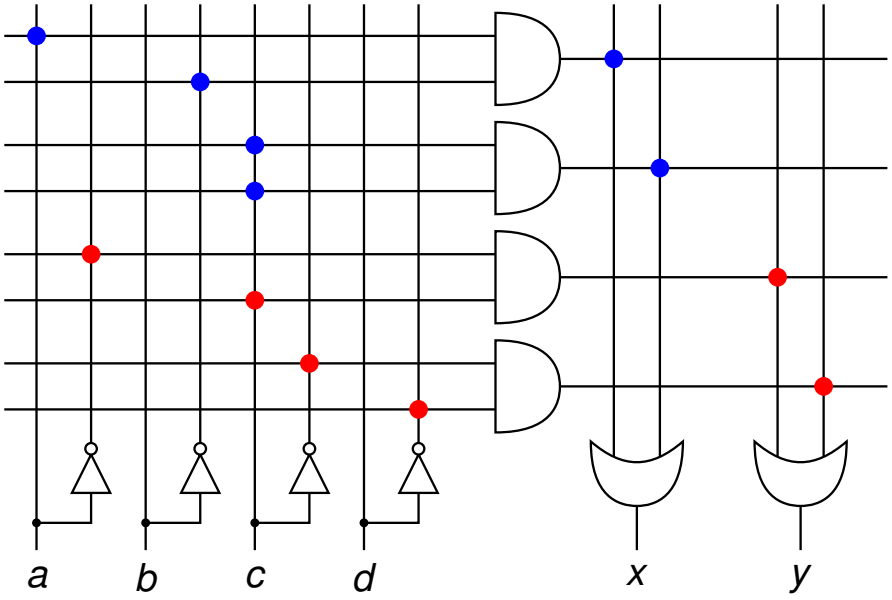


Tipologie di PLD

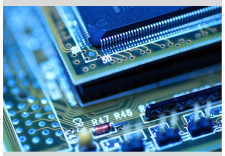
- PLA (Programmable Logic Array): costituito da zone di porte AND e OR e da aree di interconnessione configurabili (OTP)
- PAL (Programmable Array Logic): come i PLA, ma più semplici: la configurazione delle porte OR è prefissata
- GAL (Generic Array Logic): include diversi PAL, con multiplexer per retro-azionare 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
- FPGA (Field Programmable Gate Array): celle complesse e interconnessioni sono distribuite regolarmente nel chip; la configurazione è in memoria generalmente volatile



Funzionamento di un PLA



- $x = (a \wedge \overline{b}) \vee c$
- $y = (\overline{a} \wedge c) \vee (\overline{c} \wedge \overline{d})$

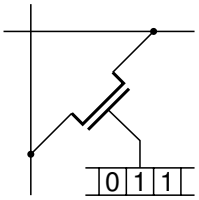


Tecnologie di interconnessione

I PLD sono programmati configurando opportunamente le connessioni elettriche del chip

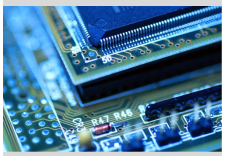
Diverse tecnologie per realizzare le interconnessioni:

- OTP: One Time Programmable
  - Fuse technology: connessioni normalmente chiuse, una corrente elevata fonde il collegamento ed apre il circuito
  - Antifuse technology: connessioni normalmente aperte, l'applicazione di una tensione elevata allinea gli atomi di silicio amorfo e li trasforma in silicio policristallino conduttore
- Riprogrammabili, Riconfigurabili
  - Ciascuna connessione è controllata da un transistor pilotato dalla linea di uscita di un bit di memoria statica, EEPROM o flash



Tipicamente i PLD più semplici sono OTP, quelli più complessi sono riprogrammabili e riconfigurabili

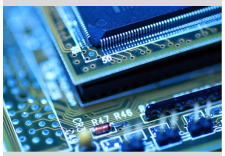
Dispositivi logici programmabili
Marco Cesati



Schema della lezione
Logiche programmabili
FPGA
VHDL

FPGA

- FPGA: Field Programmable Gate Array
- Il primo FPGA è stato costruito dalla Xilinx nel 1985
- Tradizionalmente utilizzati in
  - elaborazione di segnali digitali
  - sistemi aereospaziali
  - sistemi di difesa
  - prototipizzazione di ASIC
  - crittoanalisi
  - bio-informatica
  - calcolo scientifico
  - ...
- Maggiori venditori:
  - Xilinx (quota di mercato: > 50%)
  - Altera (quota di mercato: $\sim$ 30%)
  - Lattice Semiconductor, Actel, SiliconBlue Technologies, Achronix, QuickLogic, ...
- Diffusione sempre crescente!

