



Dispositivi Logici Programmabili

Dispositivi Programmabili a due livelli

Introduzione

ROM (Read Only Memory)

PLA (Programmable Logic Array)

PAL (Programmable Array Logic)

PLA e PAL avanzate

10/12/03



Logiche programmabili

- Sono dispositivi hardware che mettono a disposizione componenti logici più o meno complessi che possono essere connessi tra loro (**programmazione delle connessioni**) a seconda delle esigenze di progetto
- Dispongono di:
 - Componenti logici (Porte logiche, Flip-flop, Buffer...)
 - Linee di connessione
- Tipologie di dispositivi programmabili
 - ROM (Read-Only Memory), PLA (Programmable Logic Array), PAL (Programmable Array Logic): **dispositivi logici programmabili a 2 livelli**
 - il termine *2 livelli* indica che il dispositivo base è costituito da 1 sezione AND e da 1 sezione OR disgiunte
 - CPLD
 - FPGA



Logiche programmabili - *Modalità di programmazione*

- **Programmabili una sola volta** (*One-Time Programmable* - OTP) - durante la fase **non attiva** del dispositivo
 - **Fuse**: le connessioni tra le linee sono inizialmente tutte attive. La fase di programmazione disattiva permanentemente le connessioni non utili.
 - **Antifuse**: Le connessioni tra le linee sono inizialmente tutte non attive. La fase di programmazione attiva permanentemente le connessioni utili

 - **Riprogrammabili** (*Reprogrammable*) - durante la fase **non attiva** del dispositivo
 - **E²PROM**: Le connessioni tra le linee, inizialmente tutte non attive, possono essere attivate e disattivate elettricamente (deposito di carica per conduzione)
 - **SRAM**: La connessione tra le linee, inizialmente tutte non attive, viene realizzata memorizzando nella cella di RAM statica il valore 0 o 1 (maggiore velocità di programmazione rispetto alla tecnologia E²PROM)

 - **Riconfigurabili** (*Reconfigurable*) - durante la fase **attiva** del dispositivo
 - **SRAM**: oltre ad una elevata velocità di programmazione, è richiesta anche la possibilità di intervenire separatamente su parti del dispositivo.
-



Connessioni

□ Connessioni globali

- linea che attraversa buona parte del dispositivo e che è condivisa da molti elementi logici (elevati ritardi, può essere usata come uscita di un solo elemento logico limitandone la flessibilità)
- sono caratteristiche dei dispositivi logici programmabili a 2 livelli (**ROM, PLA, PAL**)
- e dei Complex Programmable Logic Devices (**CPLD**).

□ Connessioni locali e distribuite

- Linea che attraversa una parte ridotta del dispositivo e che è condivisa da pochi elementi logici (ritardi contenuti, coesistenza di differenti linee di lunghezza differente, quindi elevata flessibilità)
- sono caratteristiche dei Field Programmable Gate Array (**FPGA**).

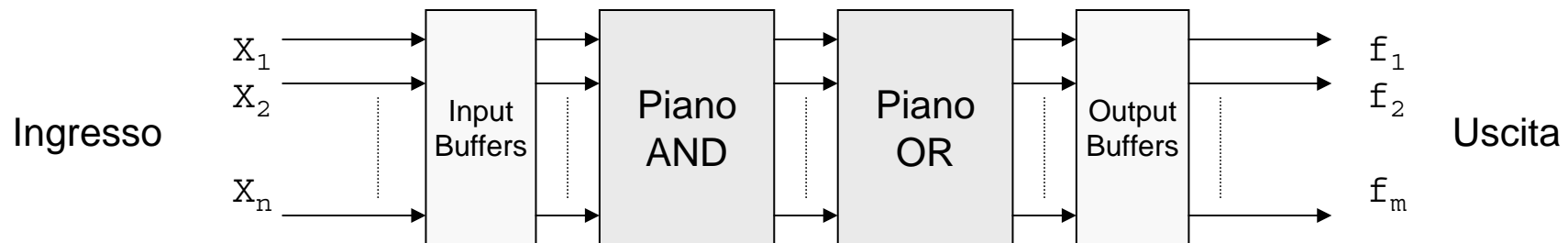


Logiche programmabili a 2 livelli

- Nello schema base realizzano funzioni a 2 livelli del tipo SOP a n ingressi e m uscite

$$f_i = f_i(x_1, x_2, \dots, x_n) \text{ con } i = \{1, 2, \dots, m\}$$

- Dispongono di:
 - Un numero di ingressi fissato;
 - Un numero di uscite fissato;
 - Un **piano AND**, per la costruzione dei mintermini o degli implicant;
 - Un **piano OR**, per la somma dei mintermini o degli implicant;
 - Buffer di ingresso e di uscita (per ragioni elettriche e funzionali).
 - In seguito, negli schemi logici, i buffer non saranno riportati per comodità sebbene siano sempre presenti





Piano AND - Piano OR: *Esempio* 1

- Prima forma canonica della funzione a più uscite:

$$f_1 = a'b'c + a'bc + ab'c' + ab'c + abc' + abc$$

$$f_2 = a'b'c + ab'c + abc$$

$$f_3 = a'b'c + a'bc' + a'bc + ab'c' + ab'c + abc'$$

$$f_4 = a'b'c' + a'b'c + a'bc + ab'c' + ab'c + abc' + abc$$

$$f_5 = a'bc' + ab'c' + ab'c + abc' + abc$$

- Tabella della verità della funzione a più uscite:

abc	f ₁	f ₂	f ₃	f ₄	f ₅
000	0	0	0	1	0
001	1	1	1	1	0
010	0	0	1	0	1
011	1	0	1	1	0
100	1	0	1	1	1
101	1	1	1	1	1
110	1	0	1	1	1
111	1	1	0	1	1

Il **piano AND** realizza tutti i **mintermini**

Il **piano OR** realizza le **singole funzioni di uscita**



Piano AND - Piano OR: *Esempio* 2

Prima forma canonica della funzione a più uscite:

$$f_1 = a'b'c + a'bc + ab'c' + ab'c + abc' + abc$$

$$f_2 = a'b'c + ab'c + abc$$

$$f_3 = a'b'c + a'bc' + a'bc + ab'c' + ab'c + abc'$$

$$f_4 = a'b'c' + a'b'c + a'bc + ab'c' + ab'c + abc' + abc$$

$$f_5 = a'bc' + ab'c' + ab'c + abc' + abc$$



Funzione a più uscite ottimizzata

$$f_1 = a + bc + b'c'$$

$$f_2 = ac$$

$$f_3 = ab' + a'c + ac' + bc'$$

$$f_4 = a + b' + bc$$

$$f_5 = a + bc'$$



**Il piano AND
realizza questi
termini
prodotto**

**Il piano OR realizza le
singole funzioni di uscita**

	abc	f1	f2	f3	f4	f5
P1=a	1--	1	0	0	0	0
P2=bc	-11	1	0	0	0	0
P3=b'c'	-00	1	0	0	0	0
P4=ac	1-1	0	1	0	0	0
P5=ab'	10-	0	0	1	0	0
P6=a'c	0-1	0	0	1	0	0
P7=ac'	1-0	0	0	1	0	0
P8=bc'	-10	0	0	1	0	0
P1=P9=a	1--	0	0	0	1	0
P10=b'	-0-	0	0	0	1	0
P2=P11=bc	-11	0	0	0	1	0
P1=P12=a	1--	0	0	0	0	1
P8=P13=bc'	-10	0	0	0	0	1



Logiche programmabili a 2 livelli

- **Read-Only Memory (ROM)**
 - Piano AND fissato.
 - Implementa **tutti** i possibili mintermini (decoder).
 - Piano di OR adattabile.
- **Programmable Logic Array (PLA)**
 - Piano AND programmabile.
 - E' dato il numero di termini prodotto generabili, si programmano solo i mintermini/implicanti necessari.
 - Piano OR programmabile.
- **Programmable Array Logic (PAL)**
 - Piano AND programmabile.
 - E' dato il numero di termini prodotto generabili, si programmano solo i mintermini/implicanti necessari.
 - Piano di OR fissato.
 - Ogni funzione (OR) può essere costruita solo con un certo sottoinsieme (cablato) di termini prodotto.



Logiche programmabili a 2 livelli: *campi di applicazione*

- Reti combinatorie a due livelli non ottimizzate:
 - Read-Only Memory (ROM)
 - Anche PLA e PAL

- Reti combinatorie a due livelli ottimizzate:
 - Programmable Logic Array (PLA), Programmable Array Logic (PAL)

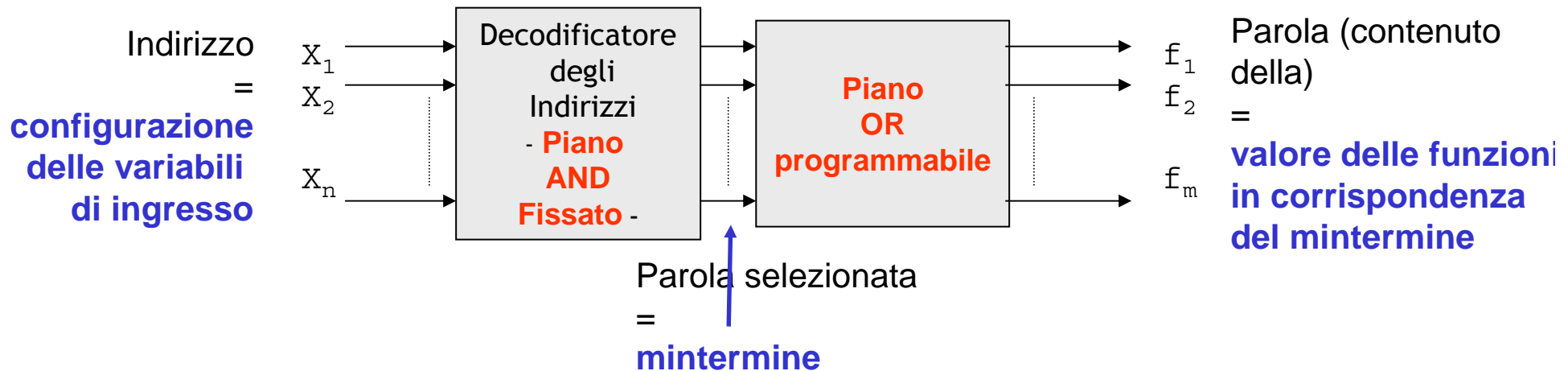
- Reti combinatorie multi livello costituite da reti a due livelli ottimizzate:
 - PLA e PAL con retroazione

