Introduzione all'uso di FPGA

UNIVERSITÀ DEGLI STUDI
DI PERUGIA

July 24, 2017

Mirko Mariotti

Contents

1 Sistemi Digitali
- Vari tipi di sistemi digitali
- Programmabilità
- Logica Programmabile
- Nomenclatura

2 FPGA
- Chipmakers
- Evaluation boards
- Architettura di un FPGA
- Software

3 Programmazione
- Linguaggi
- Sintesi
- Verifica
- Simulazione

4 Verilog

UNIVERSITÀ DEGLI STUDI
DI PERUGIA

Sistemi Digitali

Mirko Mariotti

Vari tipi di sistemi digitali

- Computer (CPU, Schede grafiche, Schede di rete, ecc)
- Microcontrollori (Arduino, PICs, ecc)
- GPU
- SoCs (RaspberryPI, Ecc)

Programmabilità

- Sono tutti componenti elettronici.

Programmabilità

- Sono tutti componenti elettronici.
- Sono tutti componenti elettronici digitali.

Programmabilità

- Sono tutti componenti elettronici.
- Sono tutti componenti elettronici digitali.
- Sono tutti componenti elettronici programmabili.

Programmabilità

- Sono tutti componenti elettronici.
- Sono tutti componenti elettronici digitali.
- Sono tutti componenti elettronici programmabili.

La logica di funzionamento del singolo componente cambia ?

Programmabilità

- Sono tutti componenti elettronici.
- Sono tutti componenti elettronici digitali.
- Sono tutti componenti elettronici programmabili.

La logica di funzionamento del singolo componente cambia ?

Esiste tuttavia un'altra classe di dispositivi digitali: quelli a logica programmabile

Logica Programmabile

Field Programmable Gate Array (FPGA) è il più importante esempio di chip a logica programmabile

C'è differenza fra:

- Chip a Logica programmabile
- Chip Programmabile

Nomenclatura

- **Microprocessore** - Componente elettronica che esegue un programma generico, sono il cuore dei modermi computer.

Nomenclatura

- **Microprocessore** - Componente elettronica che esegue un programma generico, sono il cuore dei modermi computer.
- **Microcontrollore** - Componente elettronica che esegue un programma ma avente capacità ridotte rispetto ad un processore.

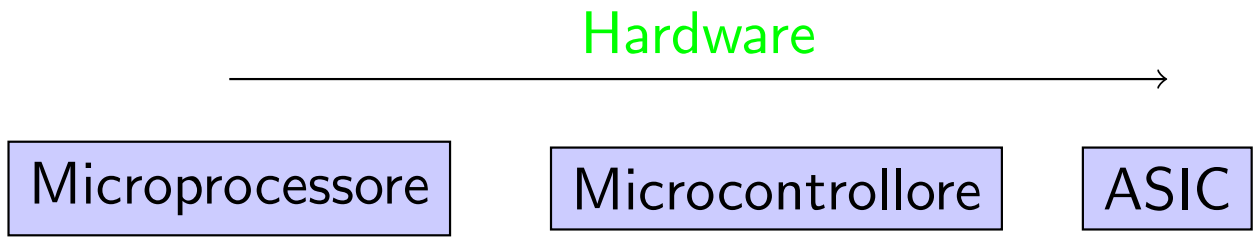
Nomenclatura

- **Microprocessore** - Componente elettronica che esegue un programma generico, sono il cuore dei modermi computer.
- **Microcontrollore** - Componente elettronica che esegue un programma ma avente capacità ridotte rispetto ad un processore.
- **ASIC** - Chip customizzato per realizzare una singola applicazione.

