



CEFRIEL

Consorzio per la Formazione e la Ricerca
in Ingegneria dell'Informazione

Politecnico
di Milano



Tecnologie Hardware

Dispositivi programmabili

Introduzione

Programmable Logic Array

Programmable Array Logic

Memorie ROM

Altri dispositivi programmabili



- Sono dispositivi hardware che mettono a disposizione
 - ▶ Componenti logici
 - Porte logiche
 - Flip-flop
 - Buffer
 - ▶ Linee di connessione
- Si dicono programmabili in quanto i componenti disponibili possono essere connessi a seconda delle esigenze di progetto
- Esistono molte tipologie di dispositivi programmabili
 - ▶ PAL, PLA, ROM, GAL
 - ▶ CPLD
 - ▶ FPGA



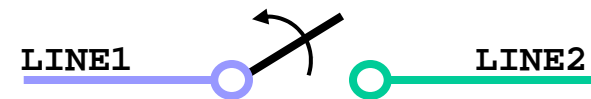
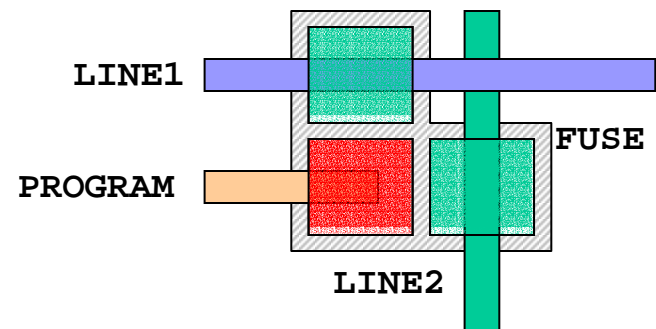
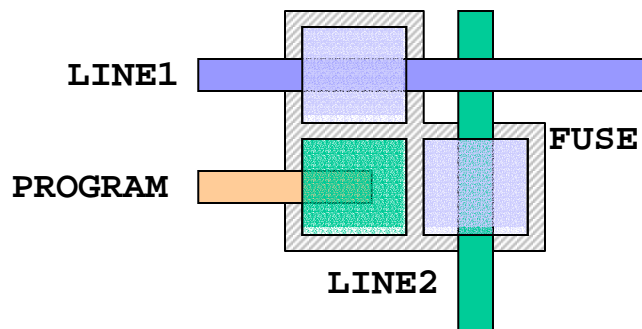
- I diversi tipi di dispositivi possono essere classificati secondo diversi criteri
- Programmabilità
 - ▶ One-Time Programmable, OTP
 - Fuse, Antifuse
 - ▶ Reprogrammable
 - E²PROM, SRAM
 - ▶ Reconfigurable
 - SRAM
- Connessioni
 - ▶ Globali
 - ▶ Locali e distribuite



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

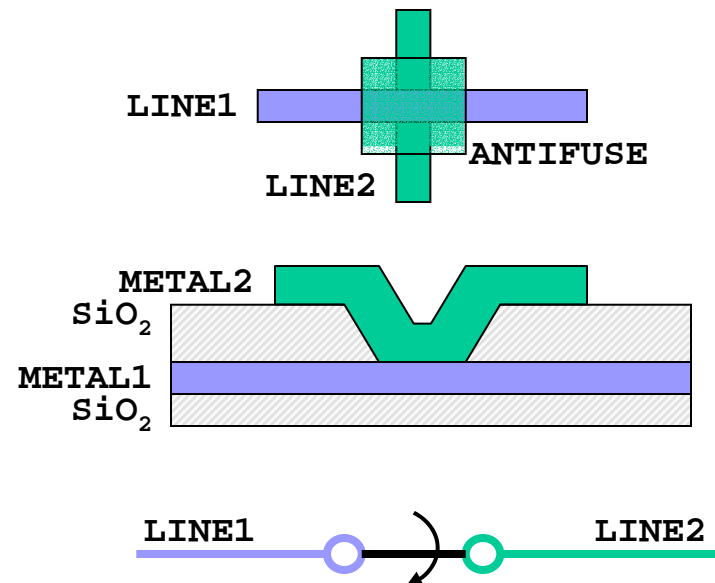
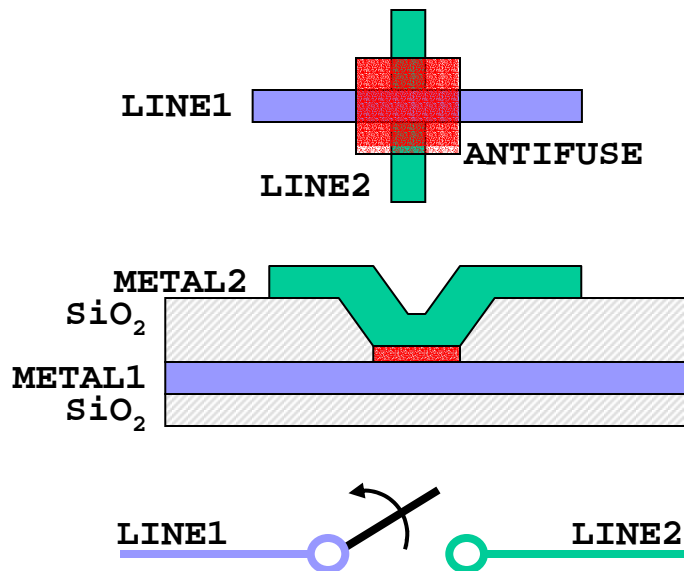




Antifuse (OTP)



- 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

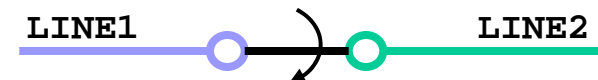
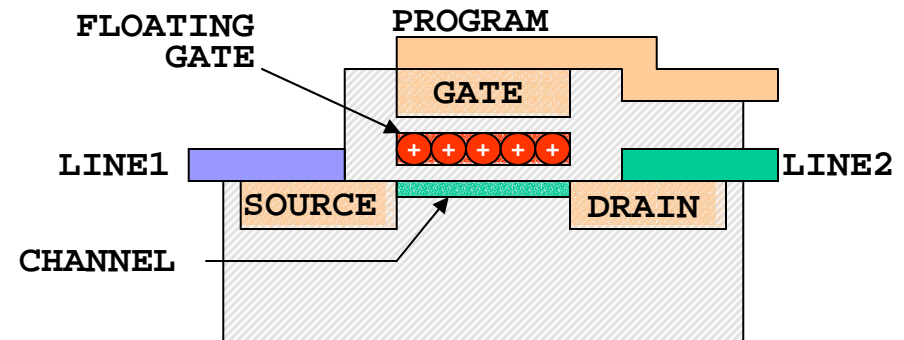
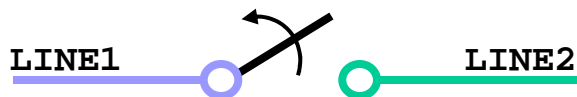
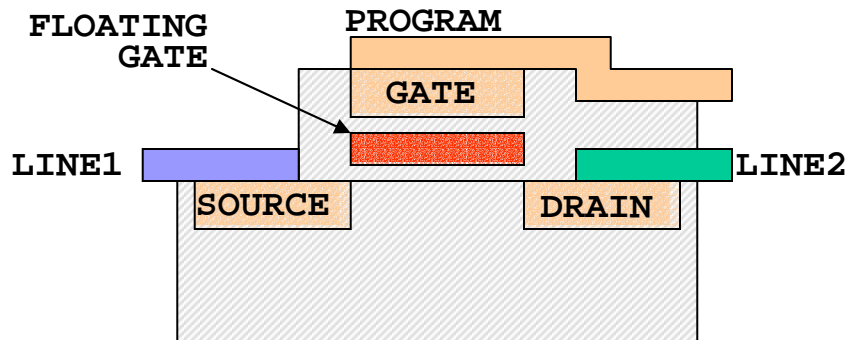




E²PROM (Reprogrammable)



