UNIVERSITÀ DI PISA

Dipartimento di Informatica
Corso di Laurea in Informatica

Ambienti per la programmazione di dispositivi FPGA

Candidato:
Alessandra Fais

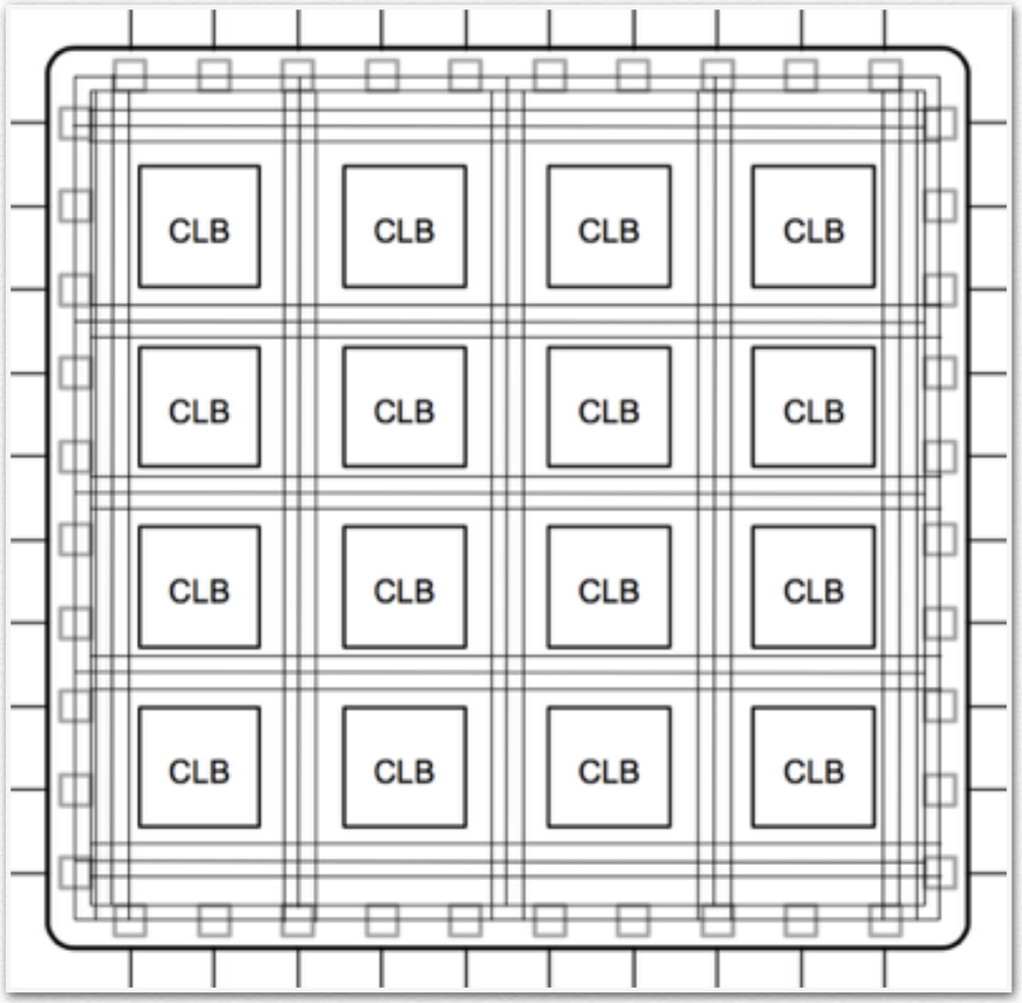
Relatore:
Prof. Marco Danelutto

Cos'è un FPGA

- Dispositivi con componenti logici programmabili
- Circuiti integrati tradizionali vs FPGA
  - immutabilità vs flessibilità
  - costo

