



---

# Dispositivi Programmabili

FPGA (*Field Programmable Gate Arrays*)

Introduzione

Famiglie di FPGA: architetture e tipi di blocchi

I blocchi logici di base

Flusso di progetto e technology mapping

---



## FPGA: Introduzione

---

- Le FPGA (*Field Programmable Gate Arrays*) sono dispositivi *programmabili direttamente dall'utente* costituiti da un array di *componenti logici* collegabili tra loro.
- Le FPGA mettono a disposizione dell'utente:
  - **Componenti logici** (costituiti da porte logiche, Flip-flop, Buffer...) offrono un insieme di funzionalità che possono essere sfruttate anche solo in parte.
    - **Componenti logici molto complessi** consentono di localizzare funzionalità molto articolate favorendone la velocità ed una ridotta connettività a discapito di un minor sfruttamento.
    - **Componenti logici poco complessi** consentono un maggior grado di sfruttamento (flessibilità) a discapito della elevata connettività.
  - **Linee di connessione locale e distribuite** (lunghe e corte):
    - Linee che attraversano una parte ridotta del dispositivo e che sono condivise da pochi elementi logici: ritardi e potenza utilizzata contenuti.
    - All'interno di uno stesso dispositivo possono coesistere differenti linee locali di lunghezza differente: elevata flessibilità



## FPGA: Introduzione

---

- Alcuni aspetti presenti e futuri che rendono le FPGA particolarmente interessanti sono:
  1. Rappresentano il miglior compromesso nella famiglia dei dispositivi programmabili tra **flessibilità e prestazione**.
    - $PLD \Rightarrow FPGA \Leftarrow MPGA$
  2. Spesso, rappresentano un ottimo compromesso tra **costo e prestazione**.
    - FPGA vs. ASIC
  3. Rappresentano, ad oggi, strumenti per una rapida verifica delle caratteristiche dei dispositivi puramente hardware
    - *fast prototyping*
  4. Rappresentano, dal punto di vista evolutivo, un ottimo compromesso tra **specializzazione e generalità** proponendosi come piattaforme per la realizzazione di sistemi completi sia come **definitivi sia come prototipi** (eliminando o riducendo anche i problemi legati alla co-simulazione dei sistemi *hw-sw*).
    - Dominio Software (GPP, DPS)  $\Rightarrow$  FPGA  $\Leftarrow$  Dominio Hardware (ASIC)



## FPGA: Introduzione

---

1. Le FPGA sono un compromesso tra le MPGA (più grandi, veloci e che consentono di realizzare funzionalità a granularità fine ma che richiedono di attraversare un parte del processo di fabbricazione) e i PLD (più piccoli e meno costosi ma che offrono meno funzionalità).
  - **MPGA** (*Mask Programmable Gate Array*): array di elementi che possono essere connessi tra loro secondo le specifiche dell'utente.
    - La tipologia più diffusa è quella dei *Sea of Gates*, costituita da una matrice di transistor *pre-realizzati* che possono essere utilizzati per implementare una funzionalità.
    - Le connessioni sono completate nella fase finale del processo di fabbricazione e consiste nel realizzare le metallizzazioni necessarie per creare le funzionalità richieste (sono *dispositivi SEMI-CUSTOM*).
  - **PLD** (*Programmable logic device*): ROM (es. E<sup>2</sup>PROM), PAL, PLA...



## FPGA: Introduzione

---

2. Le FPGA, sebbene abbiano prestazioni ridotte rispetto ad un ASIC (*Application Specific Integrated Circuit*), hanno:
  - Un costo per unità più basso per bassi volumi di produzione.
  - Un costo e tempo di avvio alla produzione inferiori.
  - Consentono di riversare sullo stesso dispositivo, in momenti diversi, configurazioni che descrivono applicazioni differenti o parti di esse.
    - Piattaforma Hardware Multi-modale (*off-line*).
      - Le descrizioni sono conservate in ROM e riversate in RAM prima dell'uso.
    - Piattaforma Riconfigurabile (*on-line*)
      - Le descrizioni di parte della applicazione sono riversate in RAM, sostituendole ad altre non più in uso, durante il periodo di funzionamento.



## FPGA: Introduzione

---

- Un flusso di sviluppo tradizionale prevede che la simulazione del comportamento di un circuito venga svolta mediante software.
- La complessità crescente dei dispositivi mette in evidenza due svantaggi di questo approccio:
  - Verifica di porzioni ridotte del dispositivo introducendo problemi di verifica del funzionamento complessivo.
  - Riduce la possibilità di rispettare le scadenze imposte dal time-to-market.
- 3. Le FPGA consentono di realizzare dei prototipi in modo rapido (*fast prototyping*) e di emulare i dispositivi realizzati come insieme di *logica generica*.