- Macchine Sequenziali Sincrone con reti combinatorie multi livello costituite da reti a due livelli ottimizzate:
 - PLA e PAL con retroazione e registri



Read-Only Memory (ROM)

- Un Memoria a Sola Lettura (ROM) implementa la prima forma canonica di m funzioni di uscita a n ingressi
 - Somma di Prodotti (SOP)
- In una ROM, una configurazione di ingresso, denominata *indirizzo*, è associata una configurazione di uscita, denominata *parola*





ROM: *decodificatore degli indirizzi*

□ **Decodificatore degli indirizzi (Address decoder)**

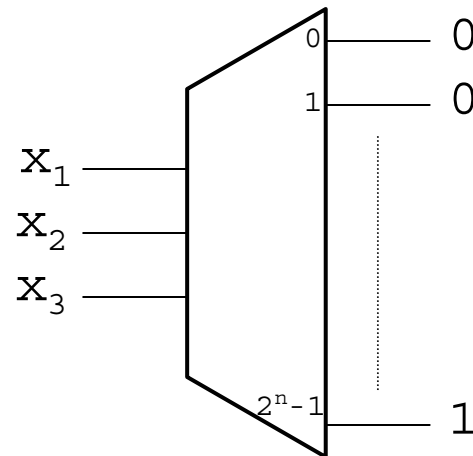
- Il decodificatore degli indirizzi nelle memorie ROM realizza tutti i 2^n mintermini, dove n sono le variabili di ingresso x_i

□ **Infatti**

- Gli ingressi sono le variabili x_i
- Una ed una sola uscita è attiva alla volta; le uscite del decoder sono tutti i mintermini costruiti a partire dalle variabili di ingresso
 - Esempio di decodificatore a 3 ingressi:

Esempio:

$$x_1 x_2 x_3 = 111$$

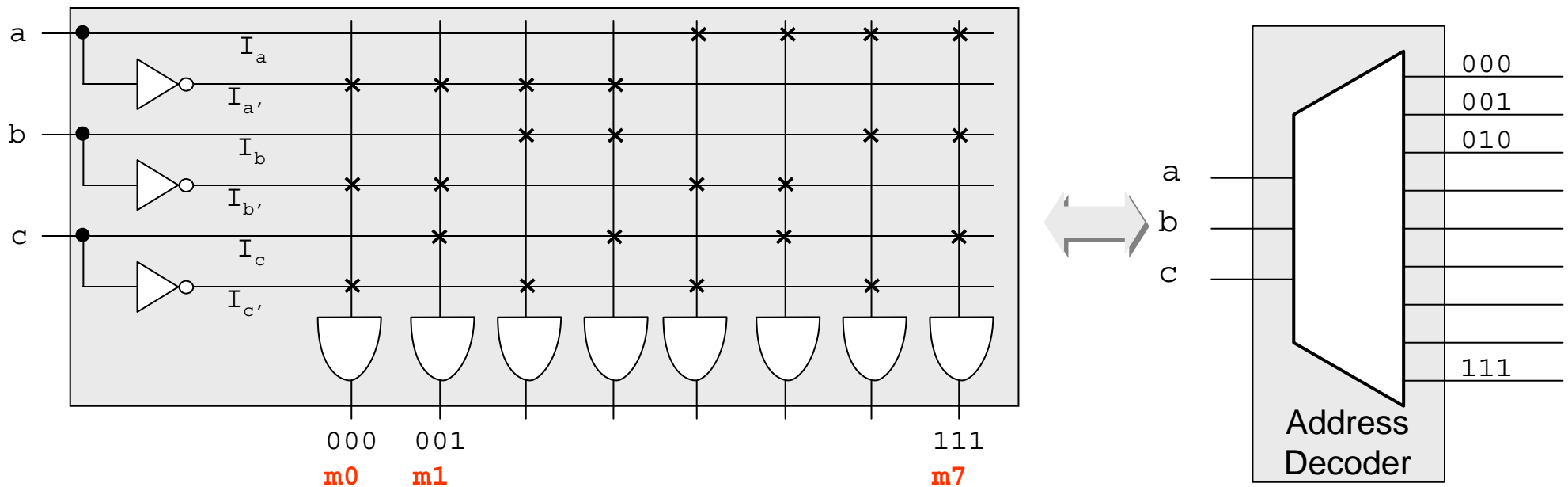




ROM: *piano AND e decodificatore degli indirizzi*

- Schema logico del piano di AND che implementa l'Address Decoder
 - Per semplicità si utilizza la rappresentazione che si riferisce al *decoder*.

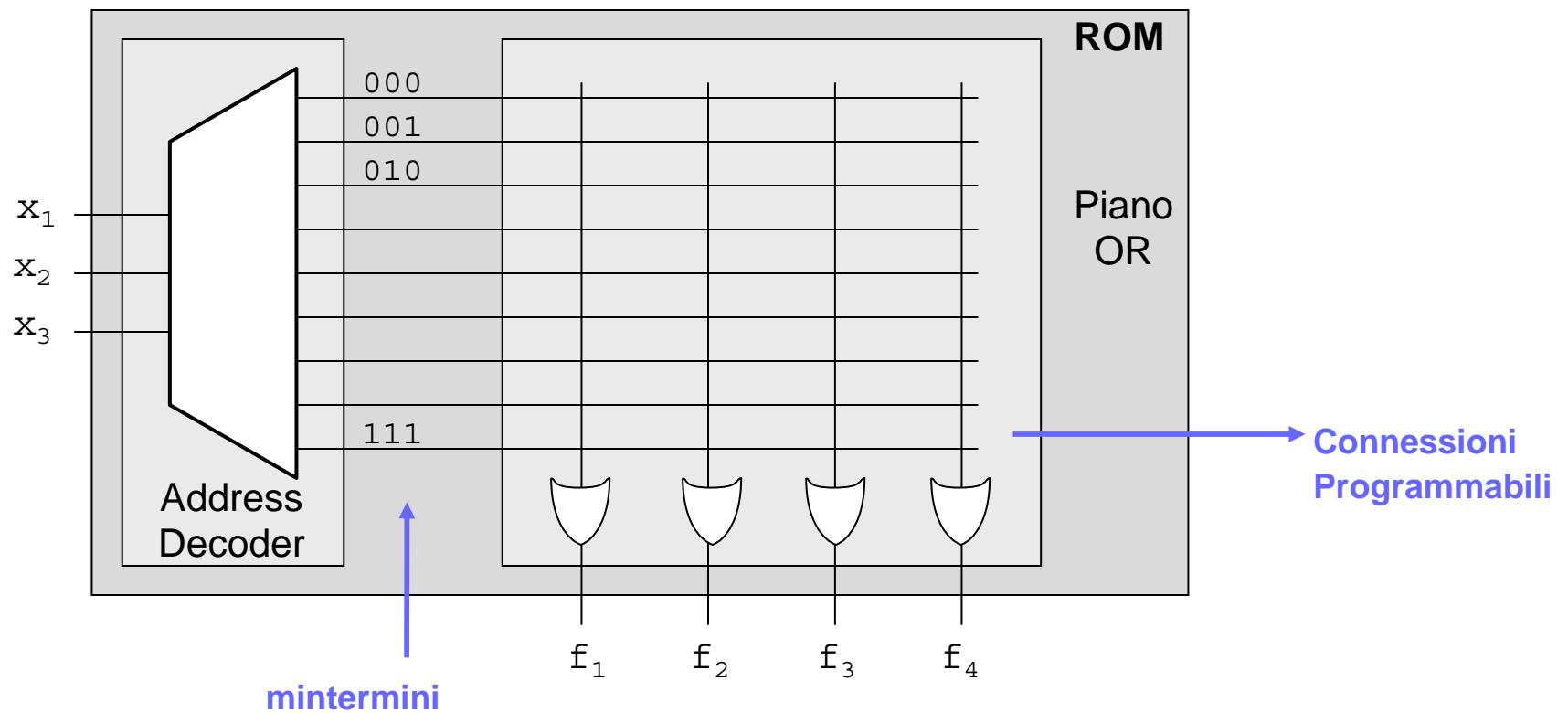
Significato: ogni croce indica quali variabili sono coinvolte, il tipo di porta indica schematicamente come tali variabili collegate tra loro.





ROM: *schema logico*

- Schema logico di una ROM:
 - Esempio di una ROM a 3 ingressi e 4 uscite (non programmata).