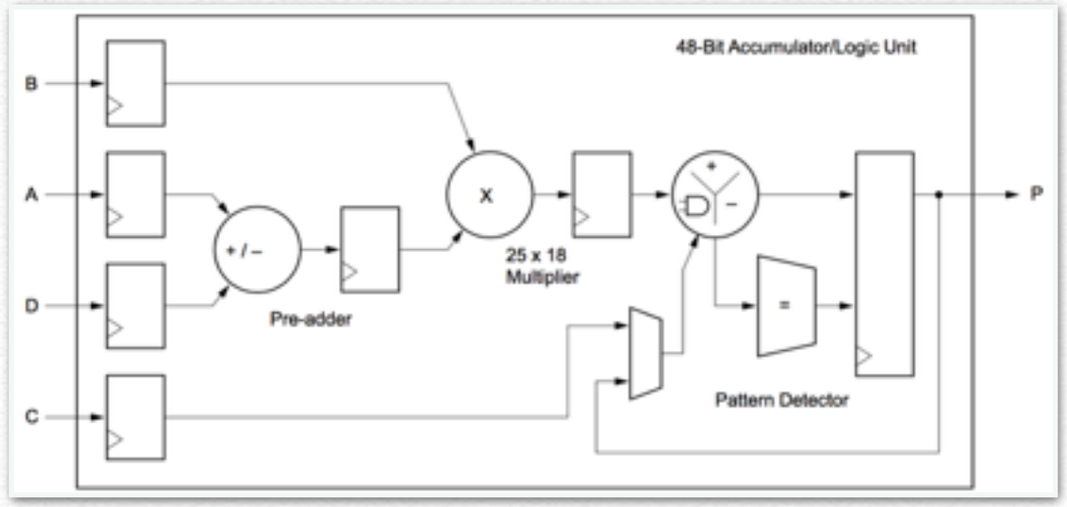
Architettura di un FPGA

- CLB
- LUT
- registro
- struttura di interconnessione regolare e configurabile



Componenti

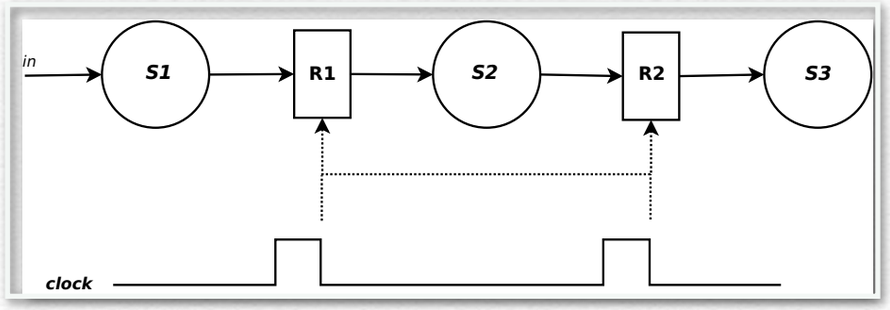
- Base:
  - Tabelle di look-up
  - Registri flip-flop
    - Connessioni
    - Porte per l'I/O



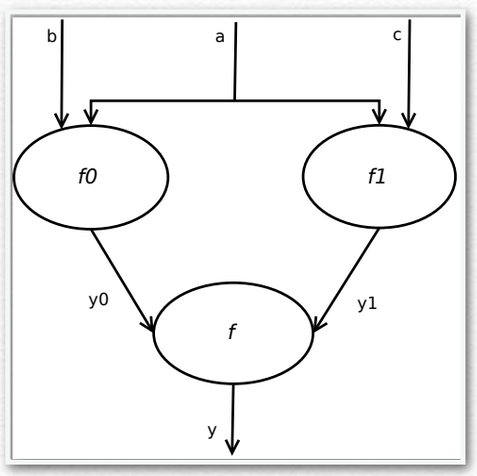
- Altri:
  - Componenti specializzati per computazioni particolari
  - Moduli Block-RAM

Meccanismi per la performance

- Pipelining



- Dataflow



- Scheduling

Metriche per la misurazione della performance

- Aumento del numero degli stadi nel pipeline
  - crescita della latenza
  - incremento della banda
- Uso di memorie distribuite
  - banda massima nell'accesso ai dati durante l'esecuzione
- Circuito implementa tutti i possibili rami di esecuzione di un programma

Come programmare un FPGA

- RTL - Register Transfer Level
- Approcci evoluti
  - Chisel
  - OpenCL

Verilog

- HDL a basso livello
- Costrutti:
  - adatti alla generazione di circuiti
  - manipolazione diretta di bit, fili, registri e altri componenti base
- Approccio ostico
  - conoscenza approfondita dell'hardware

Chisel

- HDL ad alto livello
  - domain specific language basato su Scala
- Tipi di dato:
  - interi con segno e non, collezioni di bit, vettori e bundle
- Livello di astrazione maggiore
  - gerarchia di moduli e interfacce

Esempio modulo Addizionatore

VERILOG

```
module addizionatore_binario(
  somma, rip_out, x, y, rip_in);

  output somma;
  output rip_out;
  input x;
  input y;
  input rip_in;

  wire s;
  wire r1;
  wire r2;
  wire r3;

  // genera la somma
  xor(s, x, y);
  xor(somma, rip_in, s);

  // genera il riporto
  and(r1, x, y);
  and(r2, y, rip_in);
  and(r3, x, rip_in);
  or(rip_out, r1, r2, r3);

endmodule
```