## FPGA: Introduzione

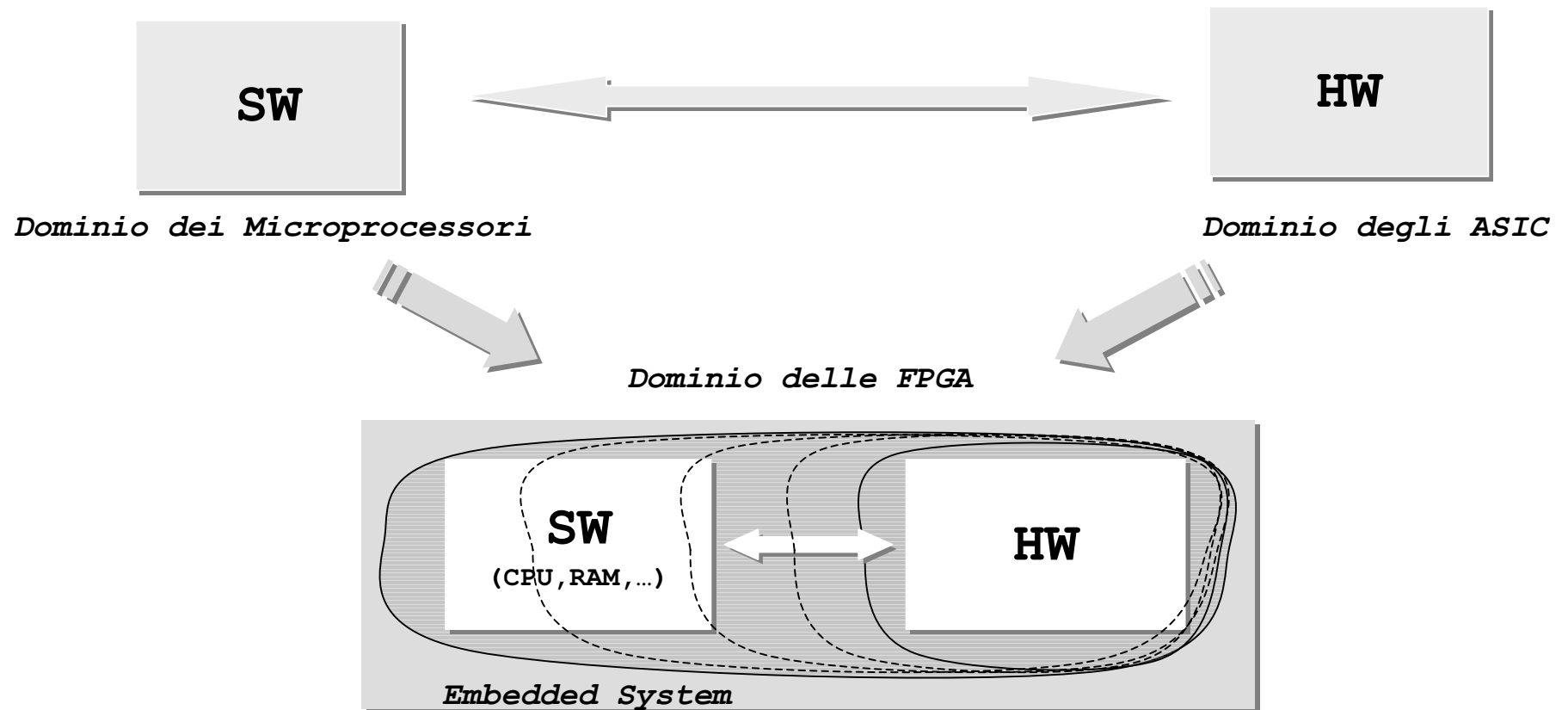
---

4. L'evoluzione delle FPGA le propone al mercato come dispositivi capaci di **supportare** lo sviluppo di **interi sistemi dedicati** (*Embedded Systems*)
  - Abbandonano il ruolo storico di dispositivi che implementano le sole sezioni puramente hardware per arricchirsi di elementi che consentono la realizzazione delle sezioni software.
    - Blocchi di RAM
    - IP per la realizzazione di CPU
    - CPU tipo GPP (General Purpose Processor)
    - Elementi specifici per sezioni DSP
    - ...
  - Le caratteristiche che rendono le FPGA inadatte per applicazioni puramente *General Purpose*, dominio dei sistemi basati su microprocessori, o puramente *Special Purpose*, dominio degli ASIC, vanno con il tempo riducendosi.
    - Ottimizzazioni e specializzazione di alcune aree riducono le distanze tra i due domini e le FPGA.



# FPGA: Introduzione

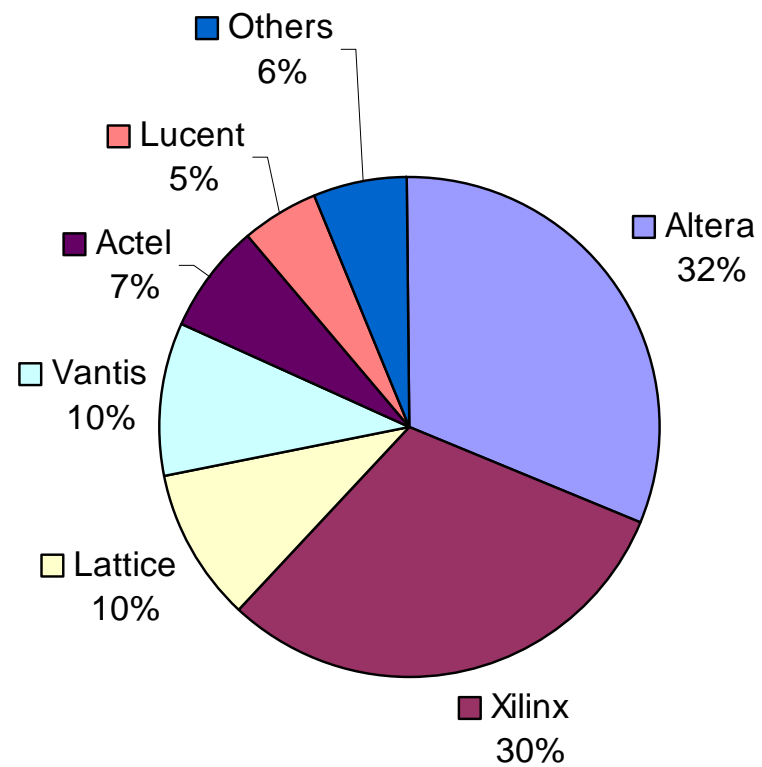
4. (continua):