ROM: *Esempio*

□ Esempio:

- Prima forma canonica della funzione a più uscite:

$$f_1 = a'b'c + a'bc + ab'c' + ab'c + abc' + abc$$

$$f_2 = a'b'c + ab'c + abc$$

$$f_3 = a'b'c + a'bc' + a'bc + ab'c' + ab'c + abc'$$

$$f_4 = a'b'c' + a'b'c + a'bc + ab'c' + ab'c + abc' + abc$$

$$f_5 = a'bc' + ab'c' + ab'c + abc' + abc$$

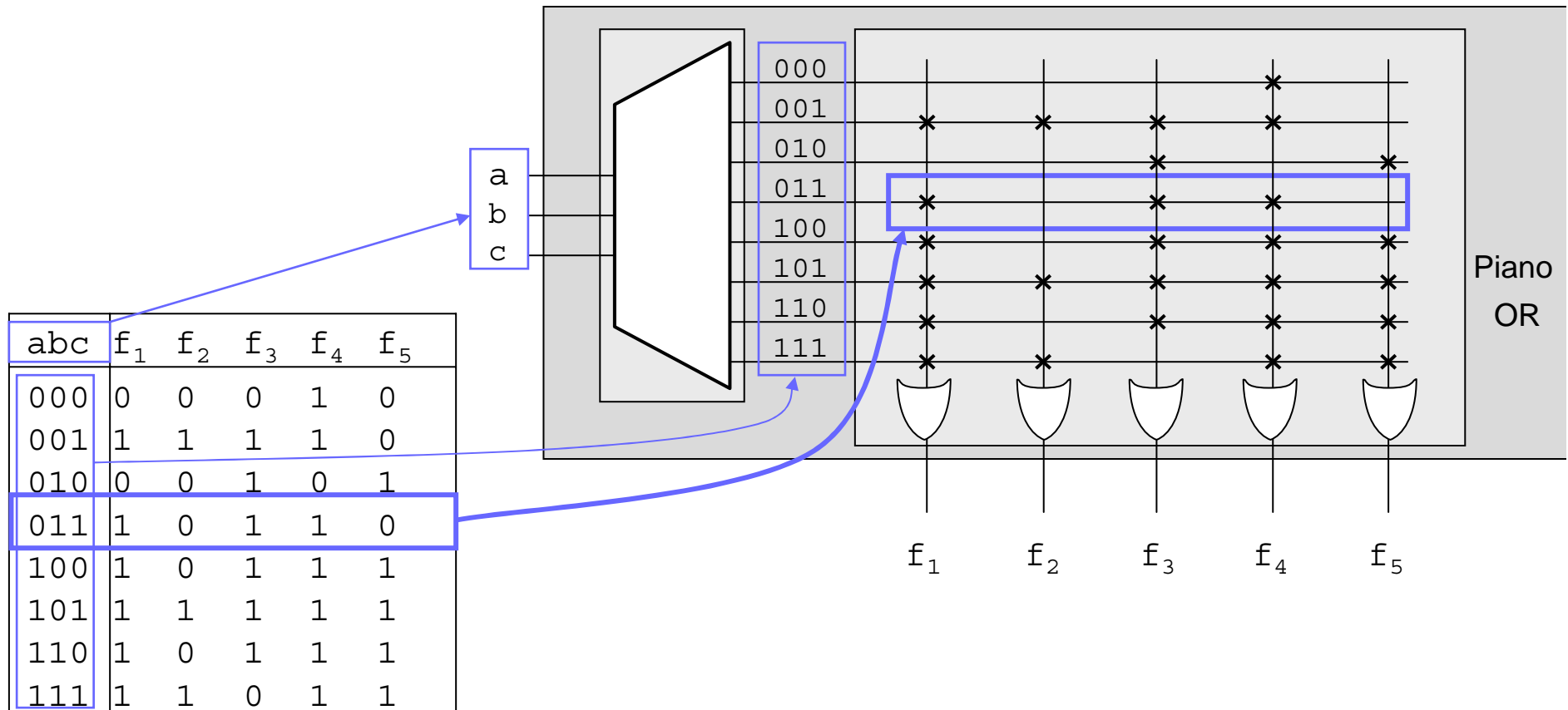
- Tabella della verità della funzione a più uscite:

abc	f ₁	f ₂	f ₃	f ₄	f ₅
000	0	0	0	1	0
001	1	1	1	1	0
010	0	0	1	0	1
011	1	0	1	1	0
100	1	0	1	1	1
101	1	1	1	1	1
110	1	0	1	1	1
111	1	1	0	1	1



ROM: *Esempio*

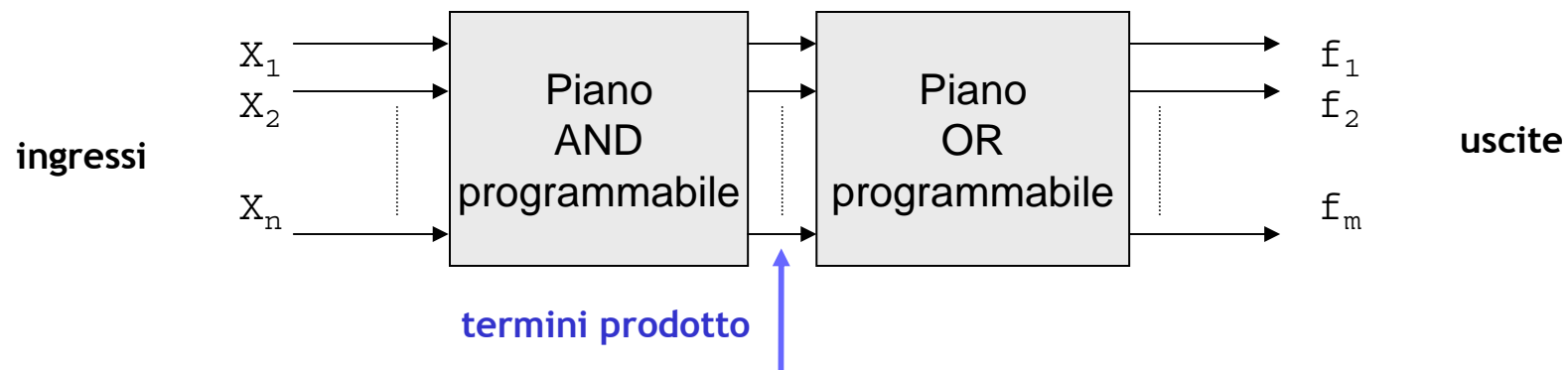
- Esempio (cont.):
 - Realizzazione della funzione a più uscite (aspetto logico):





Programmable Logic Array (PLA)

- Un Array Logico Programmabile (*Programmable Logic Array* - PLA) consente di implementare una somma di prodotti espressa in forma **minima** a due livelli (*somma di implicanti*)
 - Nota: l'estensione a forme non minime è naturale.
- In generale una PLA è definita da: **numero di ingressi** (n° variabili delle funzioni), **numero dei termini prodotto** generabili, **numero di uscite** (n° di funzioni realizzabili)

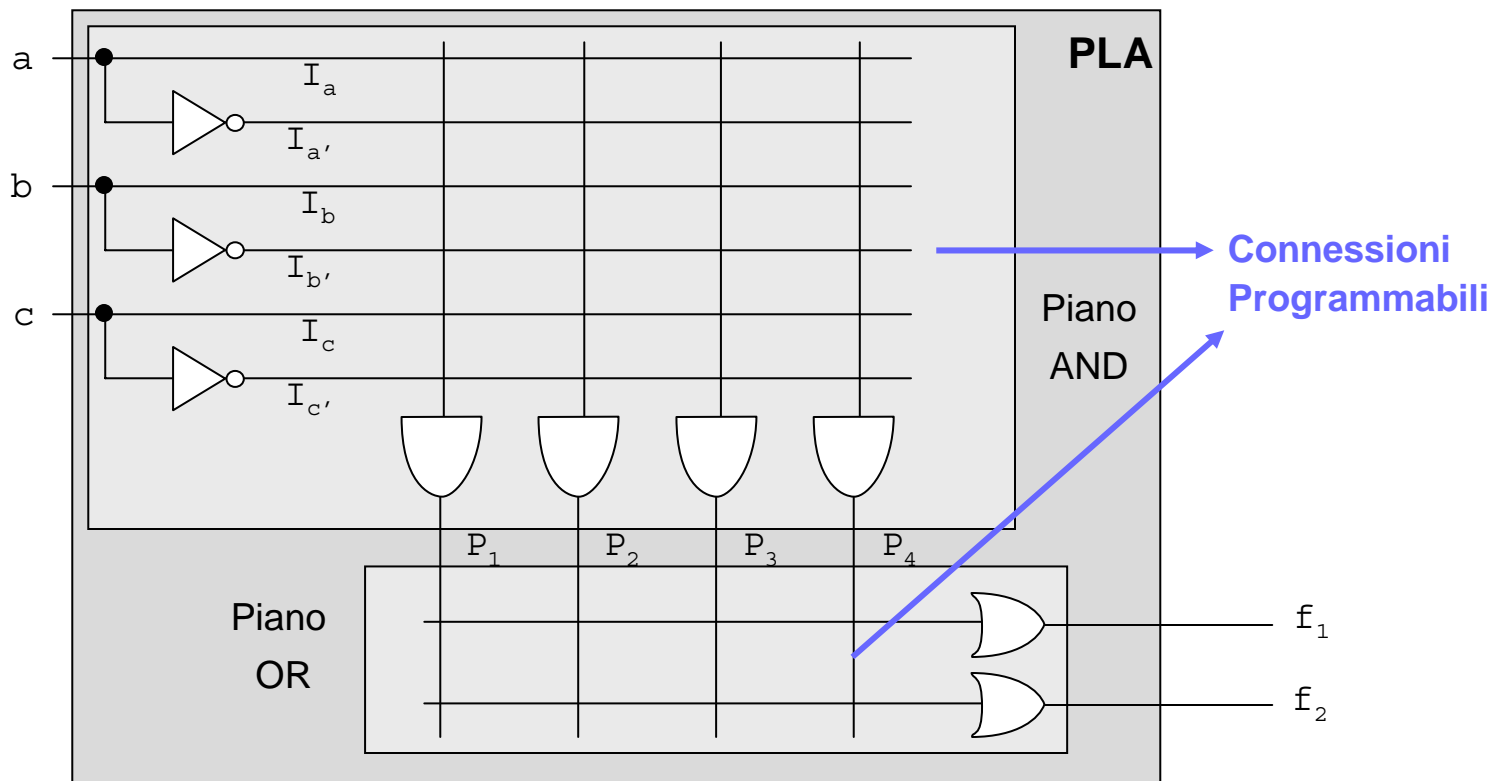




Programmable Logic Array (PLA): *schema logico*

□ Schema logico di una PLA

- Esempio di PLA a 3 ingressi, 2 uscite, 4 termini prodotto (non programmata):





PLA: *Esempio 1*

□ Esempio 1:

- Realizzazione delle funzioni:

$$f_1 = ab + ac' + a'b'c$$

$$f_2 = ab + ac + a'b'c$$

- Prodotti:

$$P_1 = ab$$

$$P_2 = ac$$

$$P_3 = ac'$$

$$P_4 = a'b'c$$

- Somme:

$$f_1 = P_1 + P_3 + P_4$$

$$f_2 = P_1 + P_2 + P_4$$

Formato PLA:

11- 10

1-0 10

001 10

11- 01

1-1 01

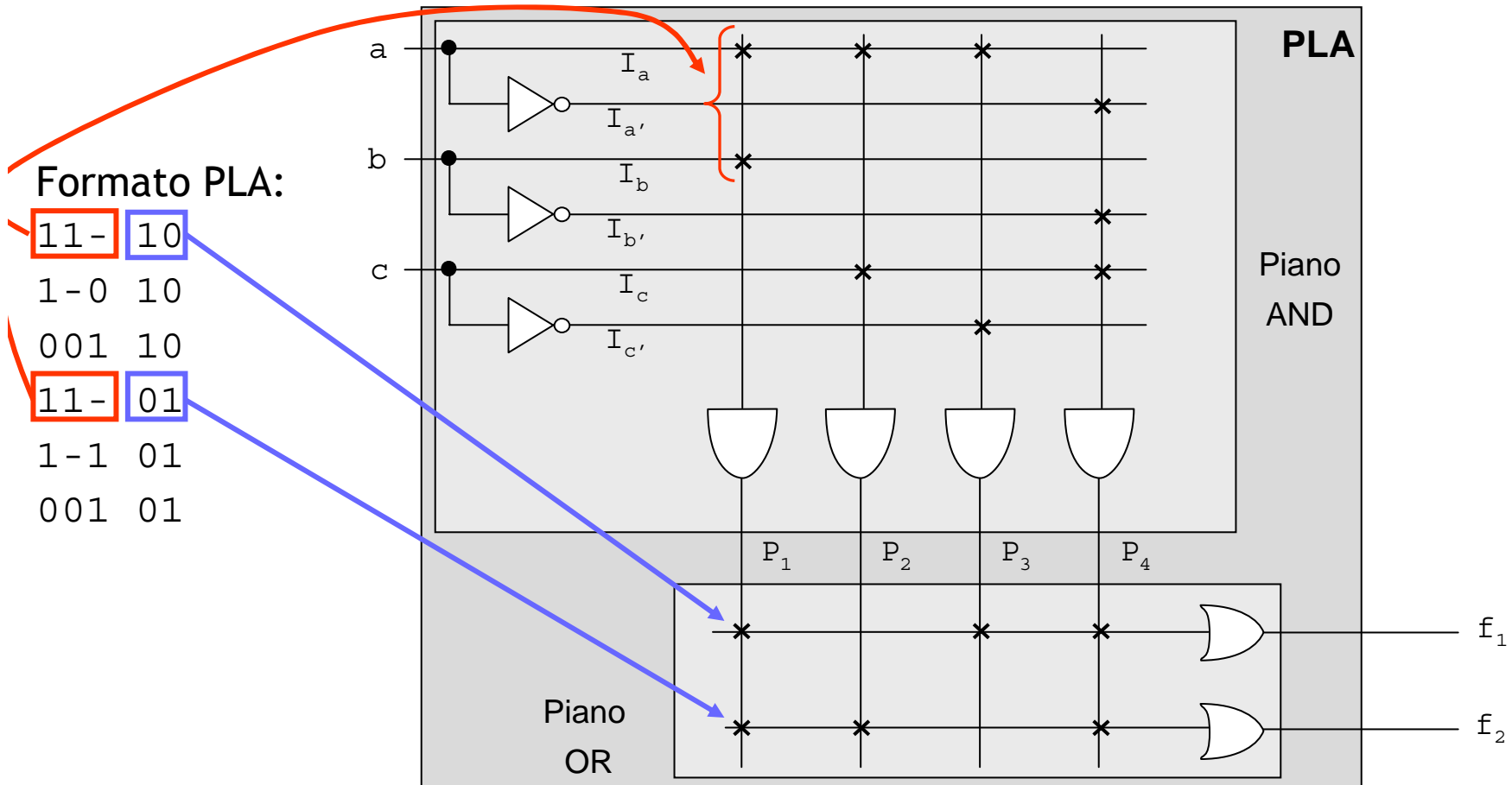
001 01





PLA: *Esempio 1*

- Esempio 1 (cont.): (PLA programmata per le funzioni f_1 ed f_2)





PLA: *Esempio 2*

□ Esempio 2:

Prima forma canonica della funzione a più uscite:

$$f_1 = a'b'c + a'bc + ab'c' + ab'c + abc' + abc$$

$$f_2 = a'b'c + ab'c + abc$$

$$f_3 = a'b'c + a'bc' + a'bc + ab'c' + ab'c + abc'$$

$$f_4 = a'b'c' + a'b'c + a'bc + ab'c' + ab'c + abc' + abc$$

$$f_5 = a'bc' + ab'c' + ab'c + abc' + abc$$



Funzione a più uscite ottimizzata (espressioni logiche):

$$f_1 = a + bc + b'c'$$

$$f_2 = ac$$

$$f_3 = ab' + a'c + ac' + bc'$$

$$f_4 = a + b' + bc$$

$$f_5 = a + bc'$$

Formato PLA:

```
1-- 10000
-11 10000
-00 10000
1-1 01000
10- 00100
0-1 00100
1-0 00100
-10 00100
1-- 00010
-0- 00010
-11 00010
1-- 00001
-10 00001
```





PLA: *Esempio 2*

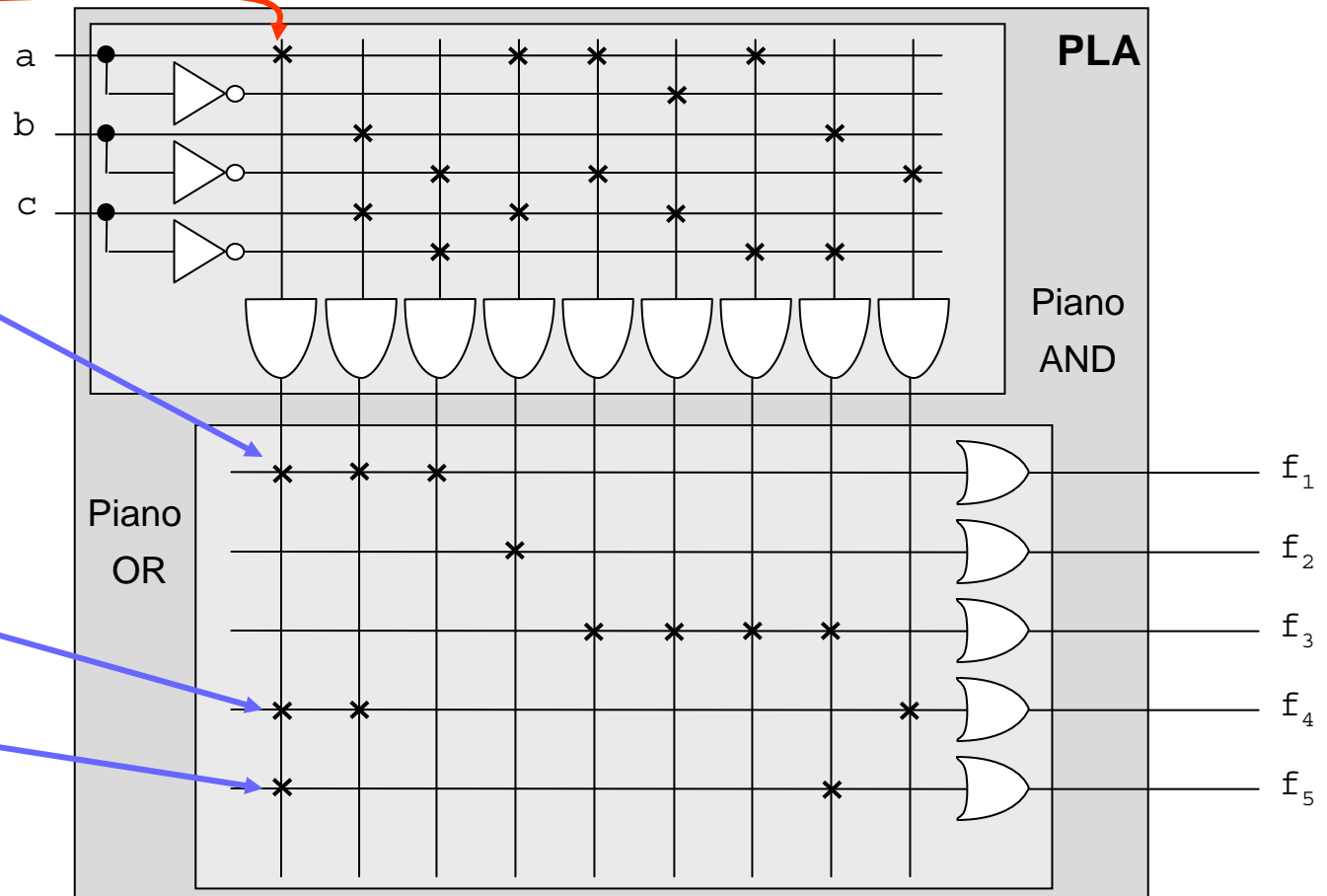
□ Esempio (cont.):

Formato PLA:

```

1-- 10000
-11 10000
-00 10000
1-1 01000
10- 00100
0-1 00100
1-0 00100
-10 00100
1-- 00010
-0- 00010
-11 00010
1-- 00001
-10 00001
  