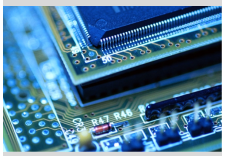


Struttura di un FPGA

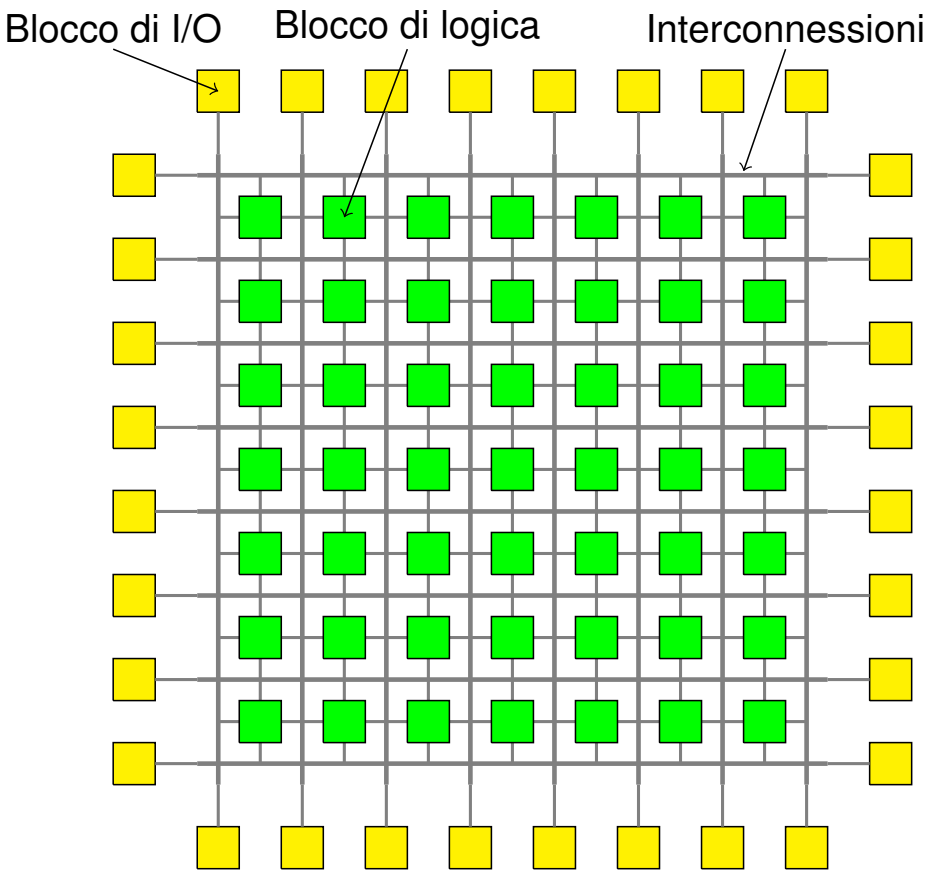
Un FPGA è costituito da:

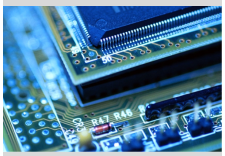
- Un insieme di interfacce di I/O digitali
- Un insieme di blocchi di logica programmabile distribuiti lungo la superfice del chip
  - Xilinx li chiama CLB (Configurable Logic Block)
  - Altera li chiama LAB (Logic Array Block)
- Unità di calcolo specializzate (ad es, moltiplicatori)
- Blocchi di RAM statica
- Collegamenti di interconnessione tra i vari elementi dell'FPGA

I dettagli variano con il venditore ed il modello



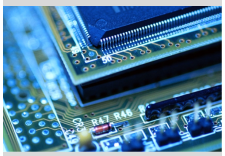
Struttura di un FPGA (2)





Gerarchie di interconnessione

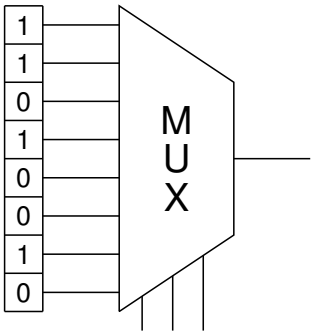
- I blocchi di logica programmabili sono distribuiti sulla superficie di silicio del chip
- Un dato può essere trasferito da un blocco ad un altro utilizzando interconnessioni programmabili
- Esiste una gerarchia di inteconnessione:
  - Blocchi logici vicini sono collegati tramite interconnessioni dotate di bassa latenza di trasmissione
  - Gruppi locali di blocchi logici sono collegati tramite interconnessioni di capacità maggiore chiamate linee di routing