# FPGA: Introduzione

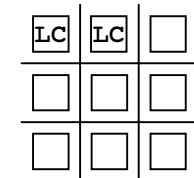
## □ Mercato delle FPGA





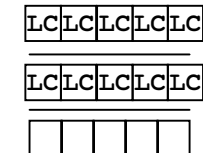
# Field Programmable Gate Array

- Le famiglie di FGPA di contraddistinguono per 3 aspetti:
  - Architettura generale
    - Array simmetrici (*Xilinx, QuickLogic*)
    - Organizzati per riga (*Actel*)
    - PLD gerarchici (*Altera*)
  - Tipi di blocchi logici
    - Basati su Look-up Table (*Xilinx*)
    - Basati su Multiplexer (*Actel*)
    - Blocchi PLD (*Altera*)
  - Tecnologia di programmazione
    - RAM statiche (*Xilinx*)
    - EPROM e EEPROM (*Altera*)
    - Anti-Fuse (*Actel, QuickLogic*)

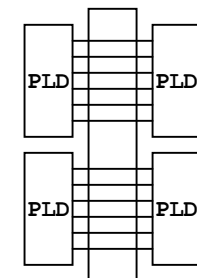


**Array Simmetrici**

LC: Logic Cell (Cella Logica)



**Organizzati per Riga**



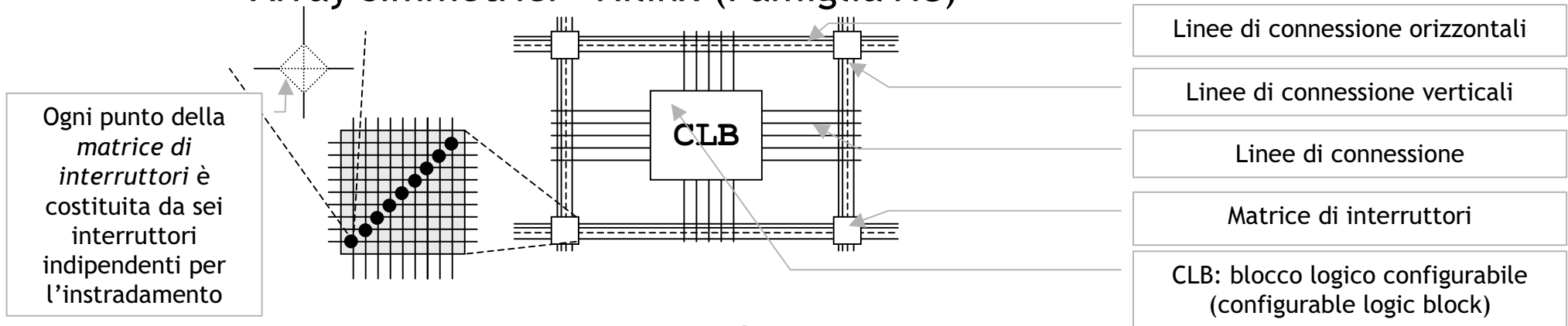
**PLD Gerarchici**



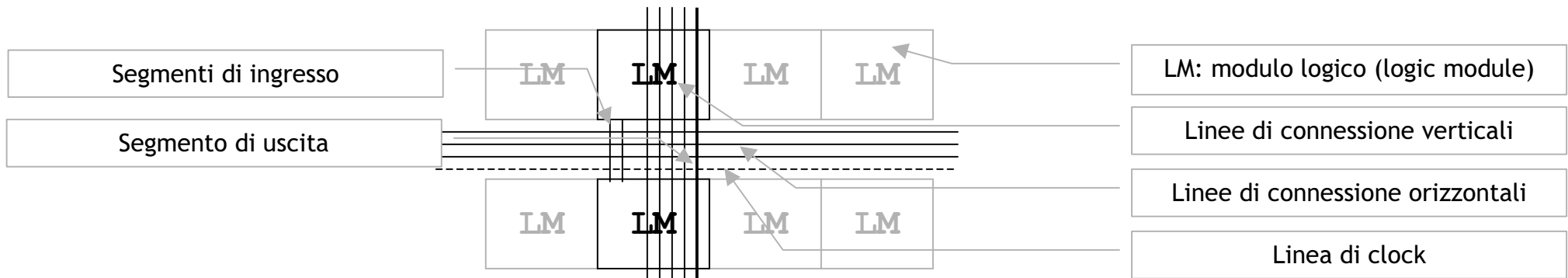
# Field Programmable Gate Array

## Architettura:

### - Array simmetrici - Xilinx (Famiglia XC)



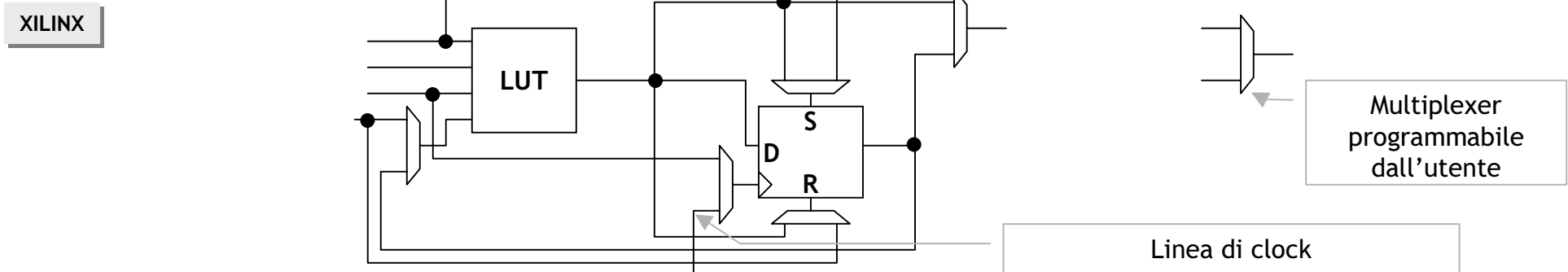
### - Organizzati per riga - Actel (ACT\_1 e 2)