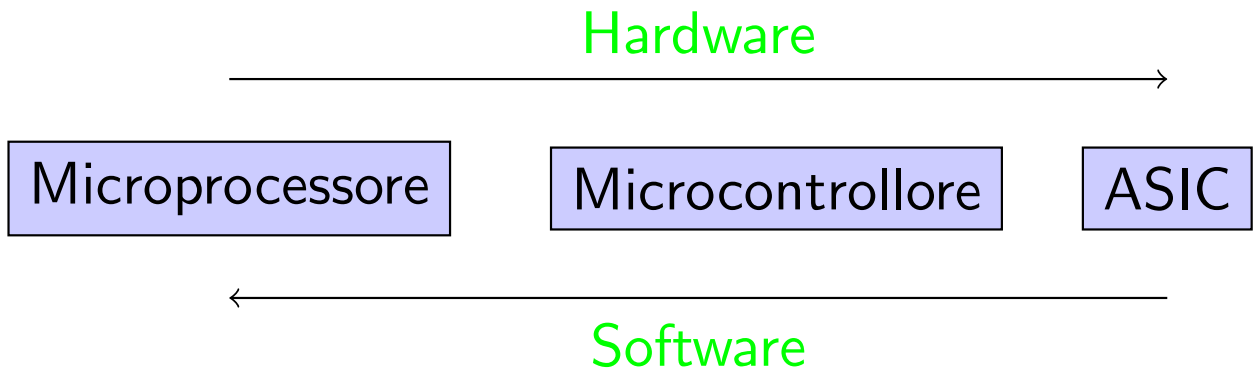
Microprocessore

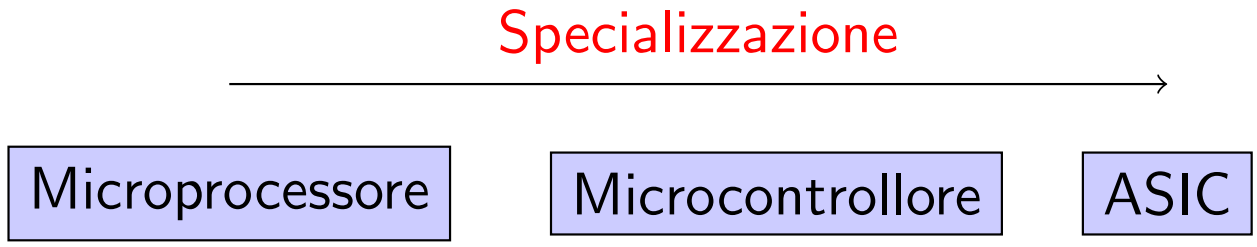
Microcontrollore

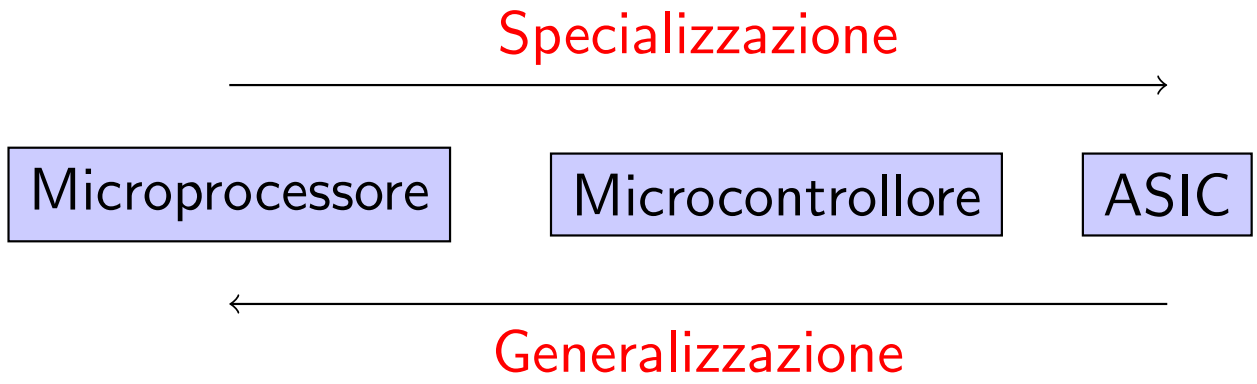
ASIC


Hardware
Microprocessore
Microcontrollore
ASIC

HW/SW


Hardware
Microprocessore
Microcontrollore
ASIC
Software


Specializzazione
Microprocessore
Microcontrollore
ASIC

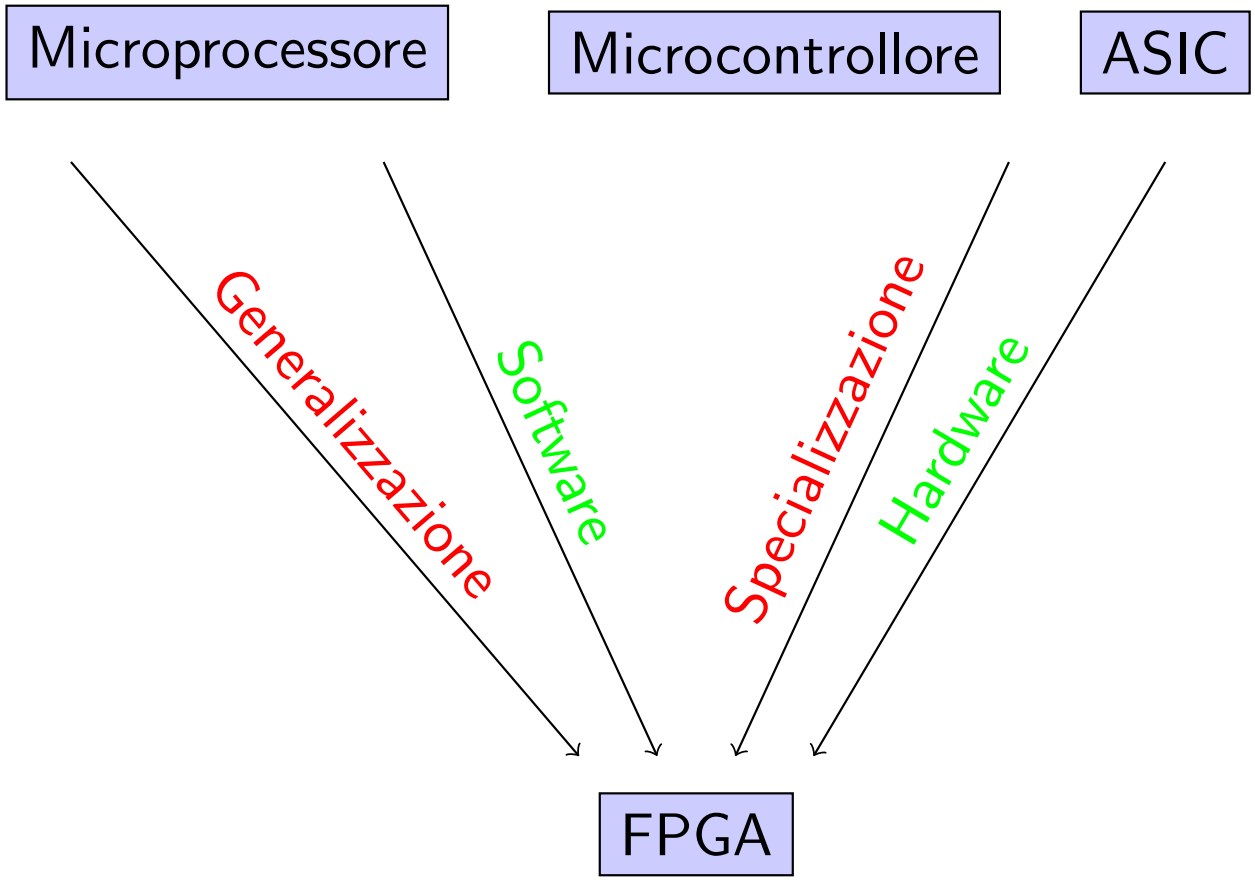

Specializzazione
Microprocessore
Microcontrollore
ASIC
Generalizzazione

Microprocessore

Microcontrollore

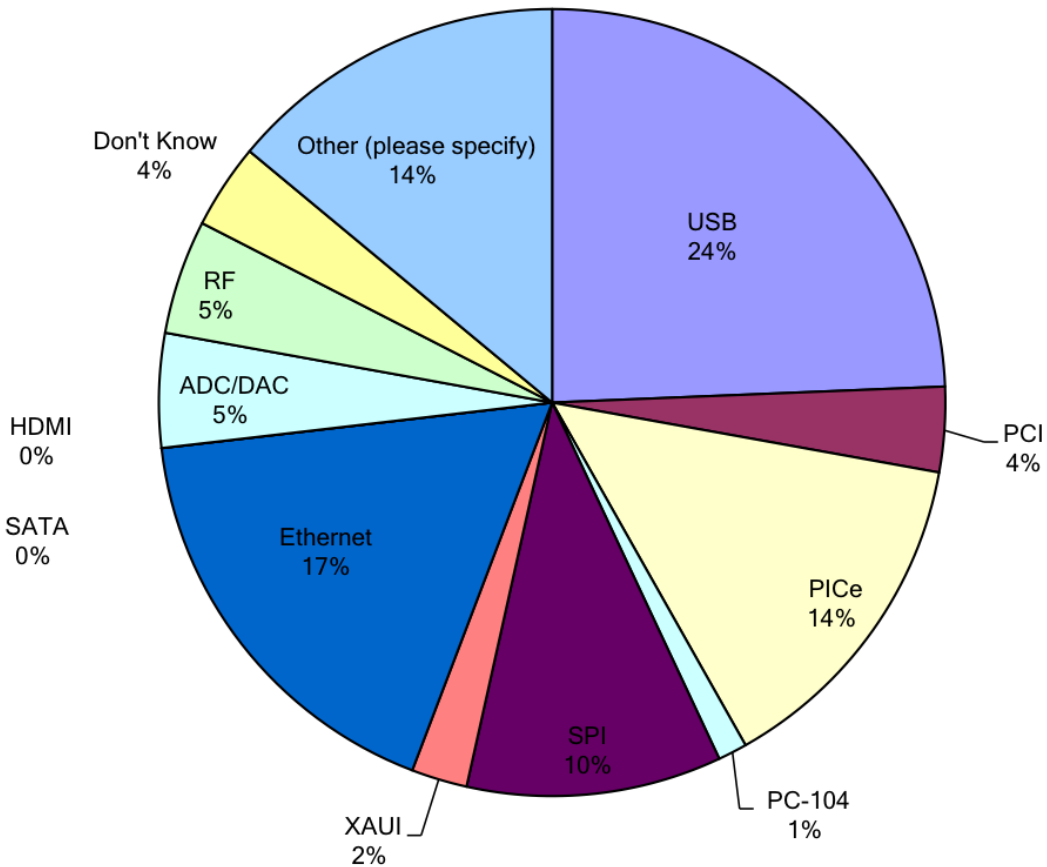
ASIC

FPGA


Microprocessore
Microcontrollore
ASIC
Generalizzazione
Software
Specializzazione
Hardware
FPGA

FPGA per Prototipizzazione

Questa estrema versatilità fa si che il primo utilizzo di FPGA sia stato per prototipizzazione.



FPGA per applicazioni generiche

Successivamente (circa 2008) miglioramenti tecnologici hanno fatto si che:

FPGA per applicazioni generiche

Successivamente (circa 2008) miglioramenti tecnologici hanno fatto si che:

- Le FPGA hanno aumentato il numero di celle.

FPGA per applicazioni generiche

Successivamente (circa 2008) miglioramenti tecnologici hanno fatto si che:

- Le FPGA hanno aumentato il numero di celle.
- Hanno aumentato le frequenze.

FPGA per applicazioni generiche

Successivamente (circa 2008) miglioramenti tecnologici hanno fatto si che:

- Le FPGA hanno aumentato il numero di celle.
- Hanno aumentato le frequenze.
- Si sono abbassati i costi dei dispositivi.