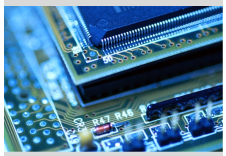


Celle logiche

Ogni blocco di logica programmabile (CLB o LAB) è costituito da tipicamente poche celle logiche

- Sono i circuiti più piccoli individualmente programmabili
- Xilinx: LC (Logic Cell), Altera: LE (Logic Element)
- Ciascuna cella è tipicamente costituita da
  - uno o due LUT (Lookup Table)
  - un circuito addizionatore con riporto
  - alcuni elementi di memoria (flip-flop)
  - alcuni multiplexer
- Una LUT con n ingressi realizza una funzione combinatoria di arità n
  - È costituita da 2^n celle SRAM (flip-flop) ed un multiplexer
  - Generalmente $n \in \{2, \ldots, 6\}$
  - I flip-flop possono essere usati anche come registro

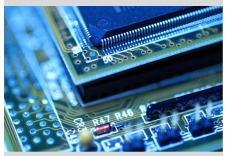




Caratteristiche costruttive di un FPGA

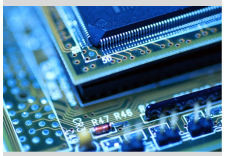
Principali caratteristiche per la scelta di un FPGA:

- Supporto fisico, ad esempio:
  - numero di pin del chip
  - integrazione su scheda con bus ad alta velocità per l'interfacciamento ad un calcolatore
- Consumo di energia e potenza
- Densità e quantità dei blocchi logici
- Quantità di RAM utilizzabile
  - contenuta nelle LUT
  - integrata nel chip come blocchi specializzati
- Funzionalità implementate in hardware
- Disponibilità di IP già pronte
- Costi:
  - del chip
  - degli strumenti di sviluppo
  - delle IP



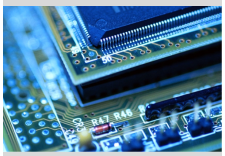
Programmazione degli FPGA

- Gli FPGA sono programmati, nel senso che lo sviluppatore definisce completamente il funzionamento del chip
- Quando si programma un microprocessore od un microcontrollore lo sviluppatore definisce le istruzioni macchina eseguite dall'unità di calcolo
  - "Programmare" significa costruire il software
- Quando si programma una dispositivo di logica programmabile (FPGA) lo sviluppatore definisce il funzionamento dei circuiti interni del dispositivo
  - "Programmare" significa definire l'hardware
- Per programmare l'hardware si utilizzano linguaggi chiamati generalmente HDL (Hardware Description Language)
- I due linguaggi HDL più diffusi:
  - VHDL (Europa, Asia)
  - Verilog (USA)



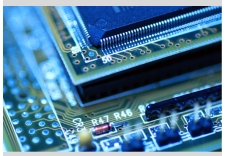
Il linguaggio VHDL

- Nasce nel 1980 per aiutare il ministero della difesa USA a documentare il funzionamento degli ASIC
- Successivamente sono stati creati dei simulatori logici che consentono di simulare l'hardware descritto da VHDL
- Ancora più tardi sono stati creati dei sintetizzatori logici in grado di produrre una descrizione fisica del circuito hardware descritto dal codice VHDL
- Concetti e sintassi del VHDL sono molto influenzati dal linguaggio di programmazione del software Ada
- Oggi il VHDL è definito dallo standard IEEE 1164 (ultima versione: VHDL-4.0 del gennaio 2009)
- Esistono inoltre numerosi sotto-standard IEEE che definiscono aspetti di interesse per applicazioni specifiche



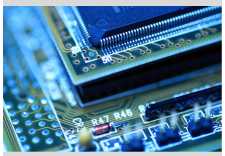
Struttura di un file VHDL

```
-- Questo e' un commento

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity UNCOMPONENTE is
    port ( ... );
end UNCOMPONENTE;

architecture NOMEARCH of UNCOMPONENTE is
...
begin
...
end NOMEARCH;
```

Struttura di un file VHDL (2)