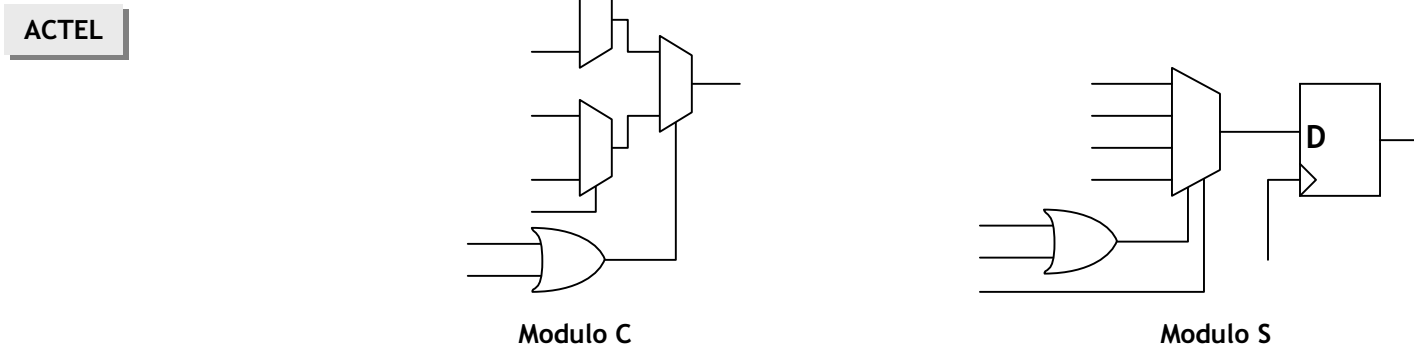


# Field Programmable Gate Array

- Schema semplificato di un CLB (blocco logico programmabile)



- Schema semplificato di un LM (modulo logico)
  - 2 tipi di moduli: tipo C (combinatorio) e tipo S (sequenziale)



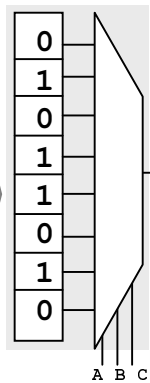


# Field Programmable Gate Array

## □ Blocchi logici base caratteristici delle due architetture

### - Xilinx (CLB)

A	B	C	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0



### • LUT e FF.

- LUT (look-up table): implementa una generica rete combinatoria a  $k$  ingressi.

» Combinazione di  $2^k$  registri ed un multiplexer a  $k$  ingressi di selezione.

» I valori caricati nei  $2^k$  registri sono i valori assunti dalla funzione combinatoria. Il caricamento avviene in fase di configurazione della FPGA.

» I  $k$  ingressi di selezione sono controllati dalle variabili di ingresso della funzione da realizzare.

### - Altera (LM)

• Multiplexer e FF.

- Nota: In relazione alla famiglia di dispositivi, **sono disponibili anche altri elementi** come: multiplexer non programmabili, catene per la propagazione dei riporti, blocchi di RAM, ...

In questa sezione si focalizza l'analisi solo sugli aspetti caratteristici di base e comuni a tutti i prodotti della stessa casa produttrice.

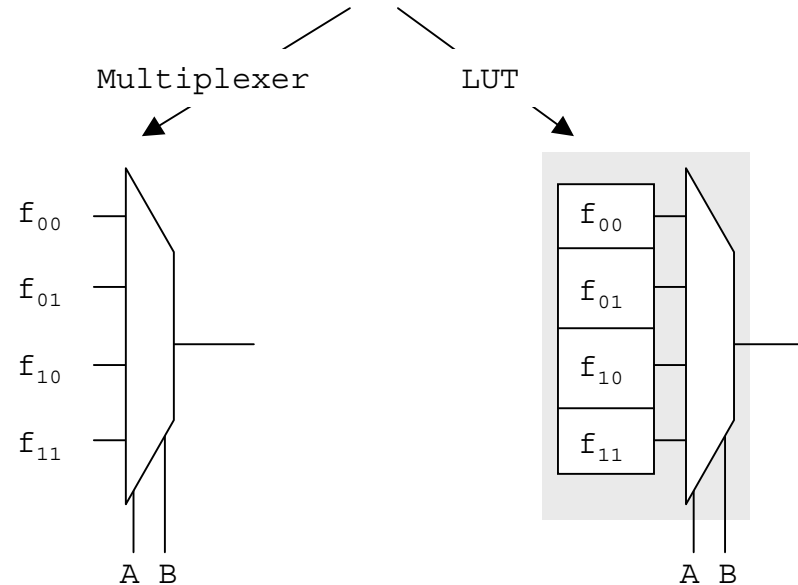


# Field Programmable Gate Array

## □ Ossevazione:

- Un multiplexer a  $2^k$  ingressi ed una LUT a  $k$  ingressi realizzano le stesse funzionalità.
- Nel seguito, per questa ragione, non si fa riferimento ad alcuna tecnologia specifica.