- 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

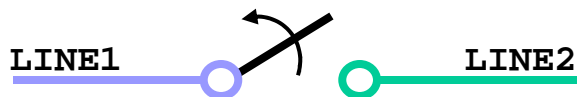
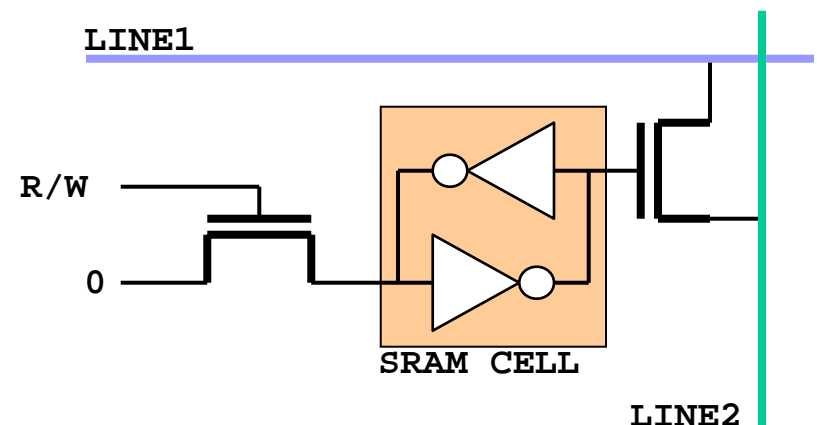
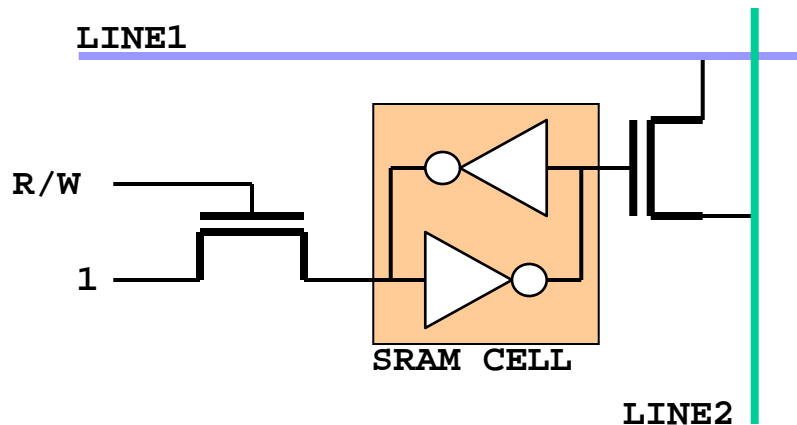




SRAM (Reprogrammable)



- 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





Connessioni globali



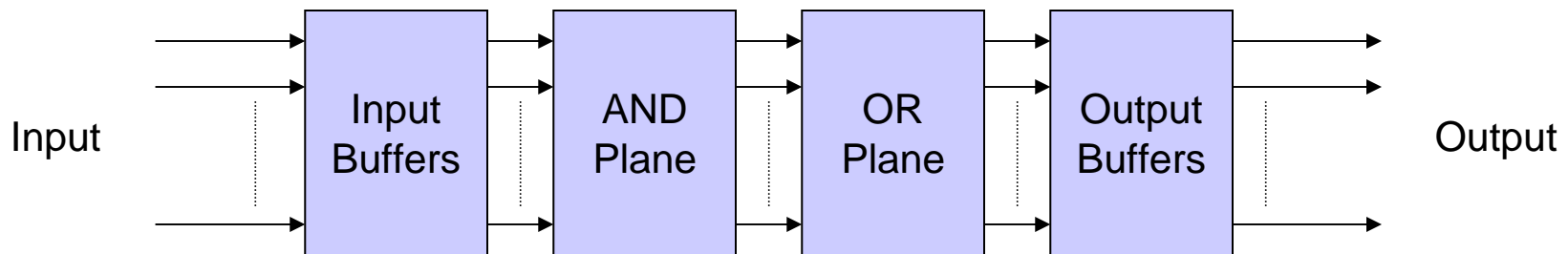
- Alcuni tipi di dispositivi dispongono di linee di connessione globali
 - ▶ La lunghezza delle linee è maggiore
 - I ritardi sono maggiori
 - ▶ Ogni linea è condivisa da molti elementi logici
 - Può essere usata come uscita di uno solo di essi
 - La flessibilità è quindi limitata
 - ▶ L'architettura è molto semplice
- Dispositivi di questo tipo sono:
 - ▶ Programmable Logic Array (PAL)
 - ▶ Programmable Array Logic (PLA)
 - ▶ Generic Array Logic (GAL)
 - ▶ Alcune versioni estese delle precedenti



Logiche programmabili a 2 livelli



- Sono usate per realizzare funzioni a due livelli
- Dispongono di
 - ▶ Un numero di ingressi e di uscite fissato
 - ▶ Un piano AND, per la costruzione dei mintermini
 - ▶ Un piano OR, per la somma dei mintermini
- Per ragioni elettriche, dispongono inoltre di
 - ▶ Buffer di ingresso
 - ▶ Buffer di uscita





Logiche programmabili a 2 livelli



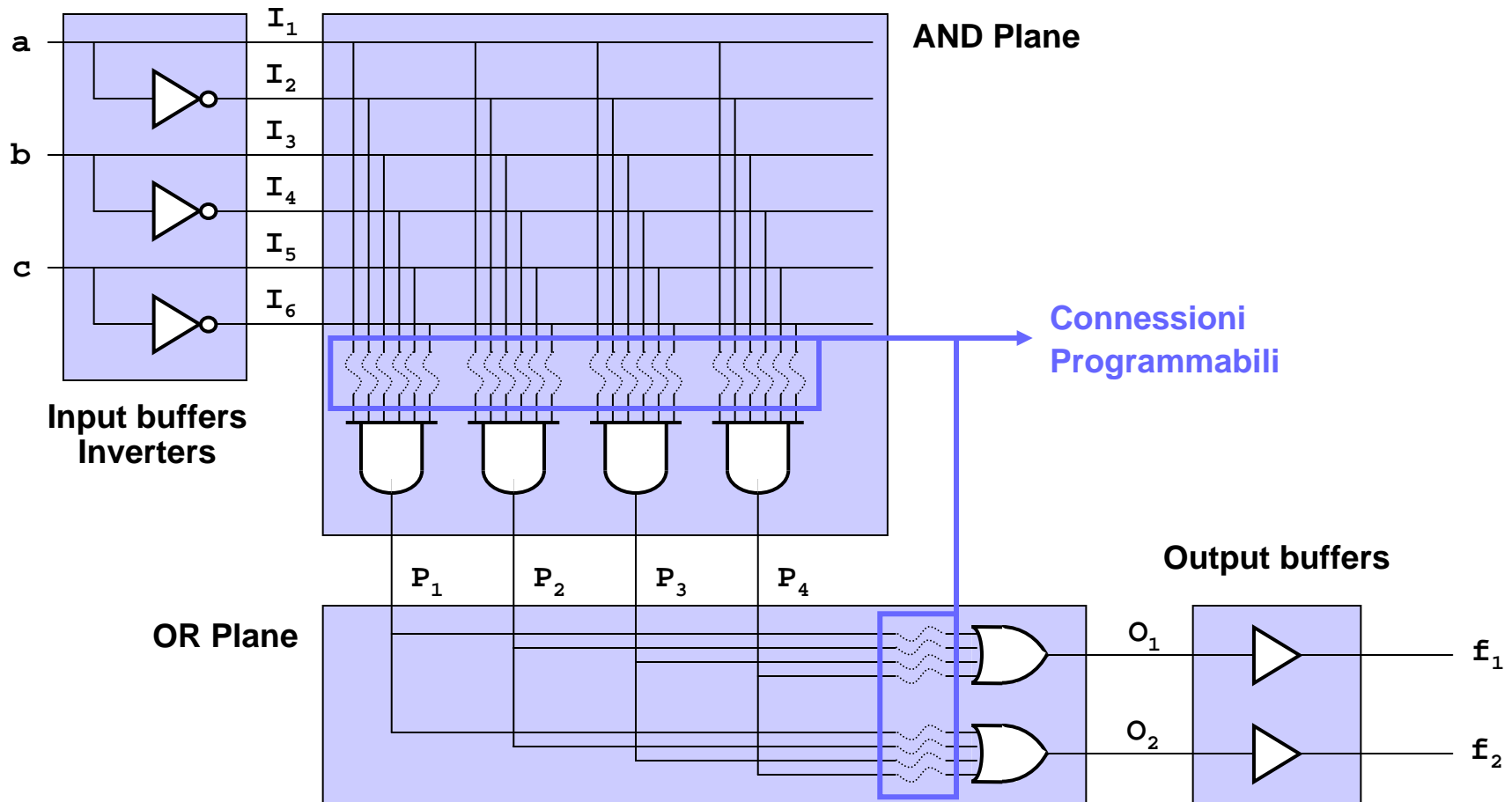
- Si possono distinguere tre tipi principali:
- Programmable Logic Array o PLA
 - ▶ Piani AND e OR programmabili
 - ▶ Si costruiscono solo i mintermini necessari
- Programmable Array Logic o PAL
 - ▶ Solo il piano AND programmabile
 - ▶ Si costruiscono solo i mintermini necessari
- Read-Only Memory o ROM
 - ▶ Piano AND pre-programmato mediante un decoder
 - ▶ Sono disponibili tutti i mintermini