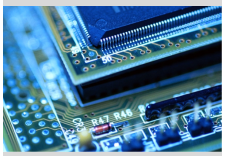
- Il linguaggio VHDL è case-insensitive
  - Nessuna differenza tra MAIUSCOLE e minuscole
- Le librerie consentono di utilizzare
  - Tipi di dati predefiniti
  - Componenti hardware standard
  - Funzioni
- La dichiarazione `entity` descrive l'interfaccia di un componente hardware: quali segnali di input e di output lo collegano agli altri componenti
- La dichiarazione `architecture` descrive come è fatto il componente hardware
- Due modalità diverse per definire l'architettura:
  - Descrizione strutturale: si descrive il circuito definendo i collegamenti tra porte logiche ed i vari componenti
  - Descrizione comportamentale: si descrive il circuito definendo il comportamento che dovrà avere



Definizione dell'interfaccia

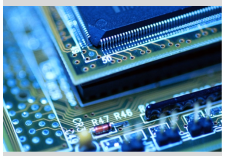
```
entity Prova is
    port(a, b, c:  IN std_logic;
         x:  OUT std_logic;
         y:  OUT std_logic;
    );
end Prova;    -- oppure end entity;
```

- Il nome del componente è `Prova` (o `PROVA`, `prova`, ...)
- Riceve tre ingressi chiamati `a`, `b` e `c` di tipo `std_logic` (segnale logico standard)
- Produce due uscite chiamate `x` e `y` di tipo `std_logic`



Tipi di dati predefiniti

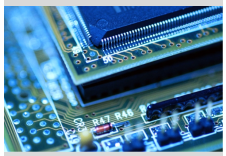
- `integer`: numero intero con segno di (almeno) 32 bit
- `natural`: numero intero senza segno ≥ 0
- `positive`: numero intero senza segno > 0
- `real`: numero in virgola mobile con mantissa e esponente
- `time`: intervallo temporale
  - valore qualificato da: `fs ps ns us ms sec min hr`
- `character`: carattere codificato con lo standard ISO 8859 Latin 1 (8 bit)
- `boolean`: valori logici `false` e `true`



Tipi di dati predefiniti (2)

- `bit`: i valori `'0'` (valore logico basso) e `'1'` (alto)
- `std_logic`: un segnale logico standard:
  - `'U'`: non inizializzato
  - `'Z'`: stato alta impedenza
  - `'0'`: forzato logico basso
  - `'1'`: forzato logico alto
  - `'X'`: valore forzato sconosciuto
  - `'W'`: debole sconosciuto
  - `'L'`: debole logico basso
  - `'H'`: debole logico alto
  - `'-'`: non importa

Un valore “debole” è usato ad esempio per lo stato di una linea derivato da resistenze di “pull-up” o “pull-down”



Vettori

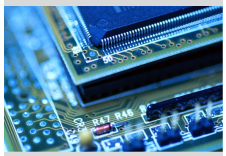
- È possibile definire vettori (`array`)
  - Ad esempio: `array (1 to 4) of boolean`
- Alcuni tipi di vettori sono predefiniti:
  - `string`: sequenza di `character`
  - `bit_vector`: sequenza di `bit`
  - `std_logic_vector`: vettore di valori logici

La direzione in cui si indicano gli indici dei vettori è importante:

`array (1 to 4) of integer` ⇒ big endian

`array (4 downto 1) of integer` ⇒ little endian

Dispositivi logici programmabili
Marco Cesati



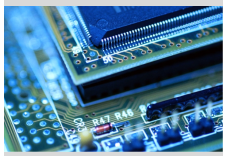
Schema della lezione
Logiche programmabili
FPGA
VHDL

Descrizione strutturale

```
architecture Strut of Prova is
    signal tmp:  std_logic;
begin
    x <= a and b;
    y <= tmp xor b;
    tmp <= c or a;
end Strut;  -- oppure end architecture;
```

- Le “istruzioni” sono “eseguite” in parallelo, non in sequenza!
- L’ordine delle “istruzioni” non è importante
- `tmp` è un segnale interno dell’architettura (una linea)

Se `a`, `b`, `c` sono eguali a `'1'`, quanto vale `y`? `'0'`



Simulazione e sintesi

Il codice VHDL può essere utilizzato per

1. Simulare il funzionamento dell’hardware
2. Sintetizzare una descrizione fisica dell’hardware

Le fasi della “compilazione”:

- Controllo della sintassi
- Sintesi della descrizione RTL del circuito
  - $\Rightarrow$ lista di componenti e collegamenti chiamata *netlist*
- Simulazione del circuito (facoltativa)
- Implementazione del progetto
  - Traduzione della netlist in un design adatto al tipo di chip
  - Mappatura del design sulle risorse a disposizione nel chip
  - Piazzamento dei componenti e instradamento dei segnali
- Generazione del flusso di bit (*bitstream*) con cui programmare il chip