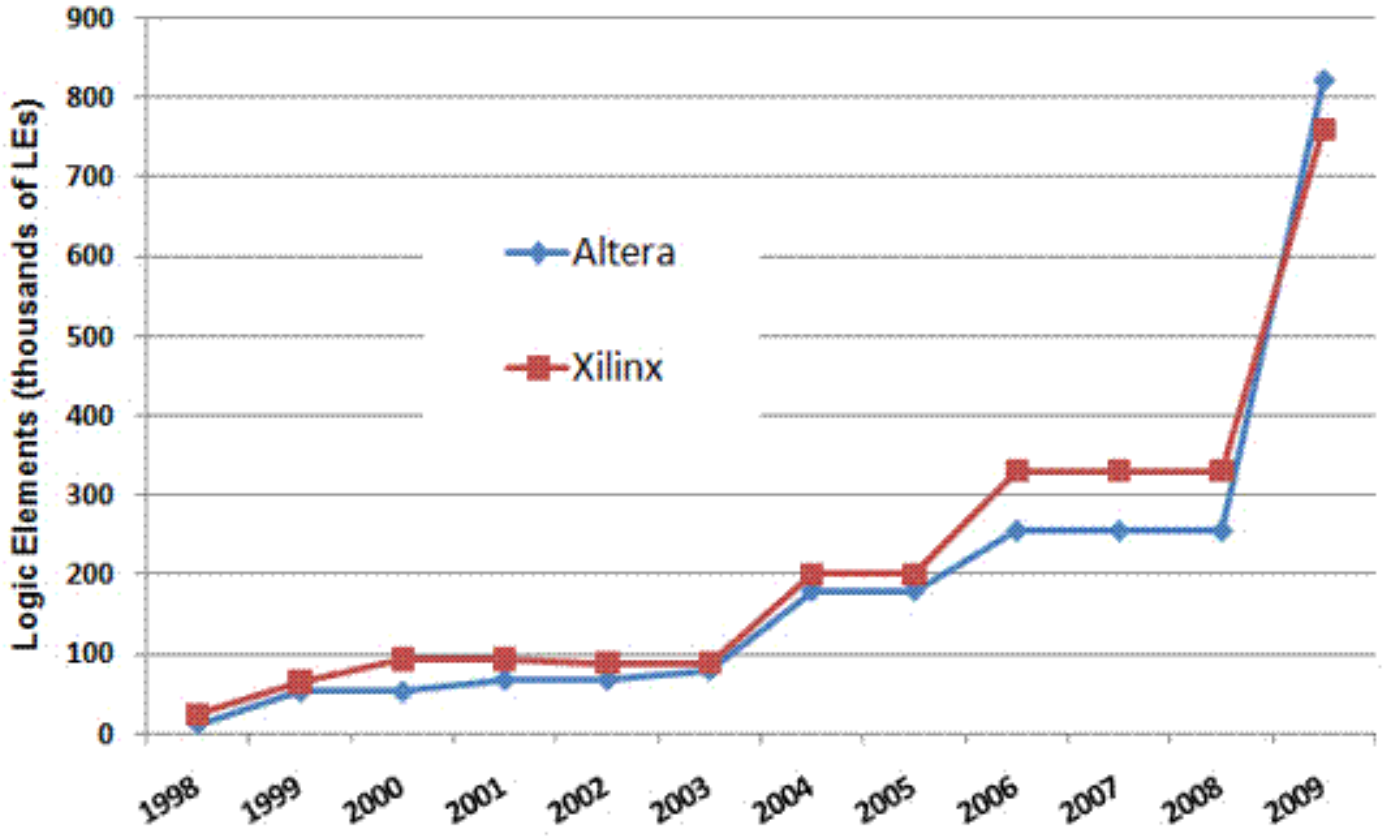
FPGA per applicazioni generiche

Successivamente (circa 2008) miglioramenti tecnologici hanno fatto si che:

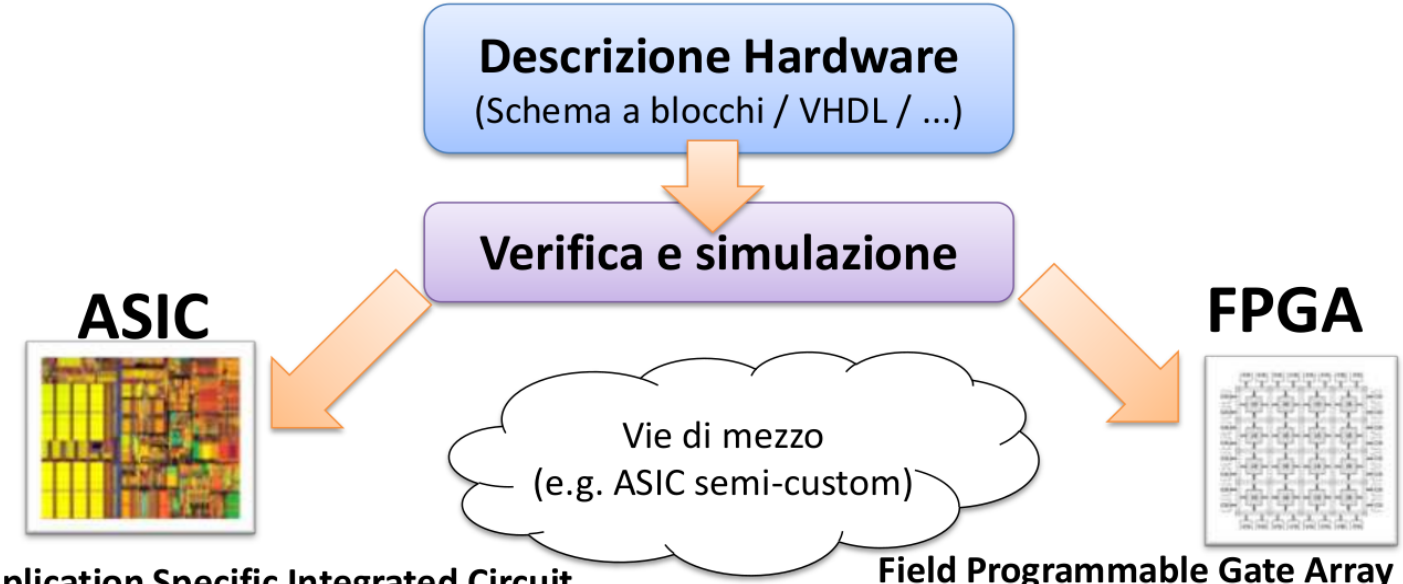
- Le FPGA hanno aumentato il numero di celle.
- Hanno aumentato le frequenze.
- Si sono abbassati i costi dei dispositivi.

Sono diventate competitive anche per essere usate direttamente nei prodotti.

Numero di Celle



ASIC e FPGA



Application Specific Integrated Circuit

- Prodotti tramite processi litografici
- Altissimo costo iniziale (NRE ≈ 500K $)
- Bassissimo costo per unità (≈ qualche $)
- Minori consumi e area
- Alto *time to market* (≈ 6 mesi)

Field Programmable Gate Array

- Hardware riconfigurabile
- Investimento iniziale praticamente nullo
- Costo per unità (50-100$)
- Immediatamente operativi
- Maggiori consumi ed area
- f_{MAX} minore (50-400 Mhz)