```

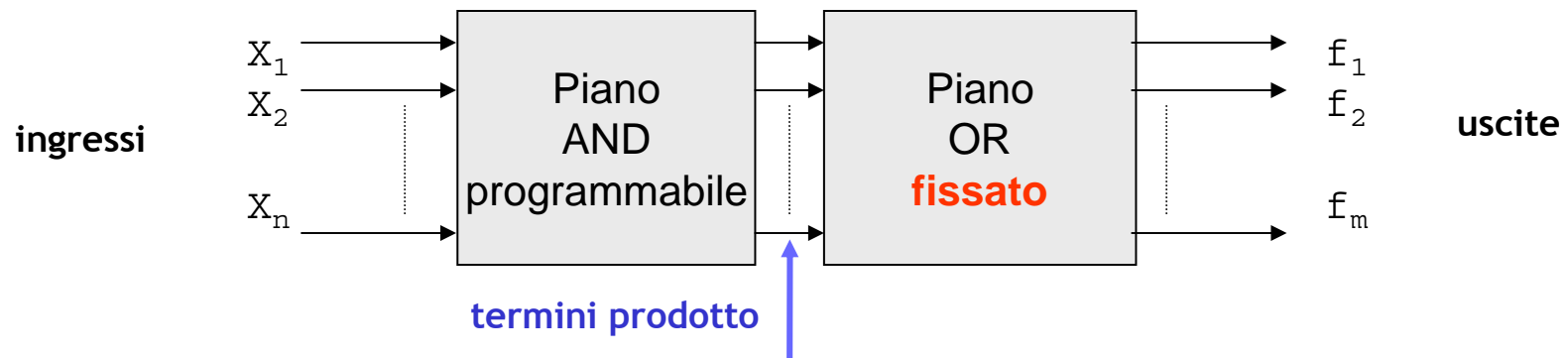
**13 termini prodotto
di cui 9 diversi**





Programmable Array Logic (PAL)

- Un Array Programmabile Logico (PAL) consente di implementare una somma di prodotti espressa in forma minima a due livelli (*somma di implicanti*)
 - PLA e PAL coprono lo stesso spazio d'applicazione.
- In generale una PAL è definita da: **numero di ingressi** (n° variabili delle funzioni), **numero dei termini prodotto** generabili, **numero di uscite** (n° di funzioni realizzabili)
 - Il piano OR fissato nasce da un trade-off tra il n° di ingressi per OR e il n° di OR

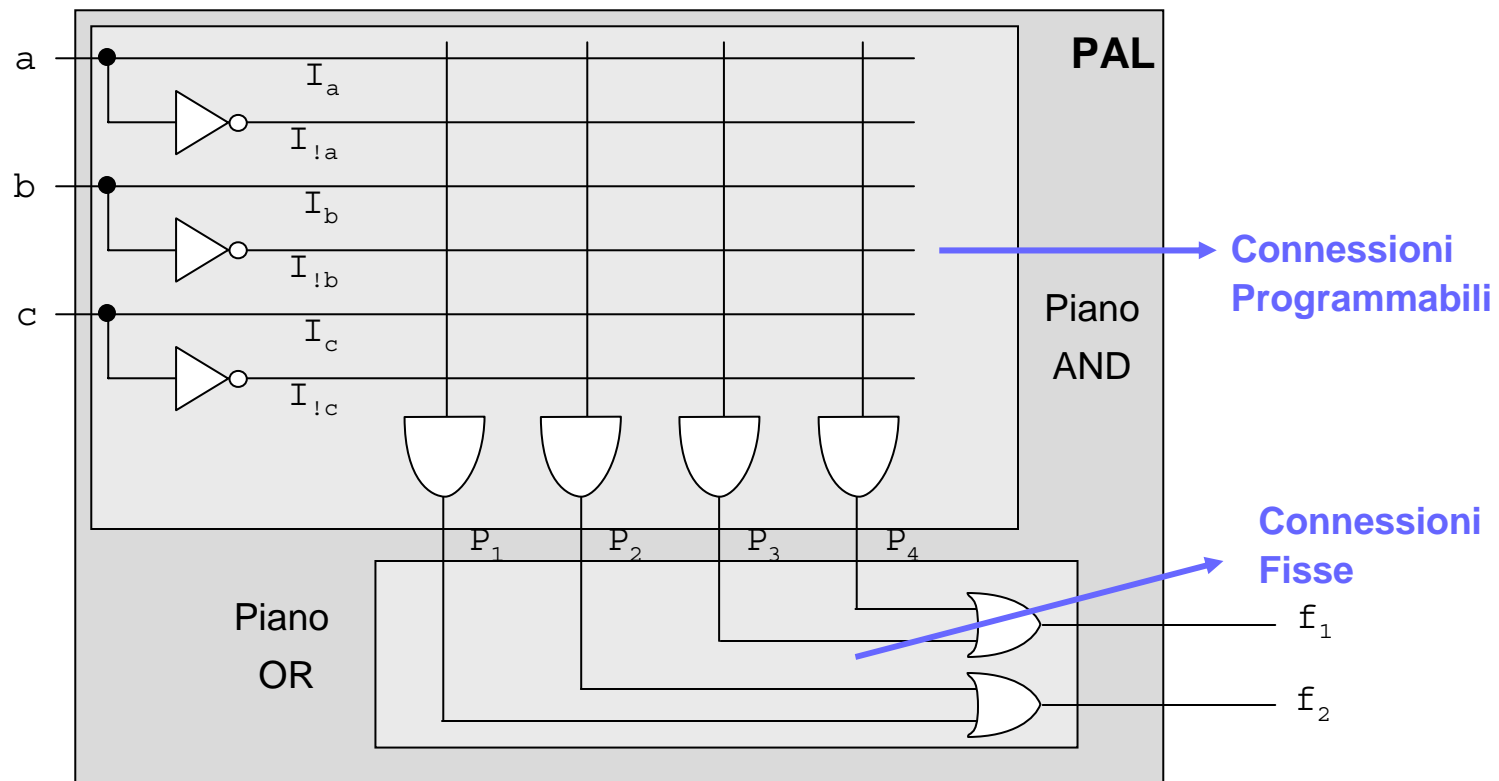




PAL: *schema logico*

□ Schema logico di una PAL

- Esempio di PAL a 3 ingressi, 4 termini prodotto, 2 uscite (non programmata):





PLA e PAL

- Nelle **PLA**, le uscite possono condividere termini prodotto
 - Nelle **PAL**, le uscite **non possono condividere termini prodotto** e inoltre il **numero di ingressi alle porte OR è fissato**
 - Il piano OR fissato può implicare comunque una realizzazione **multilivello**
 - A pari funzionalità da implementare, in caso di possibilità di condivisione di termini prodotto, la sezione AND di una PAL deve essere più grande (termini prodotto replicati) di quella di una PLA
 - Le PLA sono più lente delle PAL a causa della programmabilità della sezione OR: le connessioni fuse-based, o comunque programmate, presentano una resistenza maggiore rispetto a quelle cablate
-



PAL: Esempio 1 di PLA realizzato con PAL

- Realizzazione tramite una PAL (a 3 ingressi, 6 termini prodotto, OR a 3 ingressi e 2 uscite) delle funzioni:

$$f_1 = ab + ac' + a'b'c$$

$$f_2 = ab + ac + a'b'c$$

- Prodotti:

$$P_1 = ab$$

$$P_2 = ac'$$

$$P_3 = a'b'c$$

$$P_4 = ab$$

$$P_5 = ac$$

$$P_6 = a'b'c$$

- Somme:

$$f_1 = P_1 + P_2 + P_3$$

$$f_2 = P_4 + P_5 + P_6$$

Formato PAL:

11- 10

1-0 10

001 10

11- 01

1-1 01

001 01

E se fossero disponibili solo OR a 2 ingressi, e 4 uscite? **Realizzazione multi livello**

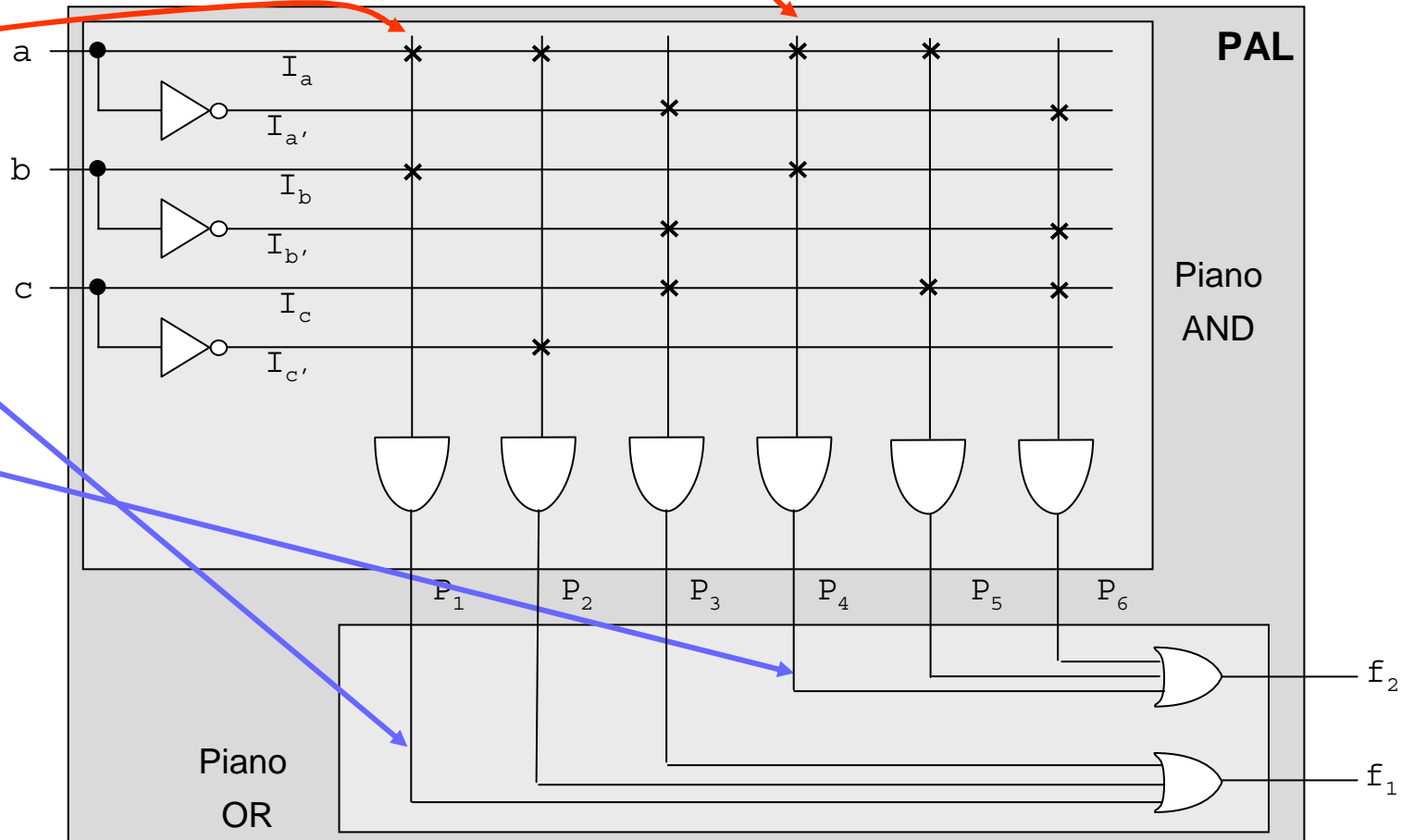


PAL: Esempio 1

- Esempio 1 (cont.): (PAL programmata per le funzioni f_1 ed f_2)