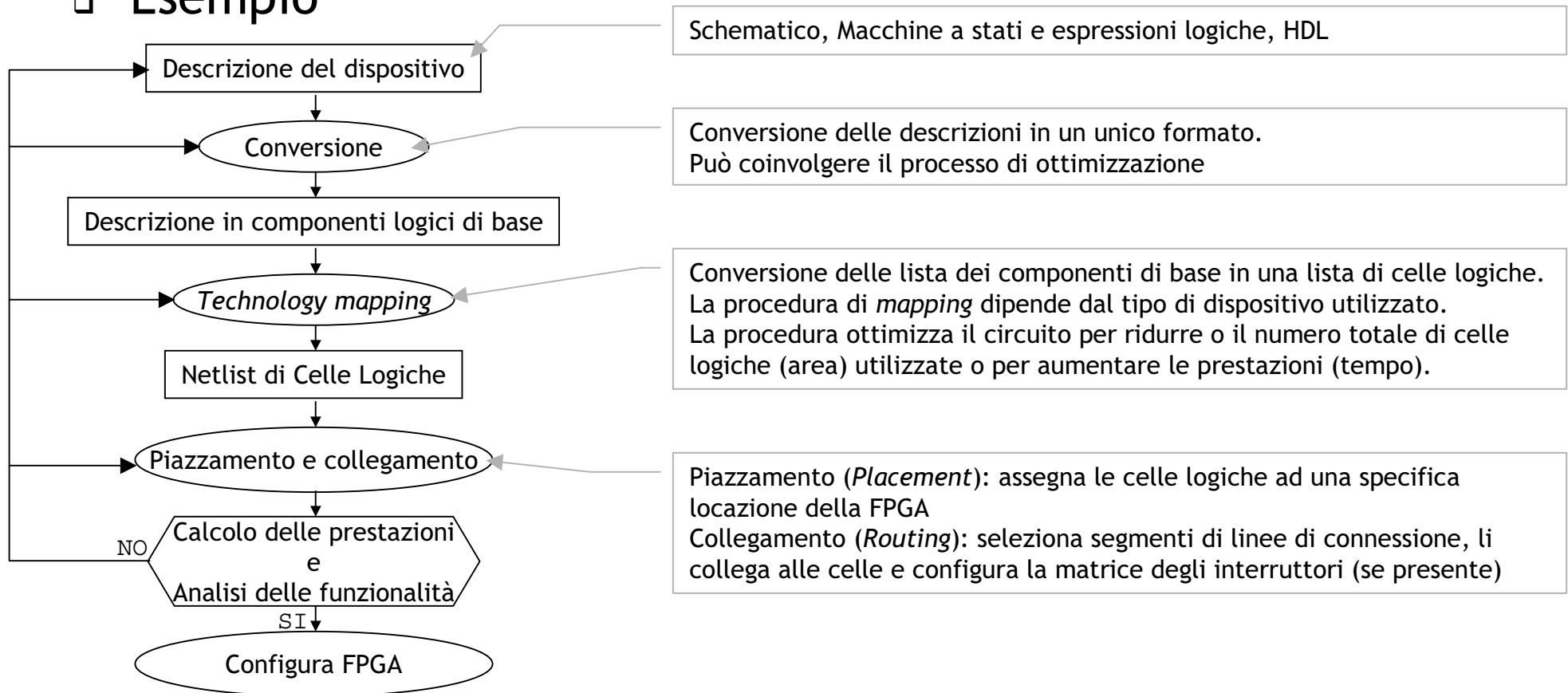
$$F(A, B) = A'B' f_{00} + A'B f_{01} + AB' f_{10} + AB f_{11}$$





# FPGA: Flusso di Progetto

## □ Esempio





## FPGA: *Technology mapping*

---

- Ottimizzazione logica (nel passo di *conversione*)
  - Ristruttura la rete, senza modificare la funzionalità, ottimizzando una o più specifiche cifre di merito senza tener conto della tecnologia a cui ci si riferirà
- Technology mapping:
  - Implementa la rete ottimizzata usando gli elementi di base disponibili e perseguendo l'obiettivo di ridurre il numero di elementi utilizzati (*area*) e/o il numero di livelli (*prestazioni*)
    - Nota: ridurre il numero di livelli comporta anche una riduzione dei ritardi causati dalle linee di connessione



# FPGA: Technology mapping

## □ Technology mapping:

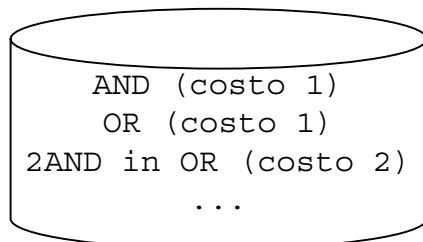
### - Due approcci:

- Basati su librerie di componenti.
- Basati su metodi specifici per LUT/MUX.

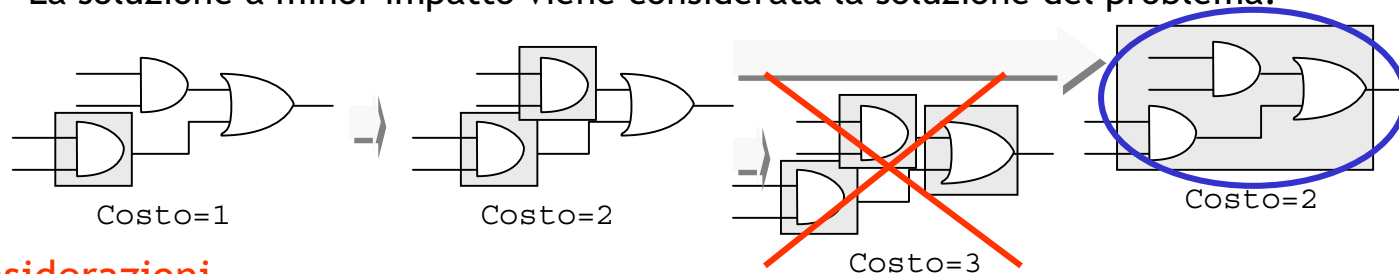
### - Approcci basati su librerie di componenti

#### • metodo

- La rete viene attraversata a partire dagli ingressi cercando, ad ogni nodo della struttura incontrato, una corrispondente funzione di libreria tra quelle disponibili
- Quando uno o più nodi verificano la corrispondenza, l'elemento di libreria viene utilizzato come sostituto dei nodi ed il costo viene incrementato
- La soluzione a minor impatto viene considerata la soluzione del problema.