UNIVERSITÀ DEGLI STUDI
DI PERUGIA

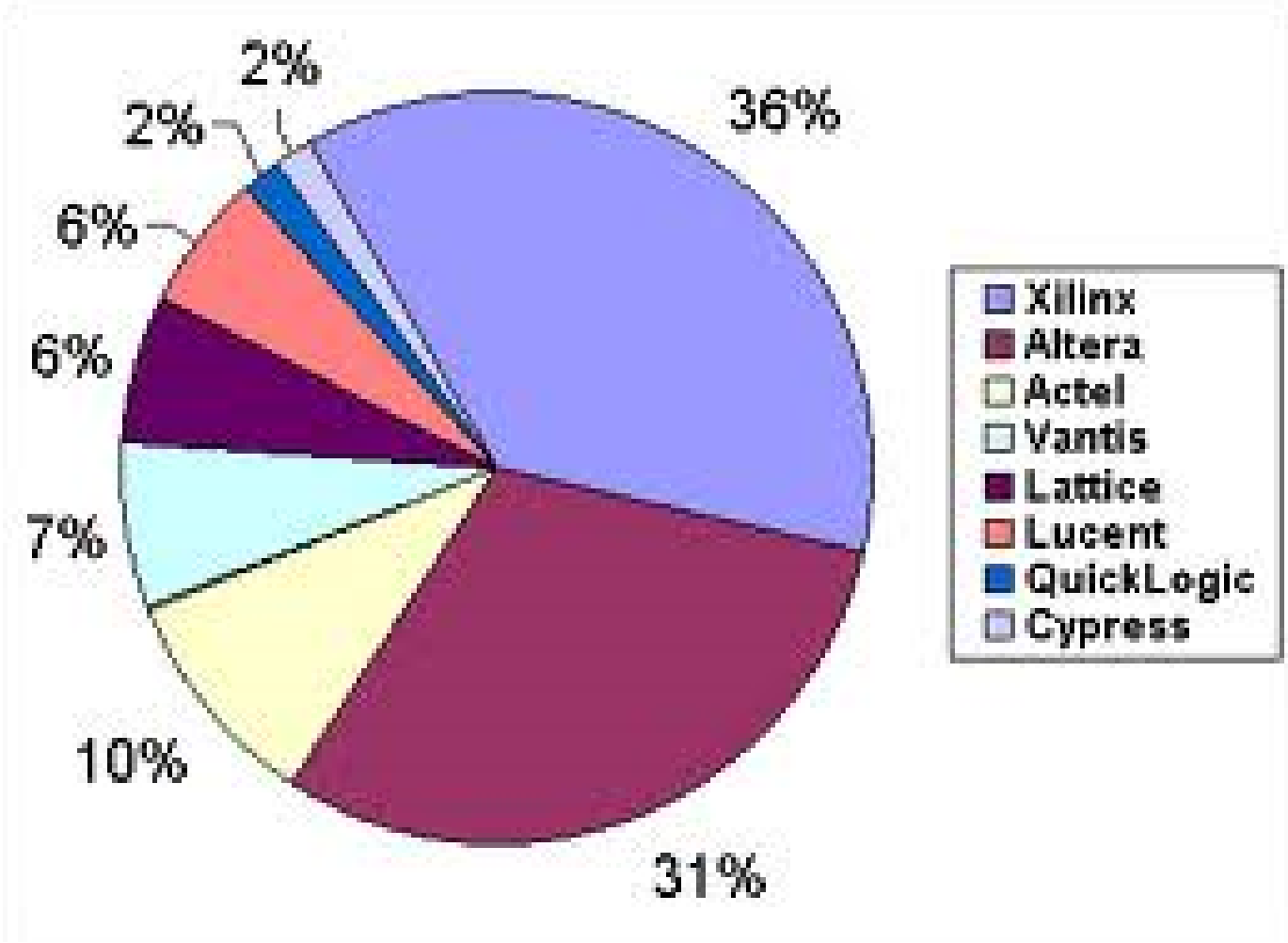
FPGA

Mirko Mariotti

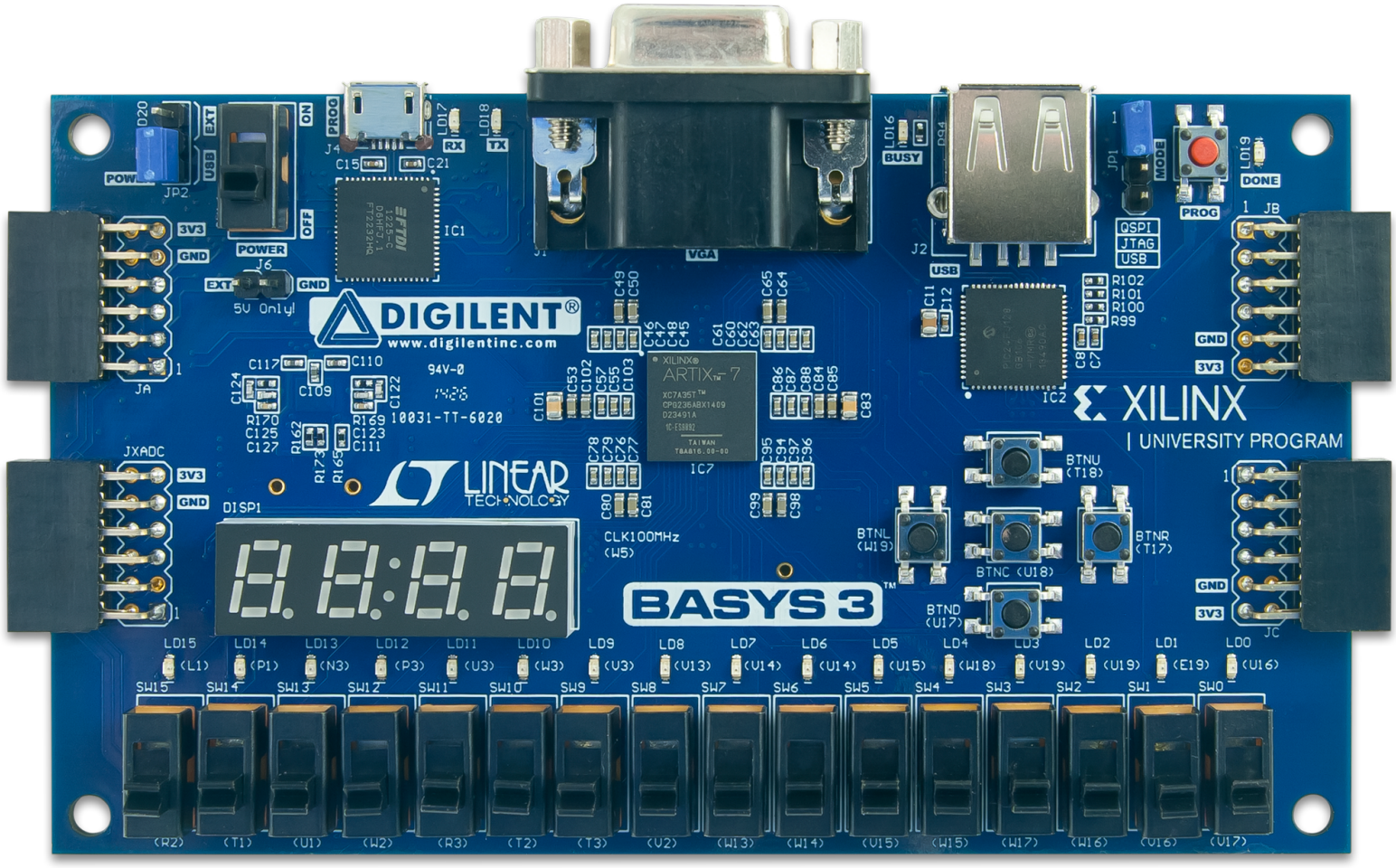
Chipmakers

- Xilinx
- Altera
- Lattice

Mercato

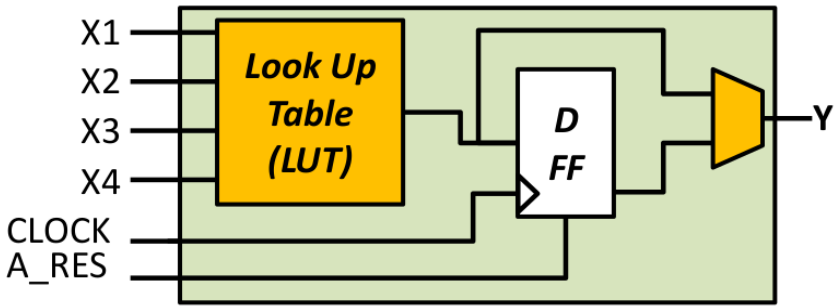

2%
2%
36%
6%
6%
7%
10%
31%
Xilinx
Altera
Actel
Vantis
Lattice
Lucent
QuickLogic