Formato PAL:

11-	10
1-0	10
001	10
11-	01
1-1	01
001	01





PLA e PAL Avanzate

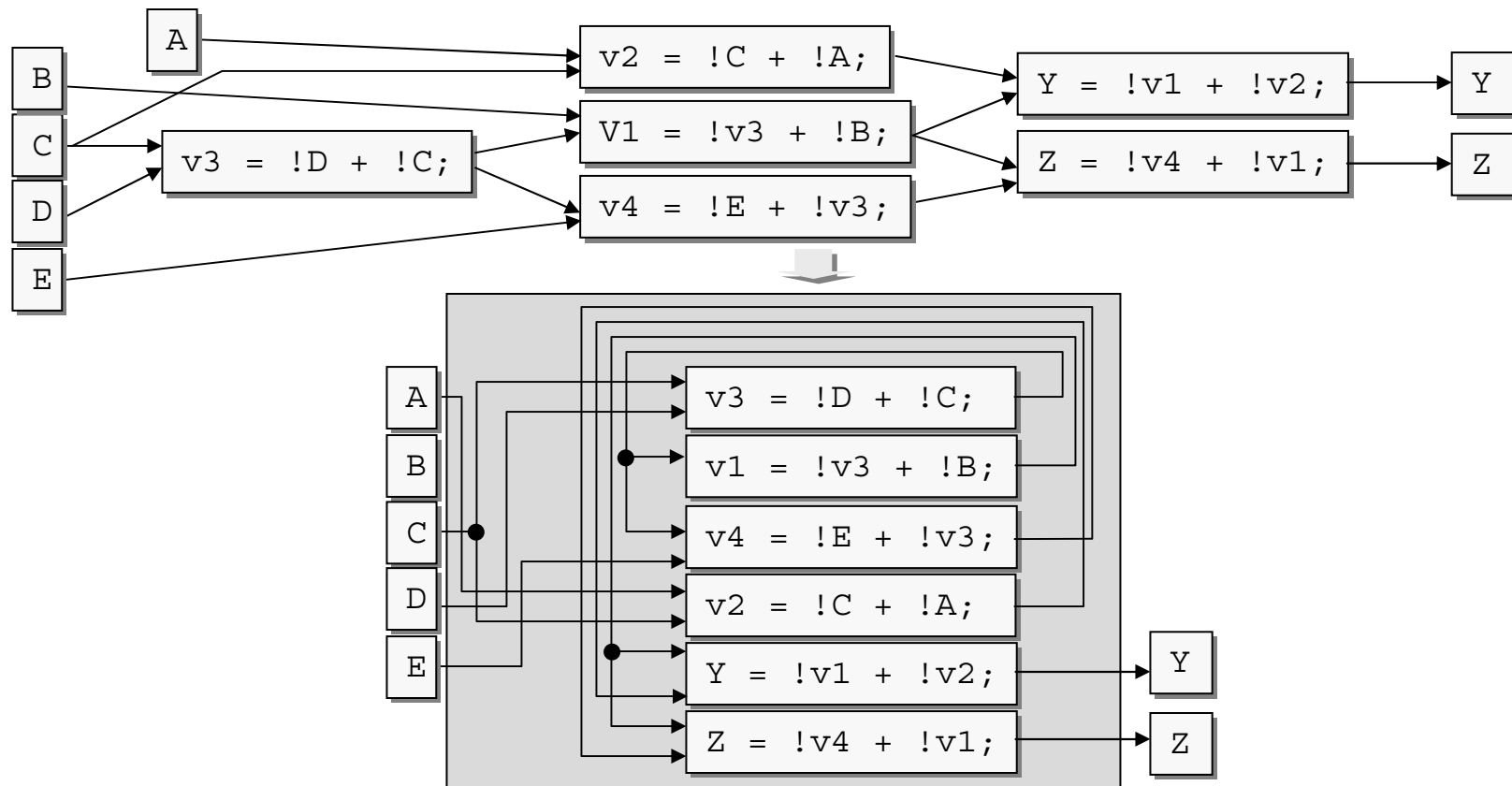
- ❑ Lo schema base mostrato consente di realizzare solo reti combinatorie a due livelli

- ❑ Questo limite è superato:
 - ❑ Introducendo delle linee di retroazione
 - Permette di implementare **reti combinatorie multi livello a più uscite**
 - ❑ Introducendo elementi di memoria (bistabili)
 - Permette di implementare **macchine sequenziali sincrone** in cui la parte combinatoria è costituita da una rete multi livello a più uscite



PLA avanzate: rete multi livello - esempio

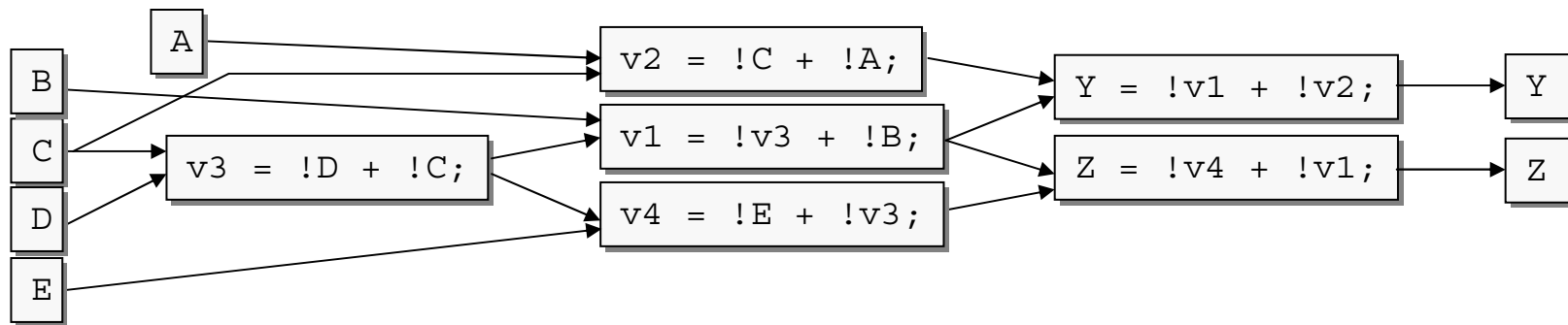
- Esempio di implementazione tramite PLA di una rete combinatoria multi livello a più uscite.





PLA: rete multi livello - esempio

- Esempio di implementazione di una rete combinatoria multi livello a più uscite.



Funzioni da realizzare (piano OR)

```
v1 = !v3 + !B
v2 = !C + !A
v3 = !D + !C
v4 = !E + !v3
Y = !v1 + !v2      uscita
Z = !v4 + !v1      uscita
```