Libreria



#### • Considerazioni

- Questo approccio **non può essere utilizzato direttamente su una FPGA** a causa dell'elevato numero di funzioni che una LUT/MUX può implementare.
  - » una LUT a  $k$  ingressi può implementare  $2^k$  differenti funzioni booleane richiedendo una libreria di possibili componenti praticamente ingestibile.



## FPGA: *Technology mapping*

---

- Technology mapping (cont.):
  - Approcci Basato su metodi specifici per LUT/MUX
    - Metodi basati sul fatto che una LUT/MUX implementa tutte le funzioni a  $k$  variabili.
    - L'obiettivo perseguito è quello di identificare una corrispondenza tra il numero degli ingressi di una sotto-rete logica ed il numero delle variabili trattate dal componente (LUT/MUX).
      - Non si ricerca alcuna corrispondenza con la funzionalità come per l'approccio basato su librerie di componenti.
    - Metodo:
      - Modello: grafo composto da nodi e archi (DAG). Ogni nodo contiene una rete combinatoria a 2 livelli ad una sola uscita; gli archi connettono i nodi
      - 3 passi
        - » **Decomposizione:** decompone i nodi non ammissibili in nodi ammissibili (es: una funzione a 20 ingressi non può essere mappata su una sola LUT a 3 ingressi)
        - » **Decomposizione e Copertura:** decompone ulteriormente i nodi mentre la copertura ricerca una soluzione ottima nell'utilizzo delle risorse.
        - » **Modifica della copertura nel caso di nodi collegati a più nodi contemporaneamente (nodi di fanout) e/o di nodi che presentano riconvergenze:** si sfruttano questi due aspetti per ulteriori ottimizzazioni (aspetto non trattato).



# FPGA: Technology mapping

## □ Decomposizione di nodi non ammissibili

### - Decomposizione disgiuntiva

$$F = ab + ab'cd + a'bc' + a'bd'$$



Free set = {a, b}  
Bound set = {c, d}

a	b	funzioni residue
0	0	0
0	1	(cd)'
1	0	cd
1	1	1

### - Decomposizione algebrica

$$F = ac + bc + bd + ce$$



$$F = cY + bd; \text{ con } Y = a + b + e$$

### - Decomposizione AND-OR

$$F = ab + ac + bc$$



$$V = ab; W = ac; X = bc; Y = V + W; F = Y + X$$

Nota: le decomposizioni non possono essere sufficienti per decomporre tutti i nodi non ammissibili

### - Cofattorizzazione di Shannon

- produce sempre un insieme di nodi ammissibili

$$F = abcd + a'b'eg + c'd'e'g';$$

(6 variabili)

$$F = a f_a + a' f_{a'}; \text{ con } f_a = bcd + c'd'e'g'; \text{ e } f_{a'} = b'eg + c'd'e'g';$$

(3 variabili)                      (5 variabili)                      (5 variabili)