Cypress

Evaluation boards



Architettura di un FPGA

Logic cell



Modello generale di riferimento

I dettagli delle celle logiche in realtà variano in base al produttore ed al modello dell'FPGA.

Look-Up Table

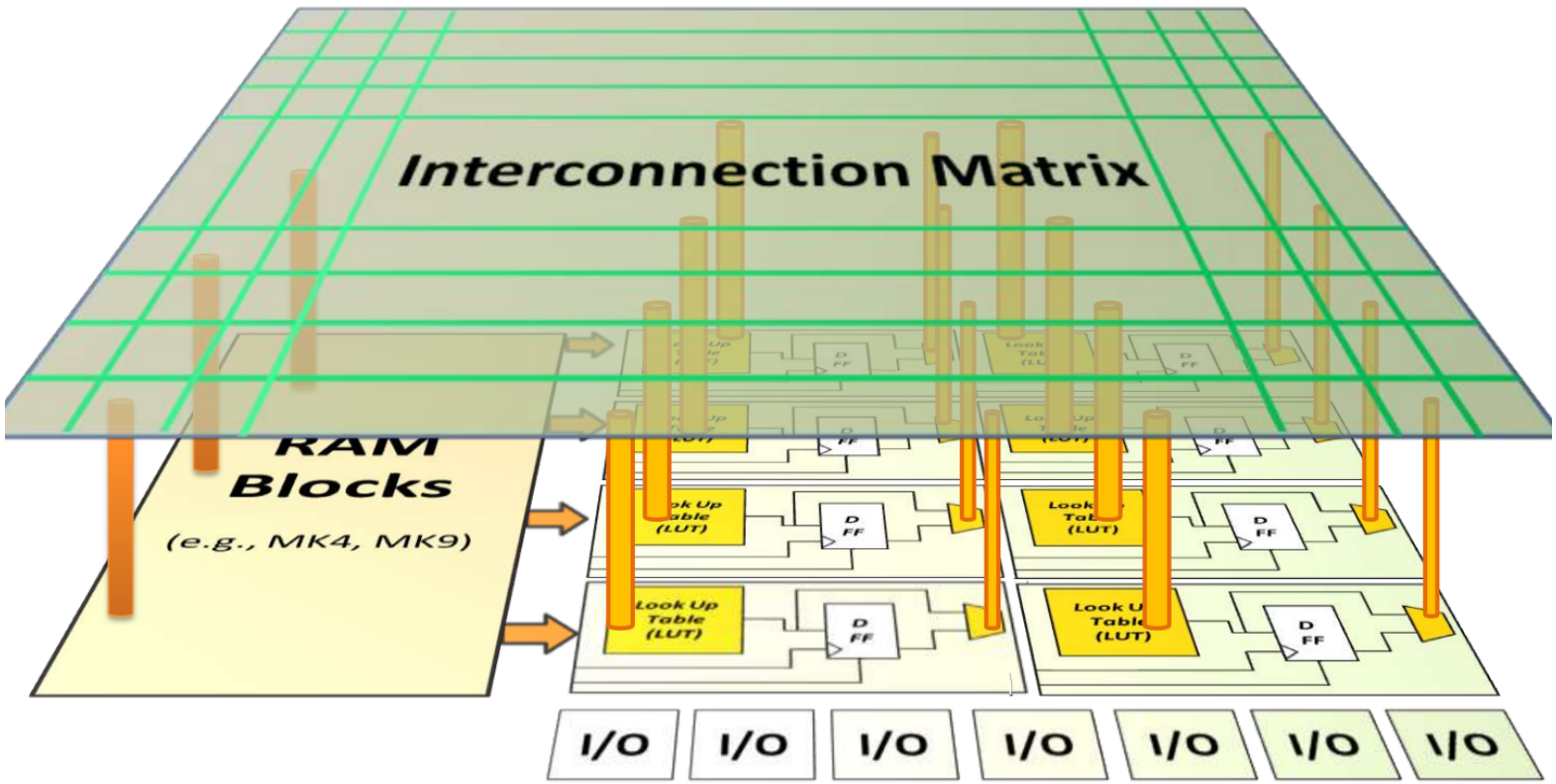
X1	X2	X3	X4	Y
0	0	0	1	?
0	0	1	0	?
...	...	...	...	?
1	1	1	1	?