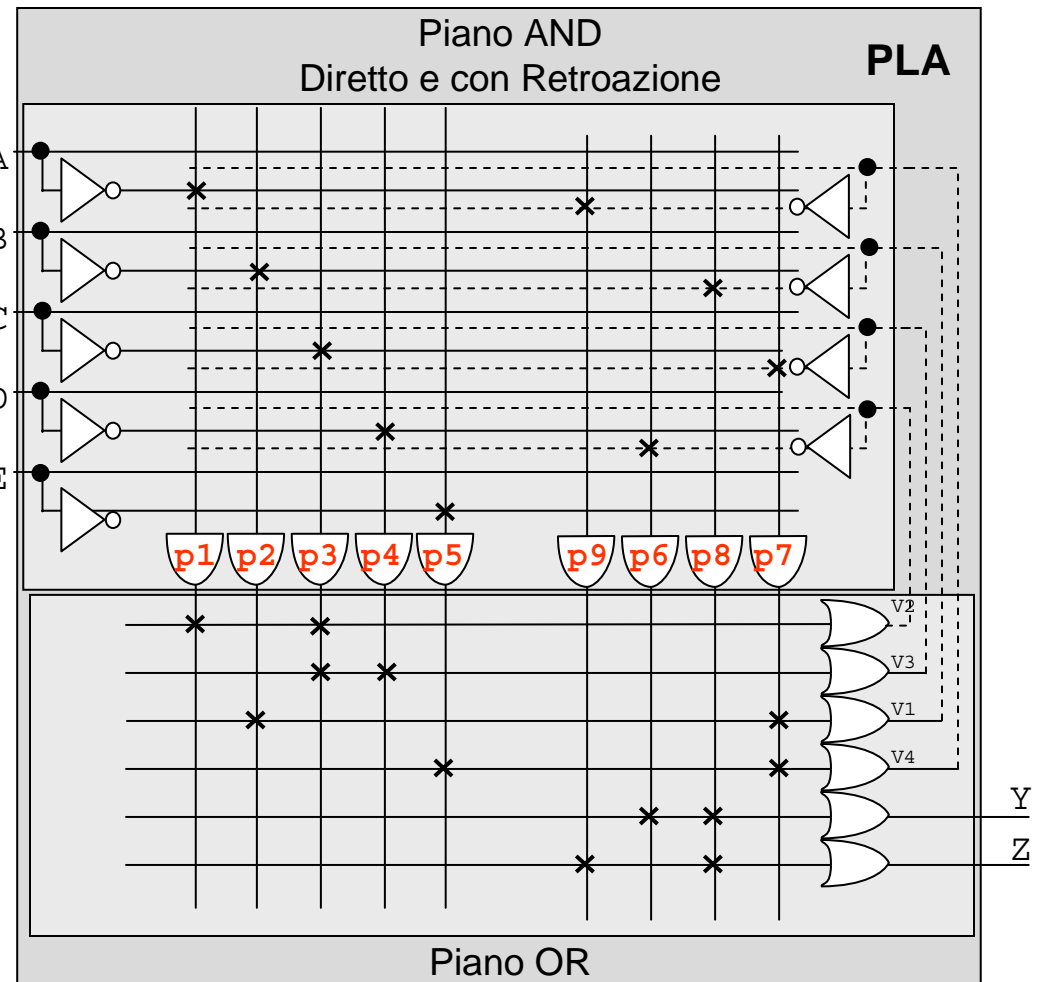
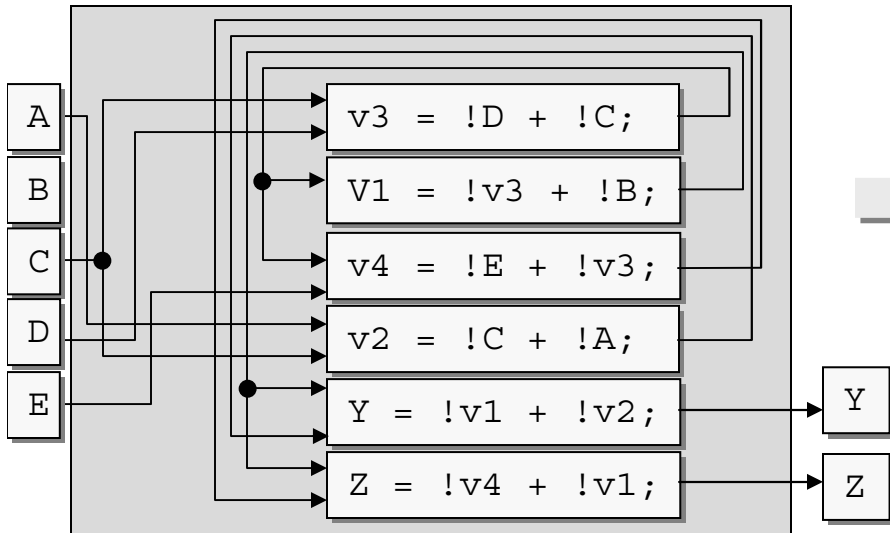
Termini prodotto da realizzare (piano AND)

```
p1 = !A           p6 = !v1
p2 = !B           p7 = !v2
p3 = !C           p8 = !v3
p4 = !D           p9 = !v4
p5 = !E
```



PLA: rete multi livello - esempio

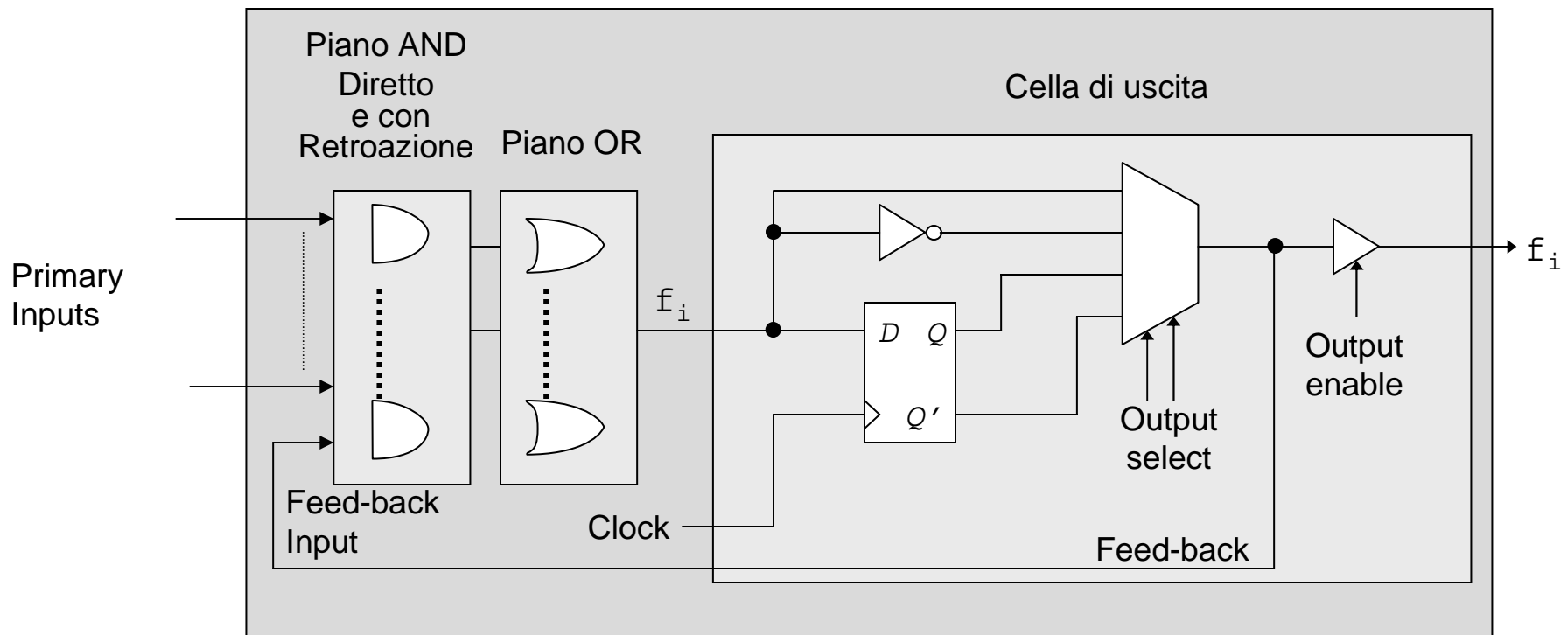
□ Esempio (cont):





PLA e PAL Avanzate: **struttura generale**

- Struttura logica dei dispositivi avanzati basati su PLA a PAL con **retroazione e bistabili** (per FSM)





PLA: *Esempio FSM con rete multi livello*

- Realizzazione multi livello delle macchina a stati identificata dalle espressioni logiche sotto riportate (a, b ingressi primari, Y uscita della FSM, s_1, s_2 bit di stato):

$$Y = s_1 !ab + s_2 b + !s_2 \quad (\text{uscita } \lambda)$$

$$s_1^* = s_1 !a + s_2 + s_1 b \quad (\text{funzione di transizione } \delta_1)$$

$$s_2^* = s_1 !a !b + s_2 !b + !s_1 \quad (\text{funzione di transizione } \delta_2)$$

- Si raccoglie il fattore $g = s_1 !a + s_2$ ottenendo

$$Y = s_1 !ab + s_2 b + !s_2 = b(s_1 !a + s_2) + !s_2 = \mathbf{bg} + !s_2$$

$$s_1^* = s_1 !a + s_2 + s_1 b = (s_1 !a + s_2) + s_1 b = \mathbf{g} + s_1 b$$

$$s_2^* = s_1 !a !b + s_2 !b + !s_1 = !b(s_1 !a + s_2) + !s_1 = \mathbf{!bg} + !s_1$$

- Tale forma può essere realizzata grazie alla retroazione disponibile e agli elementi di memoria presenti.



PLA: *Esempio FSM con rete multi livello*

- Le funzioni da realizzare, di transizione e d'uscita, sono:

$$g = s_1!a + s_2$$

$$Y = bg + !s_2$$

$$s_1^* = g + s_1b$$

$$s_2^* = !bg + !s_1$$

- La prima funzione è utilizzata in retroazione per realizzare le rimanenti
- La rete è a più livelli ed utilizza elementi di memoria FFD.
 - L'uso di FFD rende le funzioni di transizione uguali alle funzioni di eccitazione. *Stato presente*: $Q_1 = s_1$; $Q_2 = s_2$ *Stato futuro*: $D_1 = s_1^*$; $D_2 = s_2^*$;

- Le funzioni da realizzare, di eccitazione e d'uscita, sono:

$$g = Q_1!a + Q_2$$

$$Y = bg + !Q_2$$

$$D_1 = g + Q_1b$$

$$D_2 = !bg + !Q_1$$



PLA: Esempio FSM con rete multi livello

- Lo schema logico è:

Sezione AND

$$p1 = Q_1 !a$$

$$p2 = Q_2$$

$$p3 = bg$$

$$p4 = !Q_2$$

$$p5 = g$$

$$p6 = Q_1 b$$

$$p7 = !bg$$

$$p8 = !Q_1$$

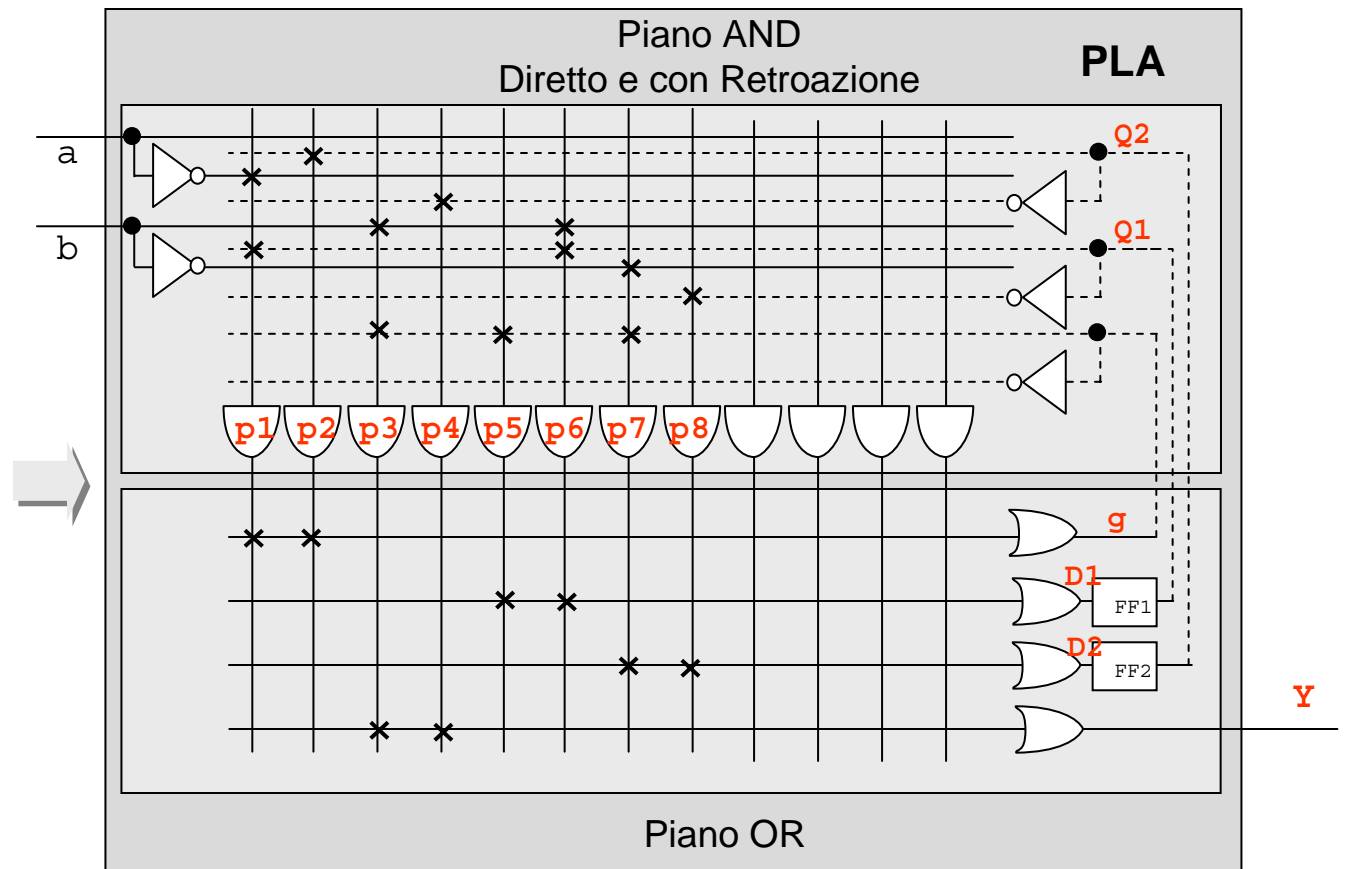
Sezione OR

$$g = Q_1 !a + Q_2$$

$$Y = bg + !Q_2$$

$$D_1 = g + Q_1 b$$

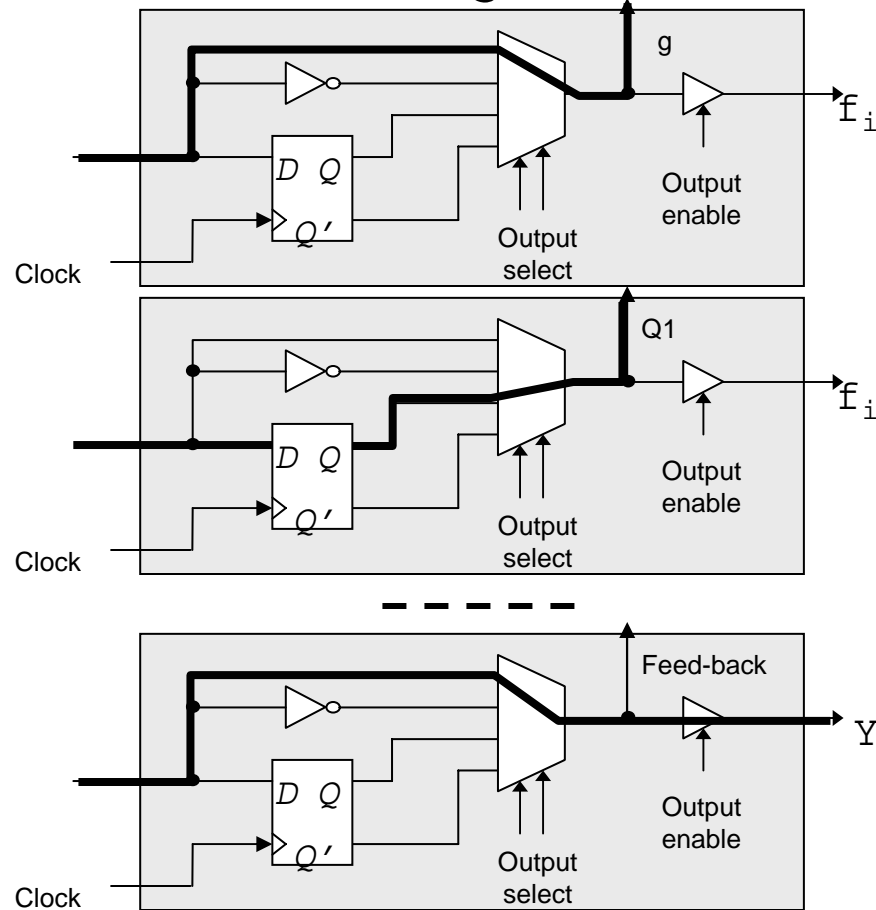
$$D_2 = !bg + !Q_1$$





PLA: *Esempio FSM con rete multi livello*

- Dove le celle di uscita sono configurate nel modo seguente





Complex Programmable Logic Devices

- I Dispositivi Logici Programmabili Complessi (*Complex Programmable Logic Devices - CPLD*) sono una evoluzione delle PLA e PAL

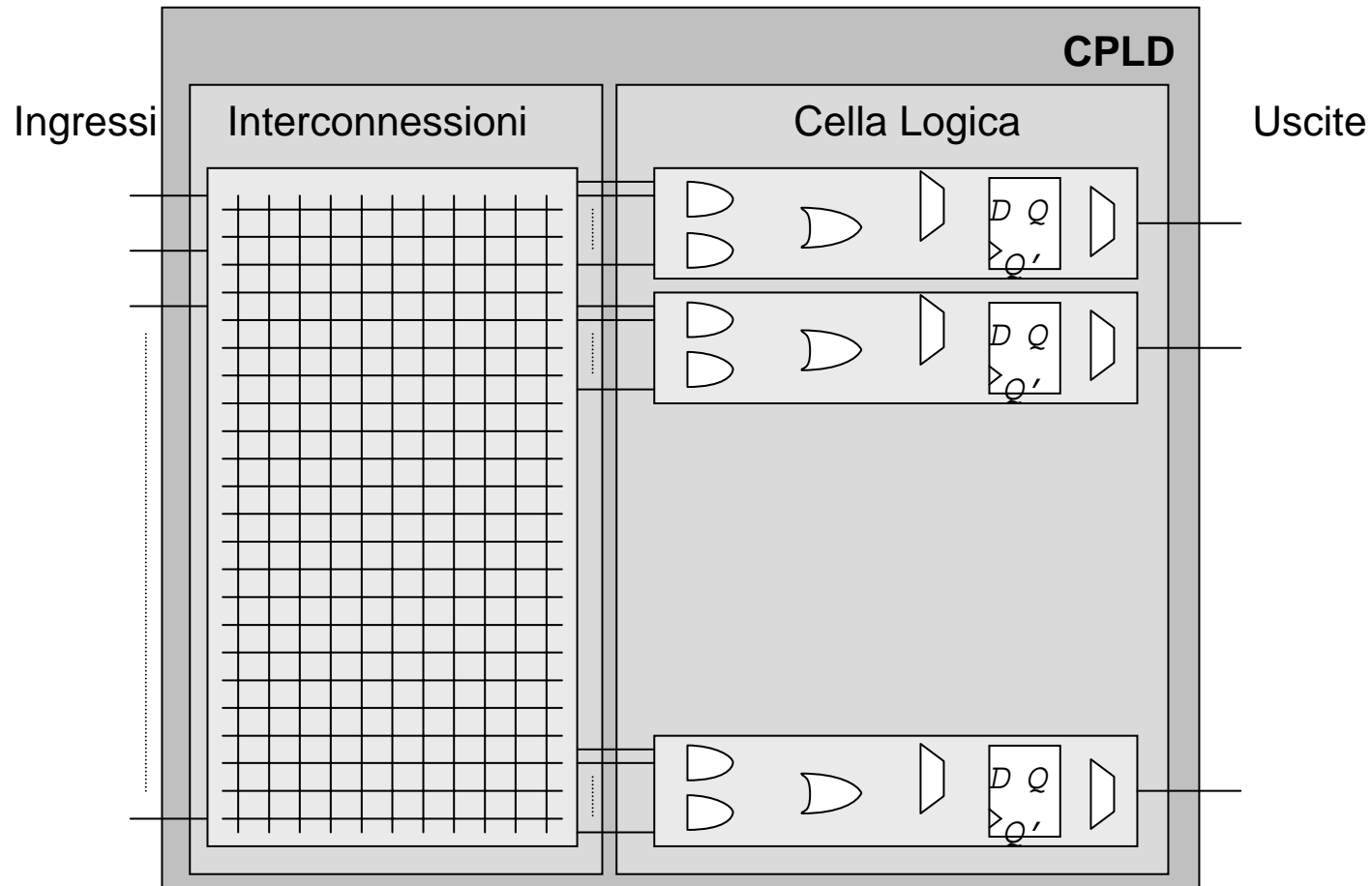
- Sono caratterizzati da:
 - Connessioni globali
 - Logica concentrata

- Rispetto a PAL e PLA:
 - Hanno dimensioni molto maggiori.
 - Sono costituite da celle più complesse.
 - Consentono di ottenere prestazioni più elevate.
 - Hanno una struttura più regolare e facilmente programmabile.



Complex Programmable Logic Devices

- L'architettura generale è la seguente





Complex Programmable Logic Devices

- Esempio: Cella logica della *Xilinx XC9500*