## FPGA: *Technology mapping*

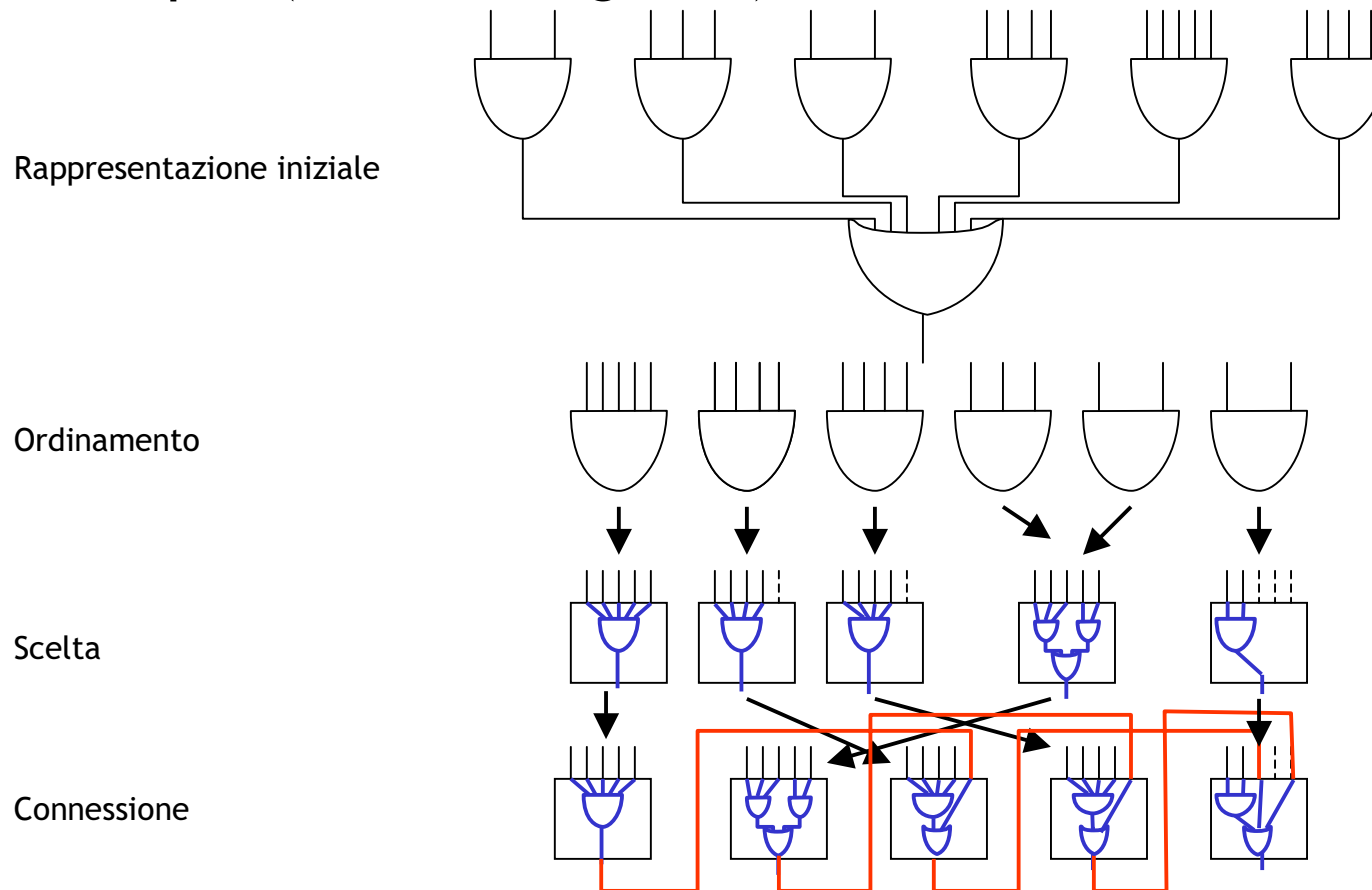
---

- Decomposizione e Copertura:
  - Viene applicata una procedura di copertura simile a quella basata su librerie di celle ad una struttura su cui è stata attuata una decomposizione AND-OR su tutti nodi prodotti dalla decomposizione.
  - 5 passi
    - Passo 1: decomposizione AND-OR
    - Passo 2: scelta dei termini AND di ingresso ad un nodo OR che possono essere allocati su di una singola LUT/MUX
      - Si uniscono i nodi di ingresso AND usati dal gruppo AND-OR con l'obiettivo di minimizzare la quantità di risorse utilizzate (problema di ottimizzazione combinatoria)
        - » Lista in ordine decrescente delle entità AND (es: 2 3 2 3 4 → 4 3 3 2 2)
        - » Seguendo l'ordine si riempiono le risorse puntando al massimo sfruttamento (es: 3 sotto-gruppi AND-OR per LUT a 6 ingressi - 4 2; 3 3 ; 2)
    - Passo3: connessione degli elementi identificati per implementare il nodo da realizzare
      - I sotto-gruppi AND-OR vengono nuovamente ordinati in base allo sfruttamento e si connettono le uscite ad ingressi non utilizzati di altri gruppi



# FPGA: *Technology mapping*

- Esempio (LUT a 5 ingressi):





# Dispositivi Programmabili

Modalità di programmazione

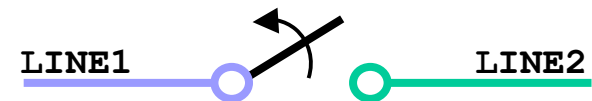
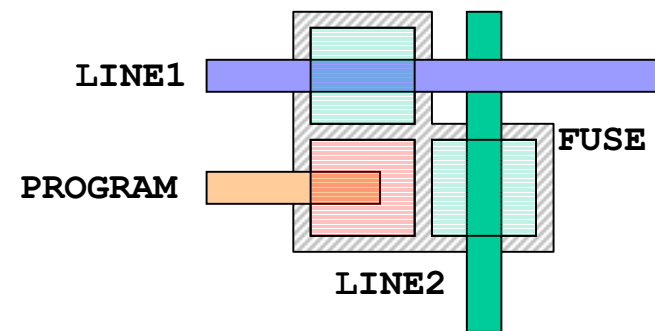
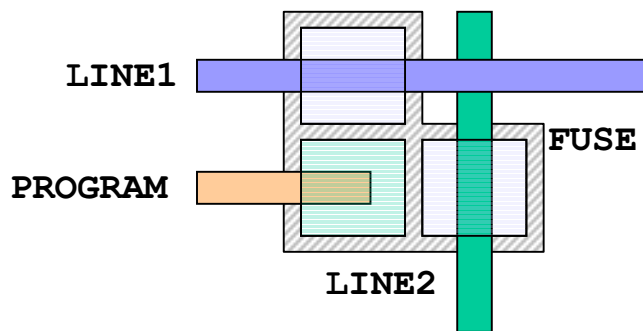
**Approfondimento**