Programmable Logic Array



- Schema completo

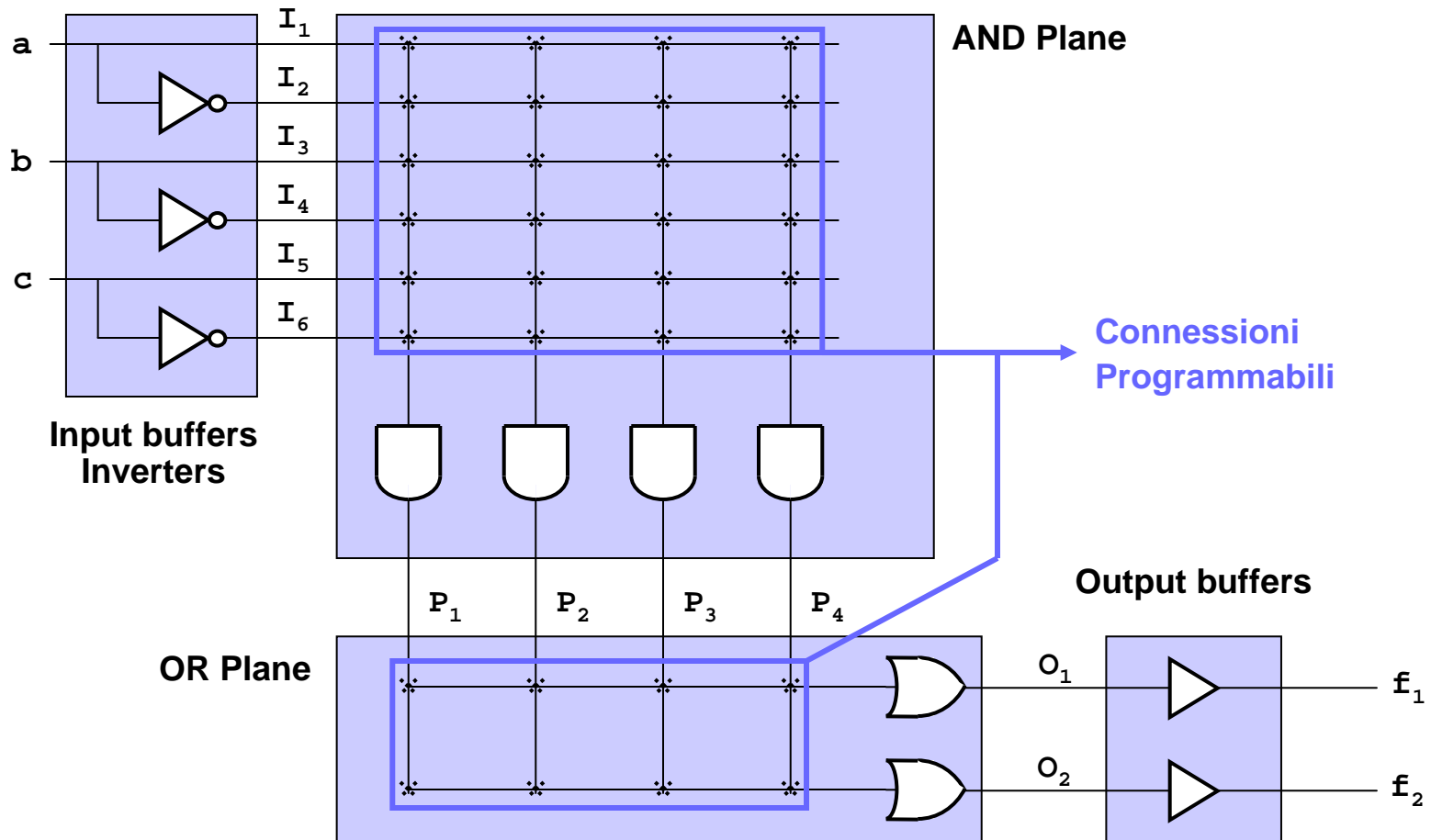




Programmable Logic Array



- Schema semplificato





- Realizzazione delle funzioni:

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

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



Formato PLA:

11- 10

1-0 10

001 10

11- 01

1-1 01

001 01

- Prodotti:

$$P1 = ab$$

$$P2 = ac$$

$$P3 = ac'$$

$$P4 = a'b'c$$

- Somme:

$$O1 = P1 + P3 + P4$$

$$O2 = P1 + P2 + P4$$

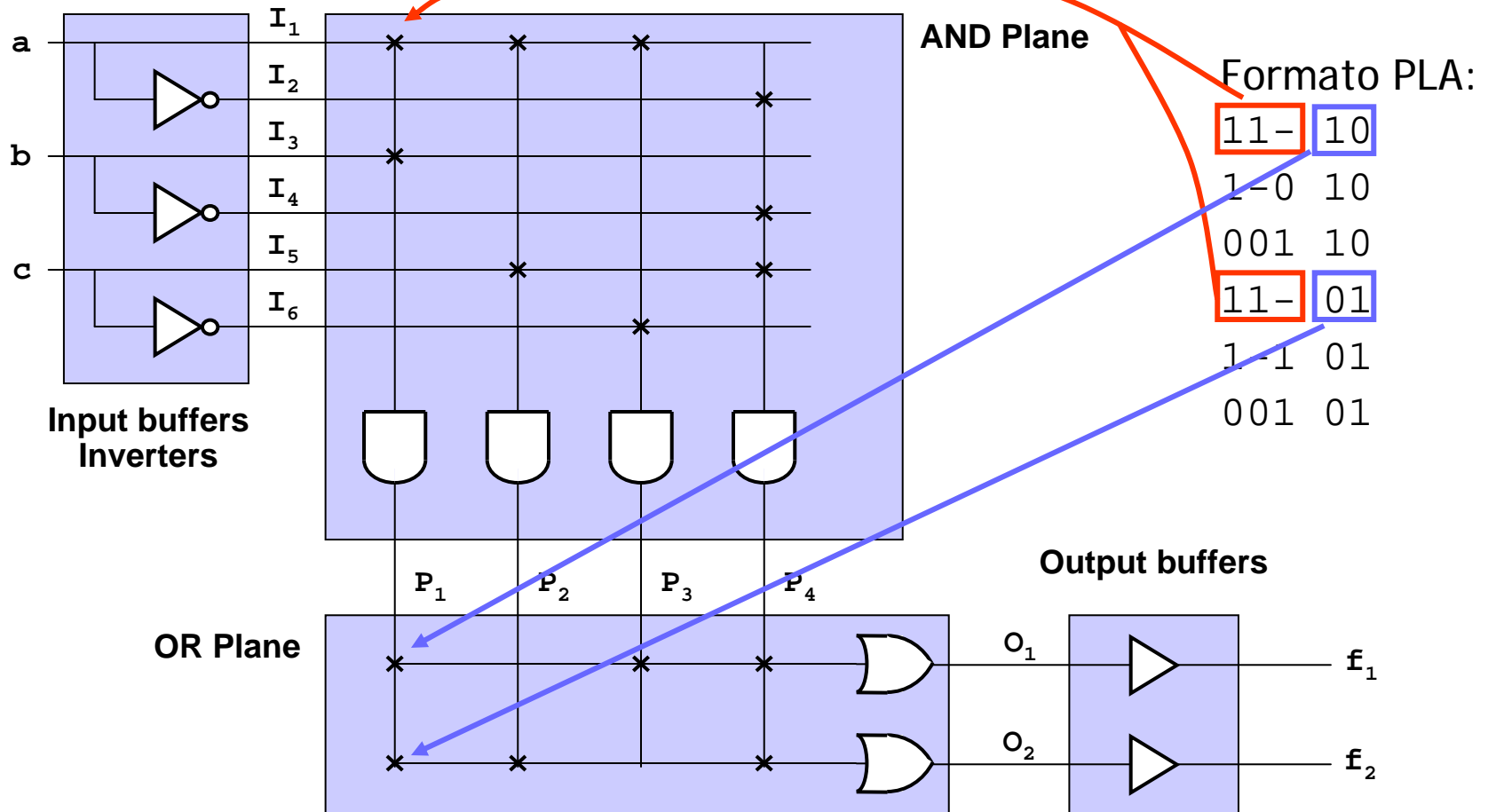


Esempio

[2]



- PLA programmata per le funzioni f_1 ed f_2





Programmable Logic Array (PLA)



- 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



Programmable Logic Array (PLA)



- Esempio (cont.):

Formato PLA:

1-- 10000

-11 10000

-00 10000

1-1 01000

10- 00100

0-1 00100

1-0 00100

-10 00100

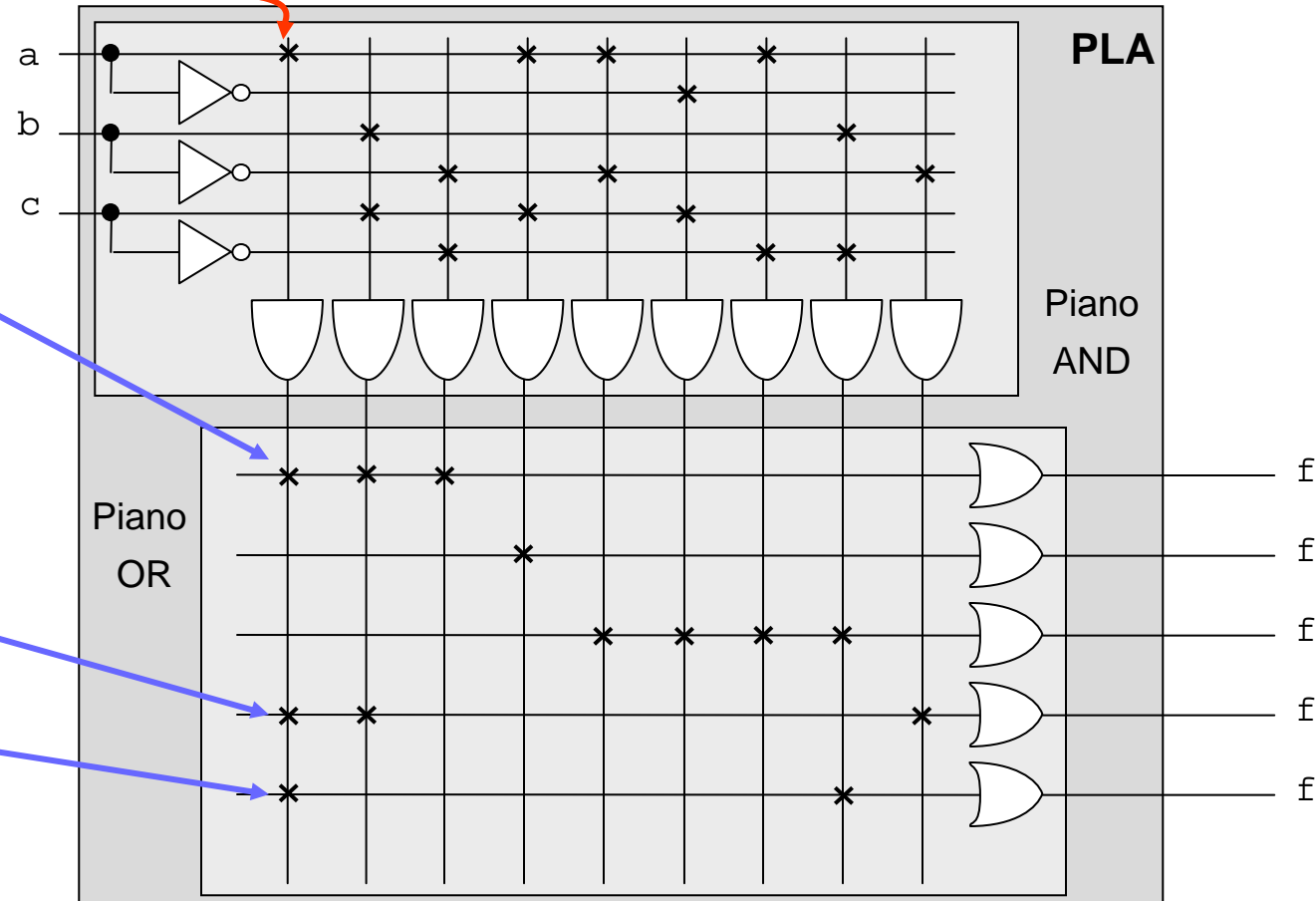
1-- 00010

-0- 00010

-11 00010

1-- 00001

-10 00001

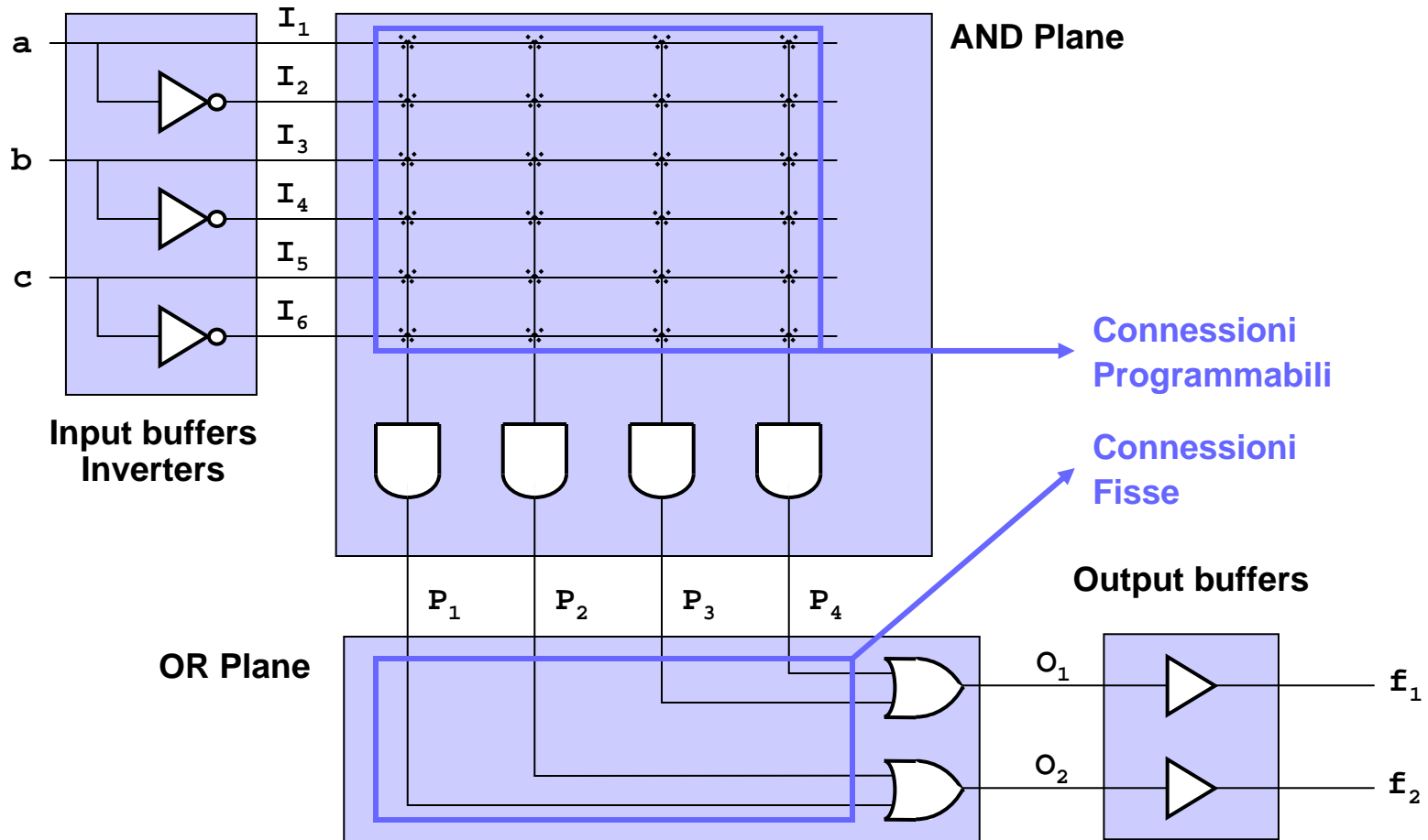




Programmable Array Logic



- Schema semplificato





Esempio 2



- Realizzazione delle funzioni

$$f_1 = x_1 x_2 \bar{x}_3 + \bar{x}_1 x_2 x_3$$

$$f_2 = \bar{x}_1 \bar{x}_2 + x_1 x_2 x_3$$

- Su una PAL con
 - 3 ingressi
 - 4 termini prodotto
 - 2 termini somma a 2 ingressi
 - 2 buffer di uscita



Esempio 2



- Si individuano i prodotti:

$$P_1 = x_1 x_2 \bar{x}_3$$

$$P_2 = \bar{x}_1 x_2 x_3$$

$$P_3 = \bar{x}_1 \bar{x}_2$$

$$P_4 = x_1 x_2 x_3$$

- E le somme

$$O_1 = P_1 + P_2$$

$$O_2 = P_3 + P_4$$

- Da cui le funzioni

$$f_1 = O_1$$

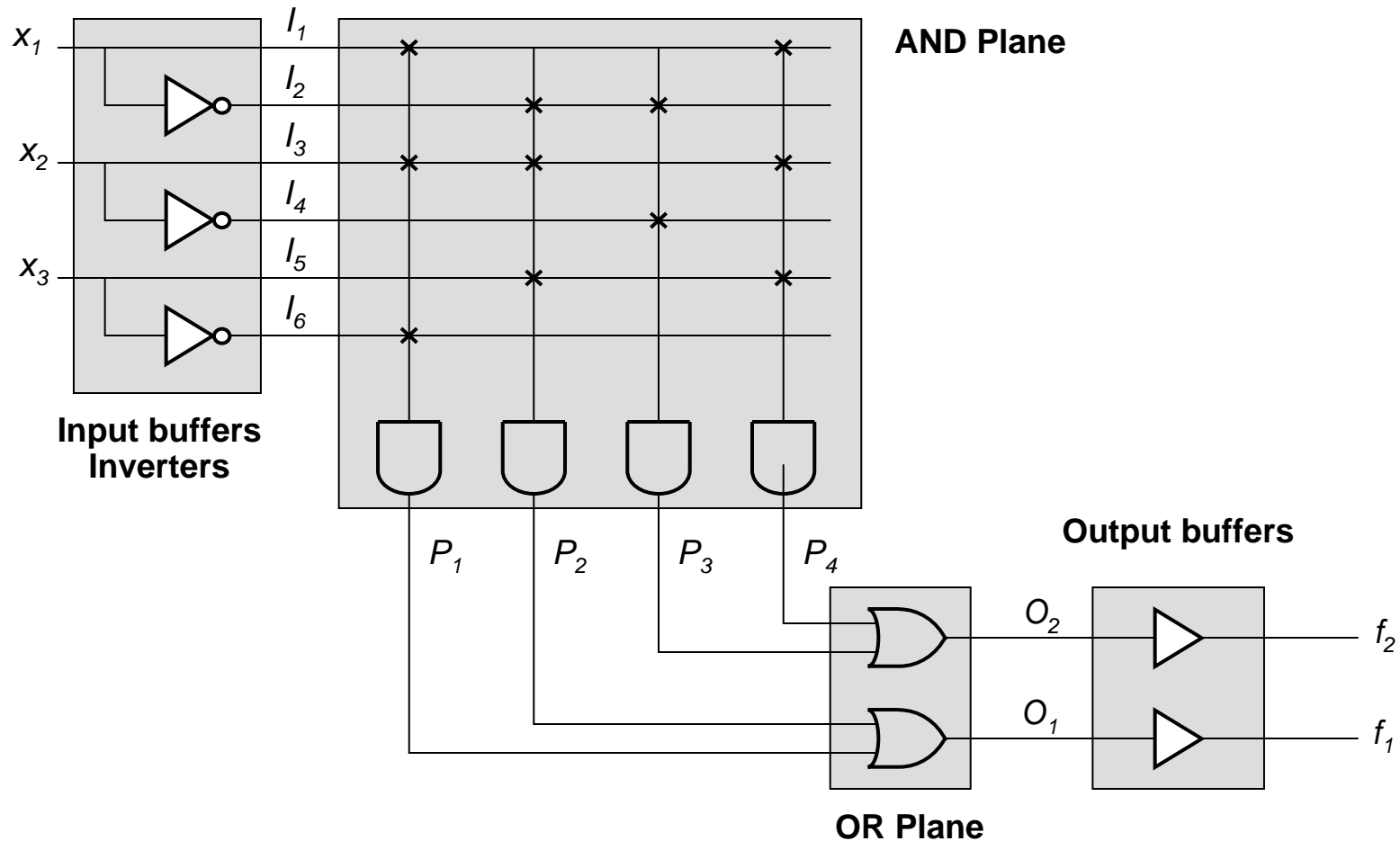
$$f_2 = O_2$$



Esempio 2



- La realizzazione che ne segue è





Read-Only Memory



- Una memoria a sola lettura o ROM associa ad ogni indirizzo (ingresso) una parola (uscita)
- In generale si ha la necessità di realizzare più funzioni

$$f_i = f_i(x_1, x_2, \dots, x_n) \quad i=\{1, 2, \dots, t\}$$

- Con una diversa notazione, si può scrivere:

$$(x_1, x_2, \dots, x_n) \Rightarrow (f_1, f_2, \dots, f_t)$$

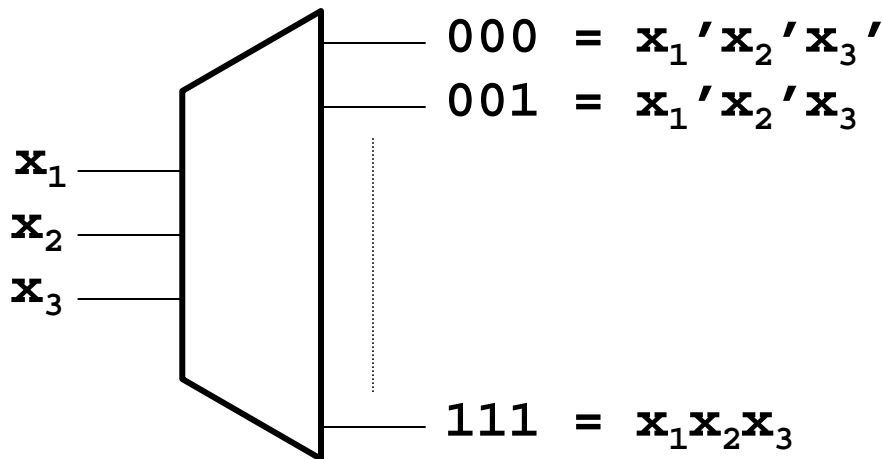
- Questa scrittura evidenzia una trasformazione che va da una n -upla di ingressi x_j ad una t -upla di uscite f_i
- Le memorie ROM, grazie all'address decoder, forniscono tutti i 2^n mintermini ottenibili dalle n variabili di ingresso x_i



Read-Only Memory



- Address decoder
 - ▶ Gli ingressi sono le variabili x_i
 - ▶ Le uscite sono tutti i mintermini costruiti a partire dalle variabili di ingresso