Elemento programmabile

Tipicamente ogni cella comprende:

- **Una look-up table:** che consente di mappare una qualsiasi funzione combinatoria 4 ingressi 1 uscita
- **Un FF di tipo D** (con set e clr asincroni)
- **Un mux 2 -> 1:** per bypassare il FF in caso di celle puramente combinatorie

Architettura di un FPGA



Software

- Vivado (Xilinx)
- Quartus II (Altera)
- Icecube2 (Lattice)

Altro software

Commerciale:

- Modelsim

Open source:

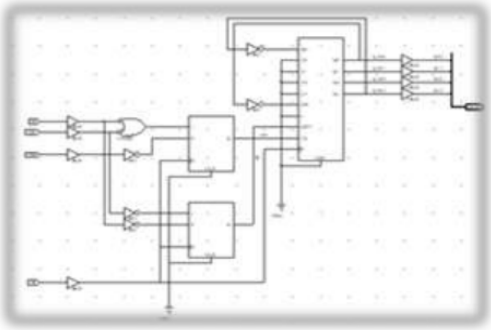
- Iverilog
- GHDL
- Gtkwave

UNIVERSITÀ DEGLI STUDI
DI PERUGIA

Programmazione

Mirko Mariotti

Schemi a blocchi



PRO

- Semplicità e rapidità di sviluppo
- Componenti legacy (e.g., 74xx)

CONTRO

- Formato file e comp. NON standard
- Scarsa manutenibilità
- Prettamente per sistemi semplici

Linguaggi di descriz. HW

VHDL, Verilog, SystemC

```
begin
  if (RESET_N = '0') then

    for col in 0 to BOARD_COLUMNS-1 loop
    ...
   elsif (rising_edge(CLOCK)) then
```

PRO

- Linguaggi standard (!)
- Flessibilità e manutenibilità
- Gestibilità di design complessi

CONTRO

- Modello computazionale

Linguaggi

La programmazione di FPGA avviene tramite l'uso di linguaggi HDL (Hardware Description Language)

I due più usati sono:

- Verilog
- VHDL

- Il modello computazionale dei linguaggi di descrizione hardware (VHDL, Verilog, …) è profondamente diverso dai linguaggi di programmazione tradizionali.
- Nei linguaggi di programmazione (C, C#, Java …) gli statement del linguaggio definiscono istruzioni, che vengono eseguite sequenzialmente da una infrastruttura (dalla CPU nel caso di C, per mezzo di una virtual machine nel caso di Java …)
- Nei linguaggi di descrizione dell'hardware gli statement del linguaggio, invece, definiscono blocchi di hardware.
- Non c'è nessuna esecuzione sequenziale, nessuna infrastruttura sottostante, nessun run-time.

Sintesi

Il processo di sintesi trasforma una descrizione HDL in una *netlist* di gate elementari.

La sintesi è applicabile ad un sub-set del linguaggio (VHDL sintetizzabile)

La descrizione avviene attenendosi a *template* che vengono analizzati e riconosciuti dai sintetizzatori e danno luogo ai componenti logici corrispondenti.

Il risultato della sintesi dipende dal sintetizzatore e dalle librerie di mapping adoperate (forniti dal produttore in caso di FPGA).

Sintesi

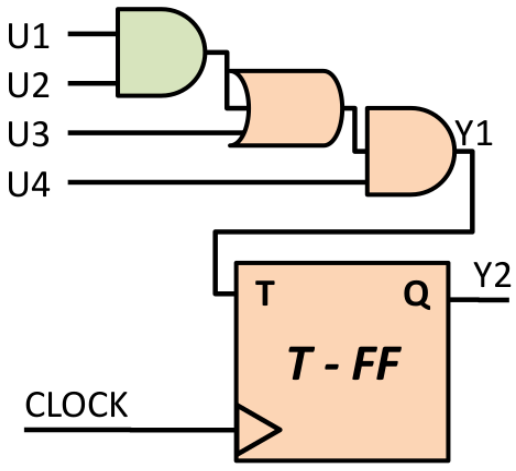
- Nella fase di Analisi e Sintesi il sintetizzatore analizza i costrutti del linguaggio (VHDL & c.), riconosce i template utilizzati e deriva i componenti di alto livello (contatori, multiplexer, decoder …).
  Sostanzialmente: trasforma il testo in uno schema a blocchi

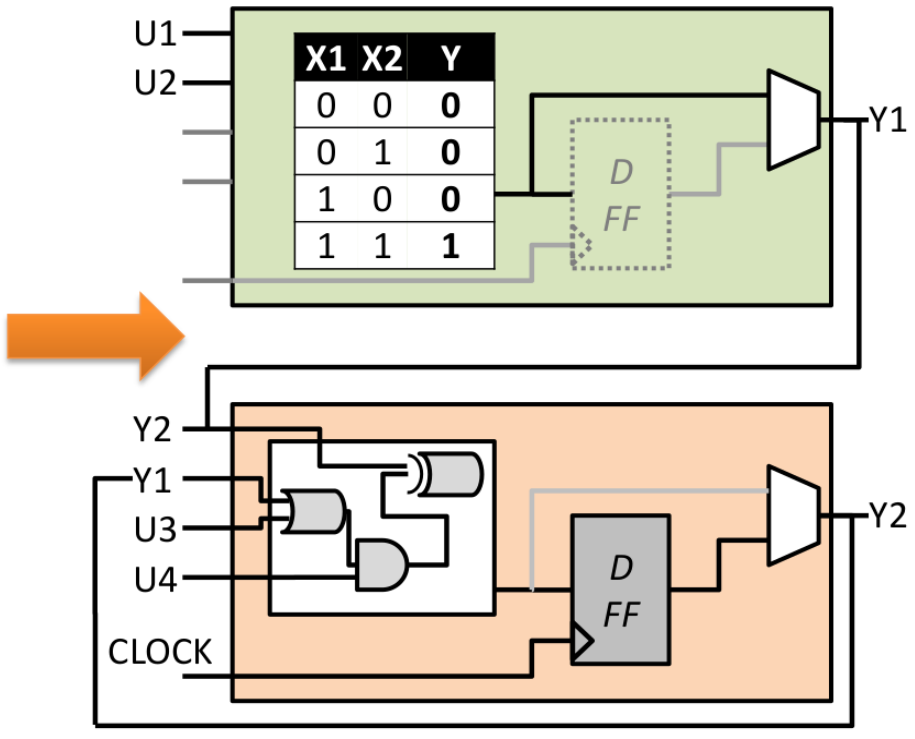
- Tale rappresentazione, però, non ha una corrispondenza diretta con l'hardware finale, ma solo "funzionale".