---



## Fuse (OTP)

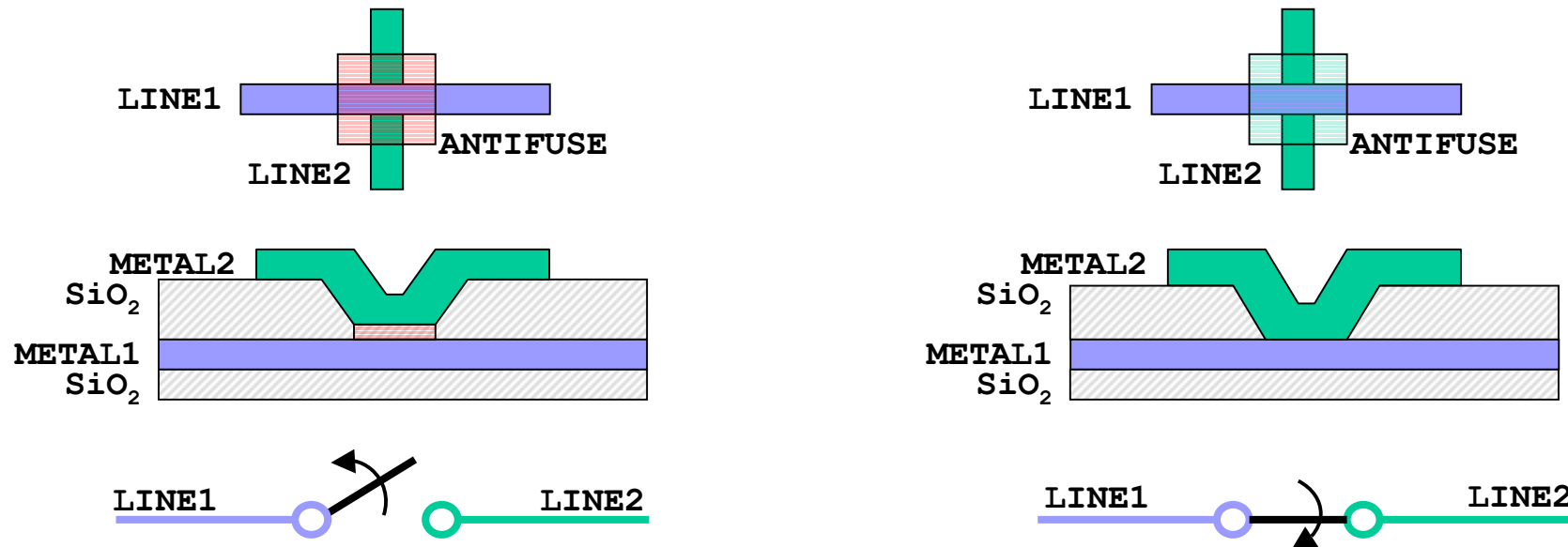
- Le linee del dispositivo sono prodotte in modo da essere sempre connesse
  - La programmazione consiste nel “bruciare” (fuse) alcune connessioni in modo da mantenere solo quelle necessarie.
  - La programmazione avviene mediante una tensione più elevata di quella di normale funzionamento.





## Antifuse (OTP)

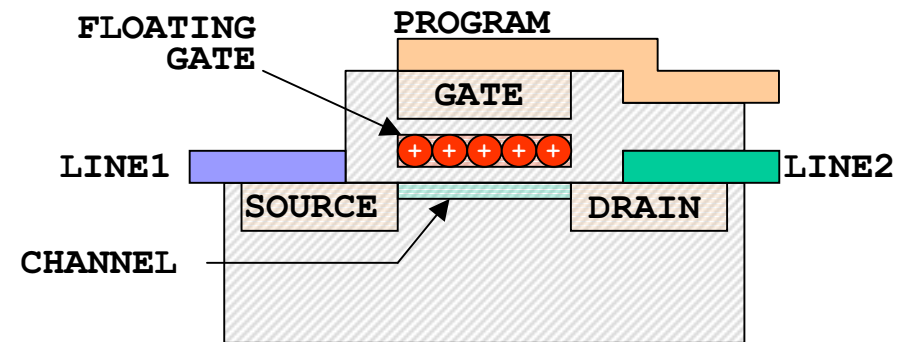
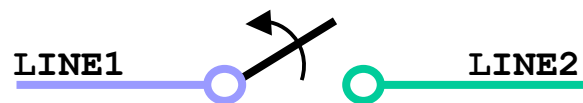
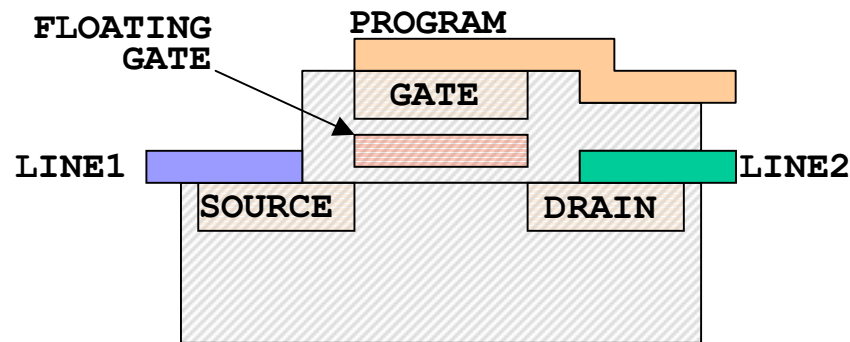
- Le linee del dispositivo sono prodotte in modo da essere sempre disconnesse
  - La programmazione consiste nel “creare” (antifuse) le connessioni necessarie
  - La programmazione avviene mediante una tensione più elevata di quella di normale funzionamento





## E<sup>2</sup>PROM (Reprogrammable)

- Le linee del dispositivo sono prodotte in modo da essere sempre disconnesse
  - La programmazione consiste nel depositare carica sul floating gate del transistor in modo da mantenerlo in conduzione





## SRAM (Reprogrammable)

- Le linee del dispositivo sono prodotte in modo da essere sempre disconnesse
  - La programmazione consiste nel memorizzare un valore logico (0 o 1) in una cella di RAM statica