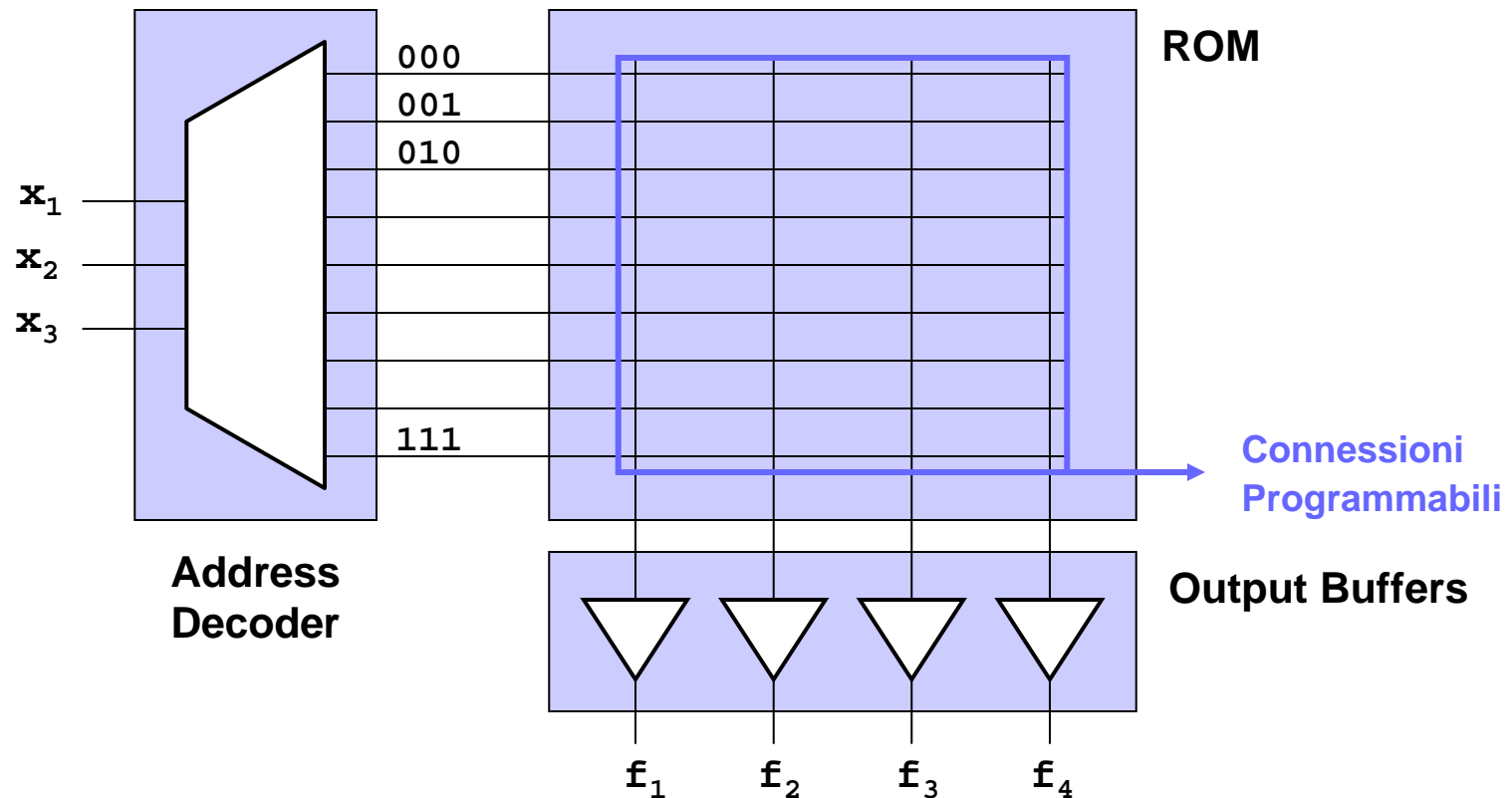




Read-Only Memory



- Read-Only Memory, struttura completa





Espressioni logiche

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

$$f_2 = abc + b'c$$

$$f_3 = a'b + b'c + ac'$$

$$f_4 = a + b' + c$$

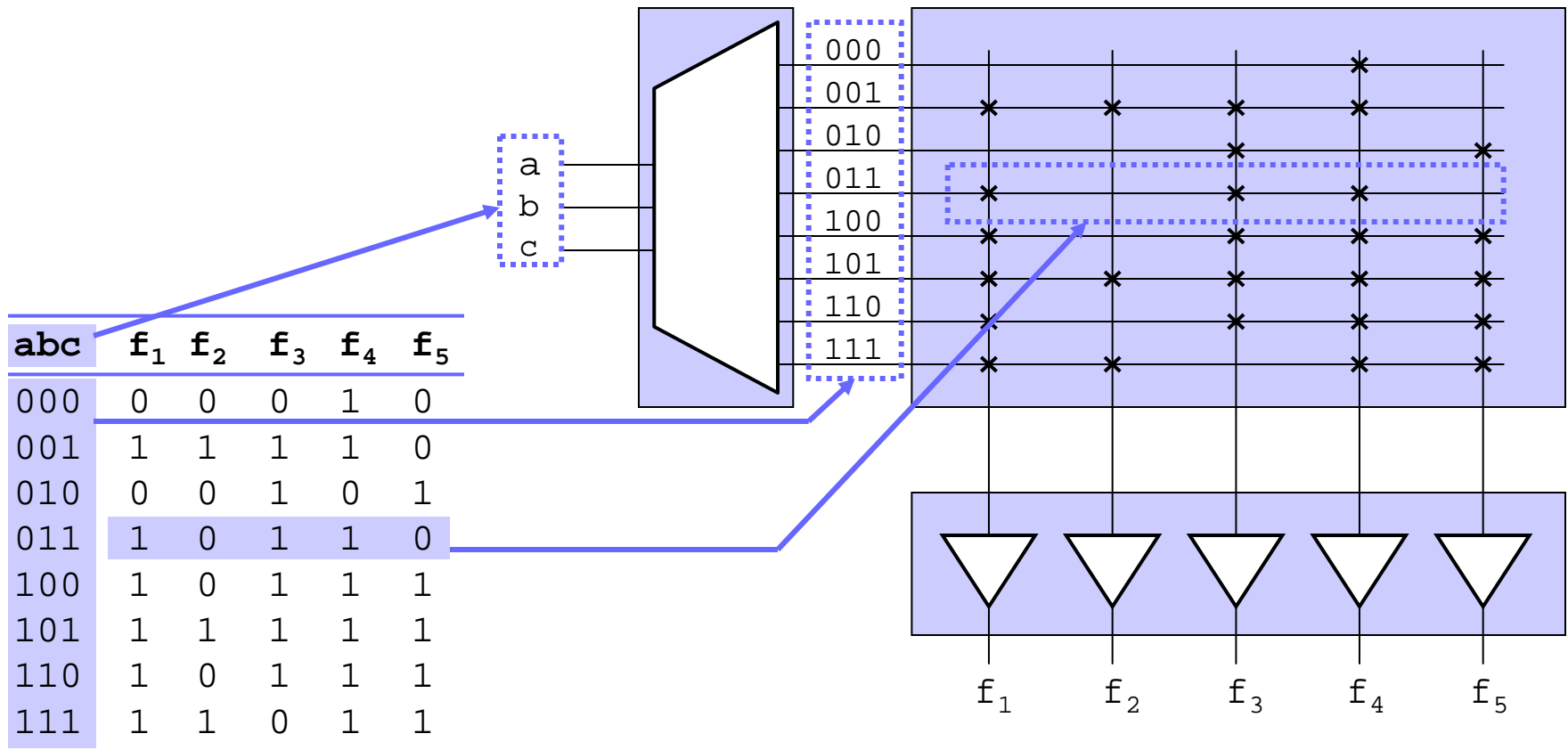
$$f_5 = a + bc'$$

Tabella della verità

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

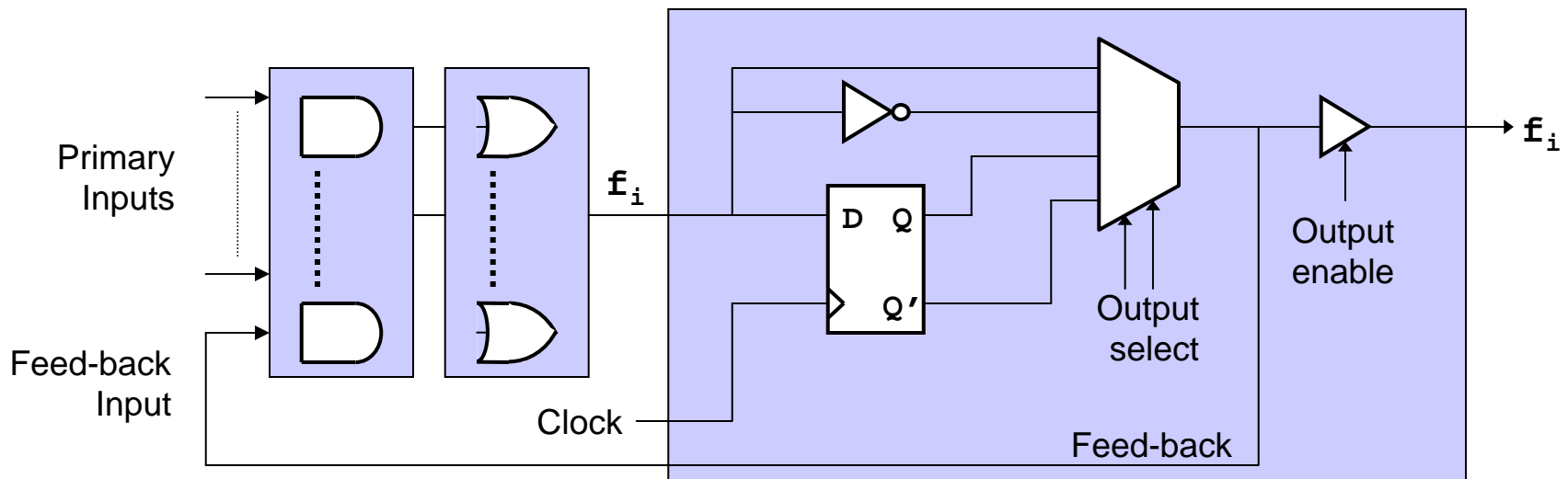


- Realizzazione delle funzioni:





- I dispositivi visti sono limitati:
 - ▶ Supportano solo funzioni a due livelli
 - ▶ Sono puramente combinatori
- Nuovi dispositivi basati su PLA:
 - ▶ Introducono una rete di retroazione
 - ▶ Dispongono di elementi sequenziali





- Realizzazione delle funzioni:

$$f_1 = ab'd + cd + c'$$

$$f_2 = ab' + c + ad$$

$$f_3 = ab'd' + acd' + a'$$

- Si raccoglie il fattore

$$g = ab' + c$$

- Ottenedo

$$f_1 = ab'd + cd + c' = d(ab' + c) + c' = dg + c'$$

$$f_2 = ab' + c + ad = (ab' + c) + ad = g + ad$$

$$f_3 = ab'd' + acd' + a' = ad'(ab' + c) + a' = ad'g + a'$$

- Tale forma può essere facilmente realizzata grazie alla retroazione disponibile



- Le funzioni da realizzare sono dunque

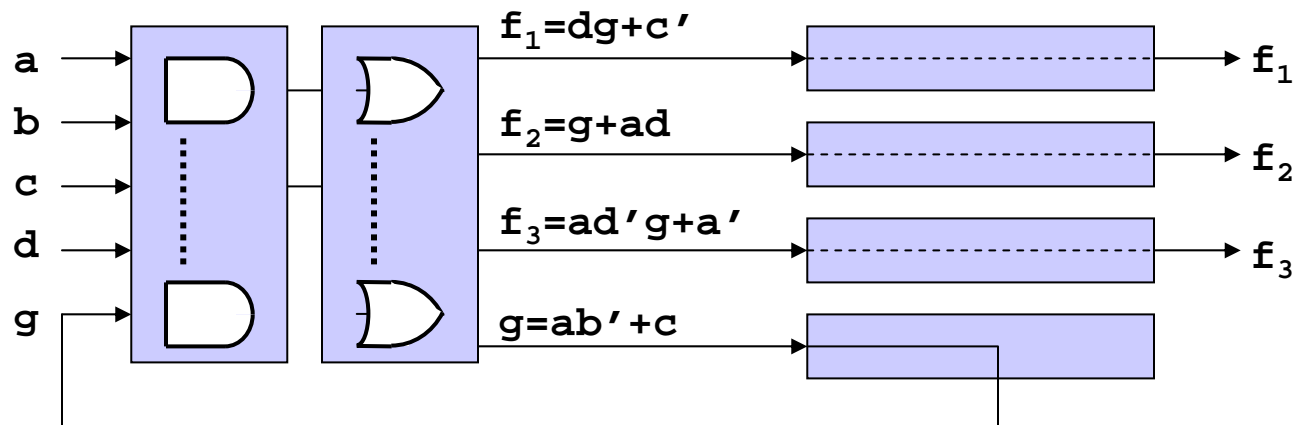
$$g = ab' + c$$

$$f_1 = dg + c'$$

$$f_2 = g + ad$$

$$f_3 = ad'g + a'$$

- La prima è usata in retroazione per costruire le rimanenti
- La rete non è più a due livelli