- L'hardware viene inferito solo nella fase di Mapping (o Fitting, Place and Route), in cui il sintetizzatore si avvale delle celle logiche dell'FPGA per mappare le funzionalità descritte su hardware reale.

- Il ruolo del progettista è di descrivere *cosa* va fatto, non *come.*
  Es: ~~*out <= not(not(not(not(a))));*~~ *(NON da luogo ad catena di invertitori)*

Implementazione





I componenti di alto livello inferiti durante la A&S vengono mappati sull' hardware sfruttando le risorse dispnibili (celle logiche programmabili nel caso di FPGA).

Implementazione

- Identificazione degli elementi di memoria: tutti i gli elementi di memoria di alto livello (registri/contatori/shift-register) vengono ricondotti ad elementari Flip-Flop.
- Identificazione delle funzioni di trasferimento (equazioni booleane) . Ovvero identificazione dei percorsi combinatori: (i) tra registri; (ii) tra input e registri; (iii) tra registri ed output; (iv) tra input ed output.
- Riduzione delle equazioni (ottimizzazione).
- Mapping delle equazioni corrispondenti sulle Look-Up Table delle celle logiche.
- Tutto questo vale non solo per VHDL ma anche per gli schemi a blocchi. Quello che si “disegna”, infatti, non riflette l’hardware finale, ma solo la funzionalità che si desidera modellare.

UNIVERSITÀ DEGLI STUDI DI PERUGIA

Una parte importante dello sviluppo su FPGA è data dalla necessità di simulare e verificare un' implementazione.

Testbench

UNIVERSITÀ DEGLI STUDI
DI PERUGIA

Verilog

Mirko Mariotti

Moduli

- I moduli verilog sono parti di codice (circuito) riutilizzabili dentro un programma. (simili alle funzioni nei linguaggi di programmazione)

Moduli

- I moduli verilog sono parti di codice (circuito) riutilizzabili dentro un programma. (simili alle funzioni nei linguaggi di programmazione)
- Sono definiti dalla keyword “module” e finiscono con “endmodule”.

Moduli

- I moduli verilog sono parti di codice (circuito) riutilizzabili dentro un programma. (simili alle funzioni nei linguaggi di programmazione)
- Sono definiti dalla keyword “module” e finiscono con “endmodule”.
- Hanno parametri di input ed output

```
module MyModule ([parameters]);
     inputs ...
     outputs ...
     internal variables ...
     ...
     Module Code ...
endmodule
```

Modulo principale

- Un modulo è il principale. (simile al main in un linguaggio di programmazione)

Modulo principale

- Un modulo è il principale. (simile al main in un linguaggio di programmazione)
- È definito come punto di ingresso del programma.

Modulo principale

- Un modulo è il principale. (simile al main in un linguaggio di programmazione)
- È definito come punto di ingresso del programma.
- Solitamente ha come input ed output ingressi ed uscite fisiche del FPGA.

Operatori

Operator Type	Operator Symbol	Operation Performed
Arithmetic	*	Multiply
	/	Division
	+	Add
	-	Subtract
	%	Modulus
	+	Unary plus
	-	Unary minus
Logical	!	Logical negation
	&&	Logical and
	||	Logical or
Relational	>	Greater than
	<	Less than
	>=	Greater than or equal
	<=	Less than or equal
Equality	==	Equality
	!=	inequality
Reduction	~	Bitwise negation
	~&	nand
	|	or
	~|	nor
	^	xor
	^~	xnor
	~^	xnor
Shift	>>	Right shift
	<<	Left shift
Concatenation	{ }	Concatenation
Conditional	?	conditional

Wire:
Sono utilizzati per connettere elementi diversi. Si possono immaginare come fili fisici. Possono essere letti o assegnati ma non contengono nessun valore. Hanno quindi bisogno di essere pilotati continuamente da un'assegnazione o dalla porta di un modulo.

Reg:
Rappresentano l'elemento che contiene un valore in Verilog. Mantengono il loro valore fino all'assegnazione successiva. Possono essere sintetizati con un FF o con un circuito combinatorio.

Numeri

Il formato generico di un intero in Verilog è:
<size>‘<base><value>

Esempi:
12 // decimale (base 10)
8 ‘h5F // esadecimale a 8 bit
6 ‘b11_0010 // binario a 6 bit
‘o576 // ottale senza dimensione
32 ‘bz // binario a 32 bit Hi-Z

Il carattere _ viene ignorato e può essere usato come separatore.

Scalari e Vettori

Le variabili possono essere scalari e vettoriali.

Esempi:

```
reg out; // scalare
reg [7:0] databus; // 8 bit bus
wire [1:0] select; // 2 bit bus
wire enabled; // scalare
```

Blocchi

I blocchi in Verilog si definiscono con begin - end.

Si possono definire due tipo di blocchi:

Blocco Initial
Esegue all'inizio della simulazione.

Blocco Always
Esegue sempre ed è associto ad una lista che specifica quando eseguire il blocco di codice

```
always  @ (a or b or sel)
begin
   ...
end
```

Assegnamento blocking

A = 3;

L' espressione viene valutata nel flusso di esecuzione e la variablile viene assegnata immediatamente.

Assegnamento NON blocking

A <= A + 1;

L' espressione viene valutata nel flusso di esecuzione, viene assegnato il risultato ad una variabile temporanea e la variablile viene assegnata prima di passare all istante di simulazione successivo.

Assegnamento continuo:

assign A = in1 & in2; // A è di tipo wire

usato per modellare semplice logica combinatoria e per cambiare nome ai segnali.

Strutture di controllo

If-else e case

```
if( A == B )
begin
   C = 0;
end
else
begin
   C = 1;
end
```

```
case (sel)
   2'b00: out = 0;
   2'b01: out = 1;
   default: out = x;
endcase
```