ingressi e uscite

logica

CHISEL

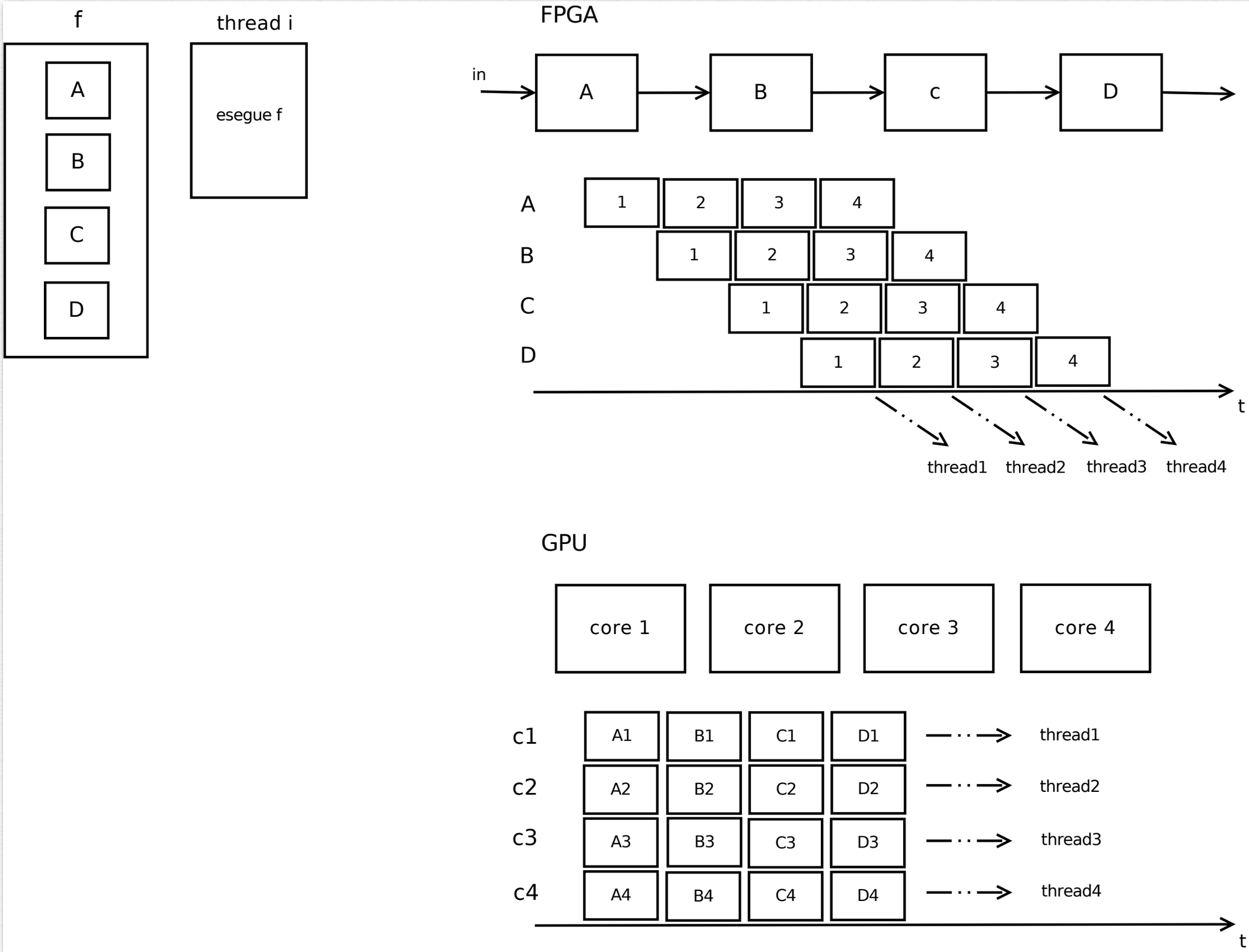
```
class AddizionatoreBinario
        extends Module {

val io = new Bundle {
  val x = UInt(INPUT, 1)
  val y = UInt(INPUT, 1)
  val rip_in = UInt(INPUT, 1)
  val somma = UInt(OUTPUT, 1)
  val rip_out = UInt(OUTPUT, 1)
}

// genera la somma
val x_xor_y = io.x ^ io.y
io.somma := x_xor_y ^ io.rip_in

// genera il riporto
val x_and_y = io.x & io.y
val y_and_rip_in = io.y & io.rip_in
val x_and_rip_in = io.x & io.rip_in
io.rip_out := x_and_y |
              y_and_rip_in |
              x_and_rip_in

}
```

OpenCL su FPGA

- Sistema host + coprocessore FPGA
- Compilatore implementa funzione (kernel) da eseguire sul device
  - pipeline + replicazione funzionale
  - stream sequenziale di dati in ingresso

Differenza con GPU

- Sistema host + coprocessore GPU
- Compilatore implementa funzione (kernel) da eseguire sul device
  - esecuzione parallela sui core
  - accesso vettoriale ai dati in ingresso


f
A
B
C
D
thread i
esegue f
FPGA
in
A
B
c
D
A
B
C
D
1
2
3
4
t
thread1
thread2
thread3
thread4
GPU
core 1
core 2
core 3
core 4
c1
c2
c3
c4
A1
B1
C1
D1
A2
B2
C2
D2
A3
B3
C3
D3
A4
B4
C4
D4
thread1
thread2
thread3
thread4
t

Conclusioni

- Struttura e funzionamento di FPGA
- Modi per programmare tali dispositivi
  - raffronto tra approccio tradizionale e nuove metodologie
    - Verilog, Chisel, OpenCL
  - evoluzione verso programmabilità e riduzione dei costi di sviluppo

Grazie!