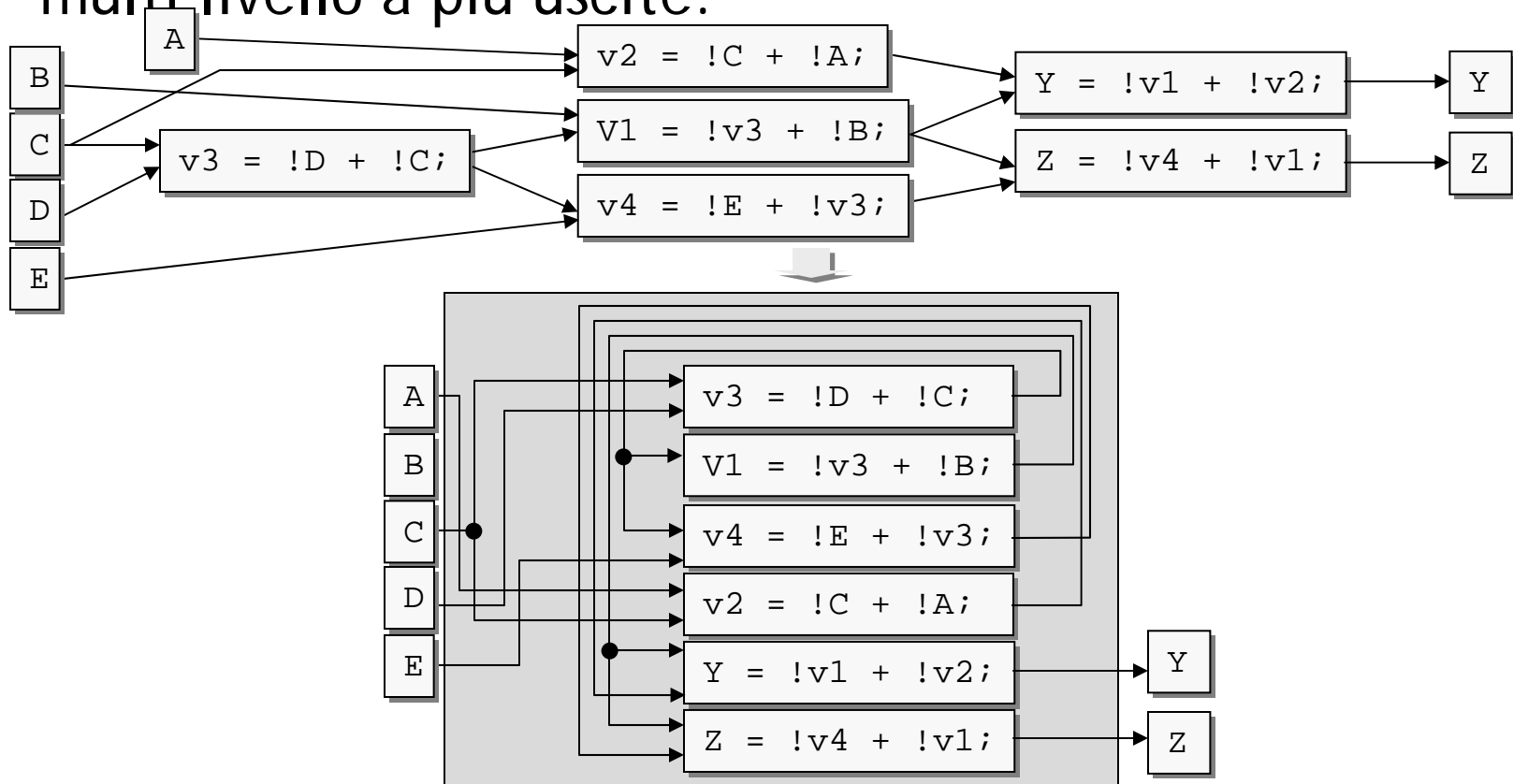




PLA e PAL Avanzate: Esempio

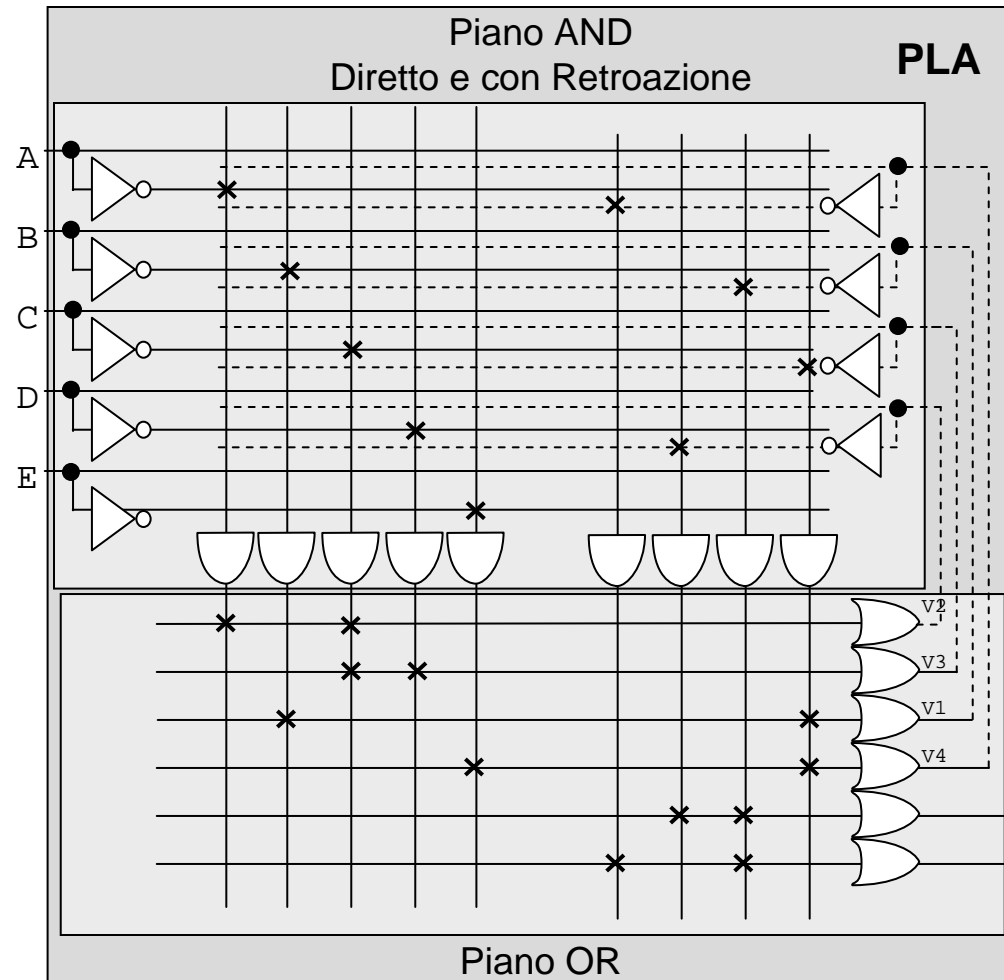
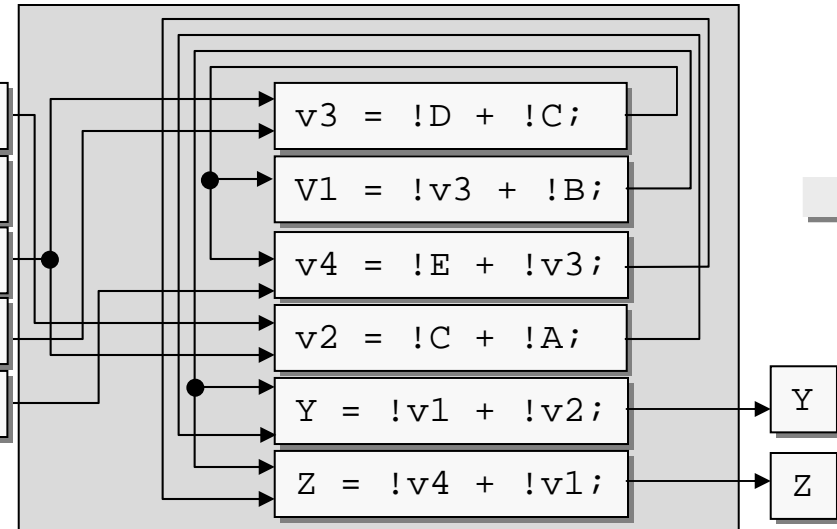


- Esempio di implementazione di una rete combinatoria multi livello a più uscite.



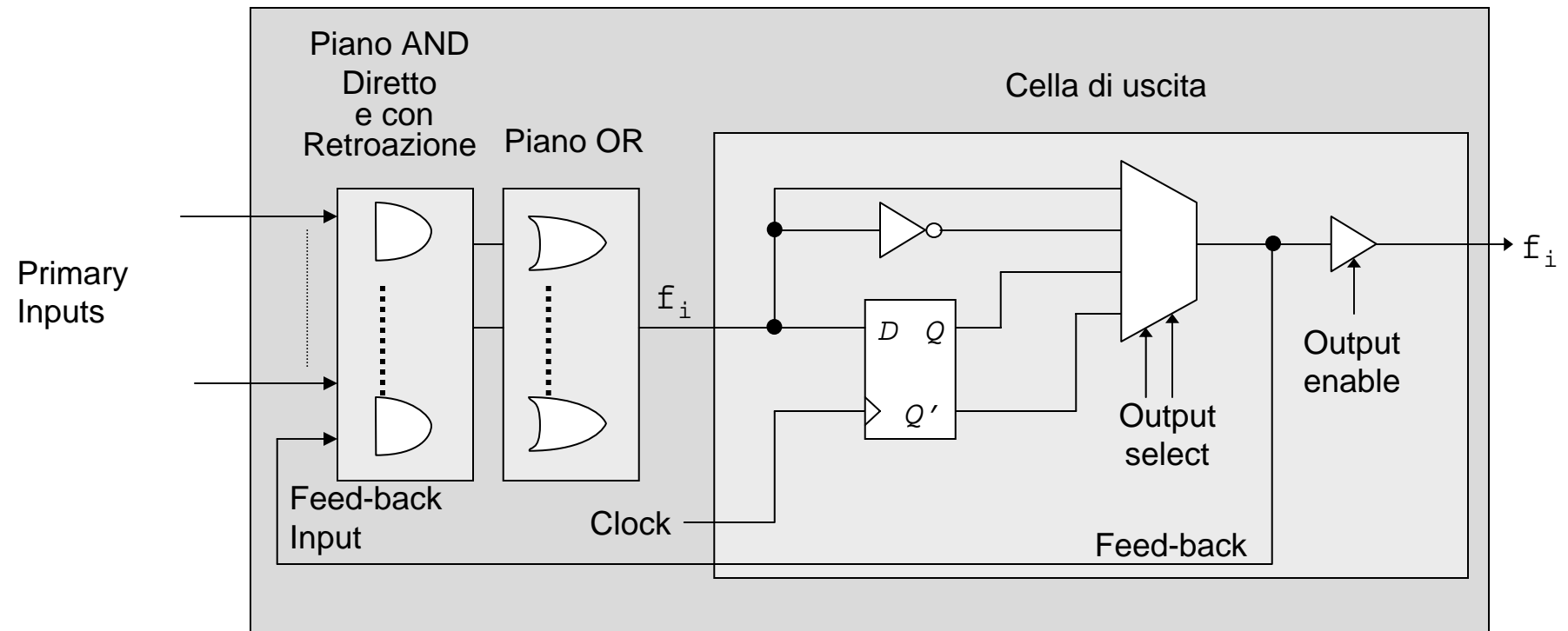


- Esempio (cont):





- Struttura logica dei dispositivi avanzati basati su PLA a PAL





- Esempio: realizzazione delle macchina a stati identificata dalle espressioni logiche seguenti dove $a; b$ sono gli ingressi primari, Y è l'uscita, ed $s_1; s_2$ sono i bit meno significativo e più significativo dello stato:

$$Y = s_1 !ab + s_2 b + !s_2$$

$$s_1^* = s_1 !a + s_2 + s_1 b$$

$$s_2^* = s_1 !a !b + s_2 !b + !s_1$$

- ▶ Si raccoglie il fattore

$$g = s_1 !a + s_2$$

- ▶ Ottenendo

$$Y = s_1 !ab + s_2 b + !s_2 = b(s_1 !a + s_2) + !s_2 = bg + !s_2$$

$$s_1^* = s_1 !a + s_2 + s_1 b = (s_1 !a + s_2) + s_1 b = g + s_1 b$$

$$s_2^* = s_1 !a !b + s_2 !b + !s_1 = !b(s_1 !a + s_2) + !s_1 = s_1 !bg + !s_1'$$

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



PLA e PAL Avanzate: *Esempio*



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

$$g = s_1!a + s_2$$

$$Y = s_1!ab + s_2b + !s_2 = b(s_1!a + s_2) + !s_2 = bg + !s_2$$

$$s_1^* = s_1!a + s_2 + s_1b = (s_1!a + s_2) + s_1d = g + s_1b$$

$$s_2^* = s_1!a!b + s_2!b + !s_1 = !b(s_1!a + s_2) + !s_1 = s_1!bg + !s_1$$

- 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:

$$g = Q_1!a + Q_2$$

$$Y = bg + !Q_2$$

$$D_1 = g + Q_1b$$

$$D_2 = Q_1!bg + !Q_1$$

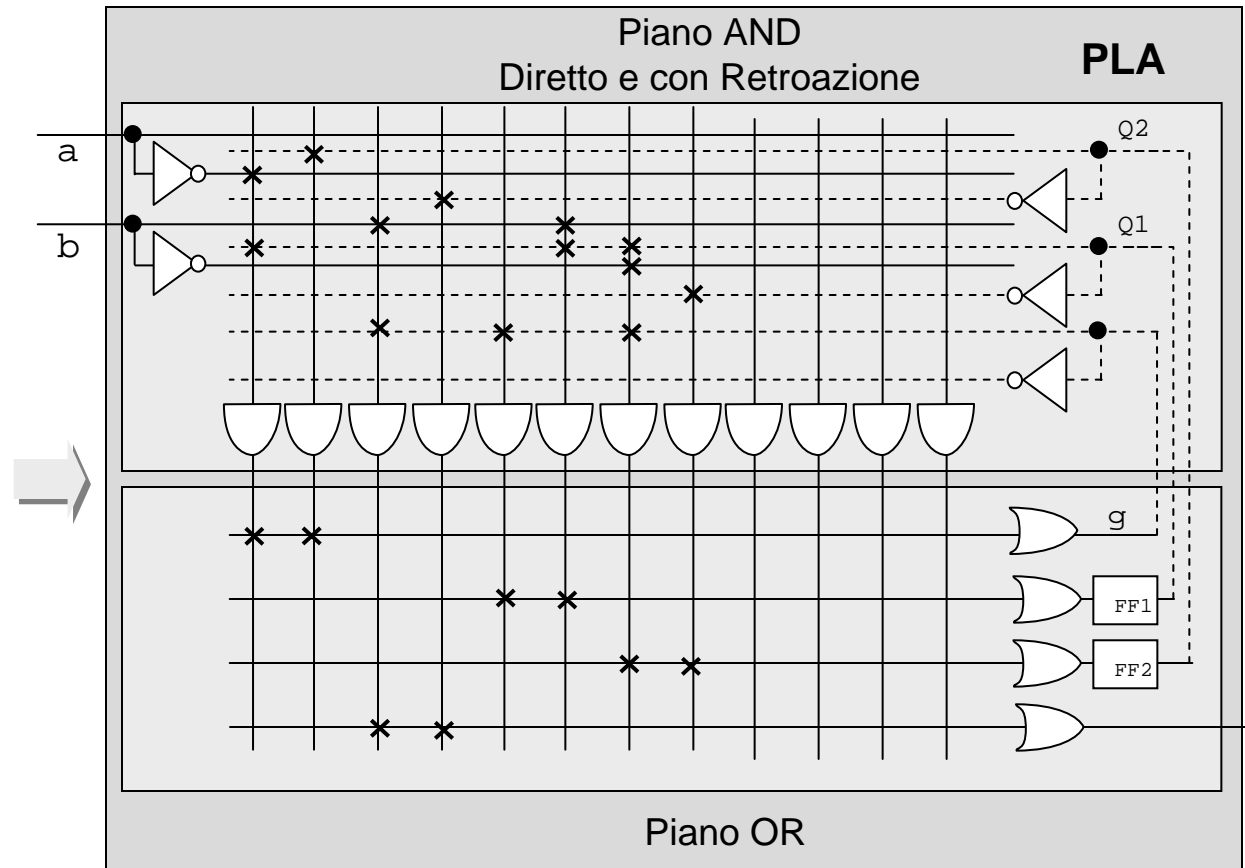


PLA e PAL Avanzate: *Esempio*



- Lo schema logico è:

$$\begin{aligned}g &= Q_1!a + Q_2 \\Y &= bg + !Q_2 \\D_1 &= g + Q_1b \\D_2 &= Q_1!bg + !Q_1\end{aligned}$$

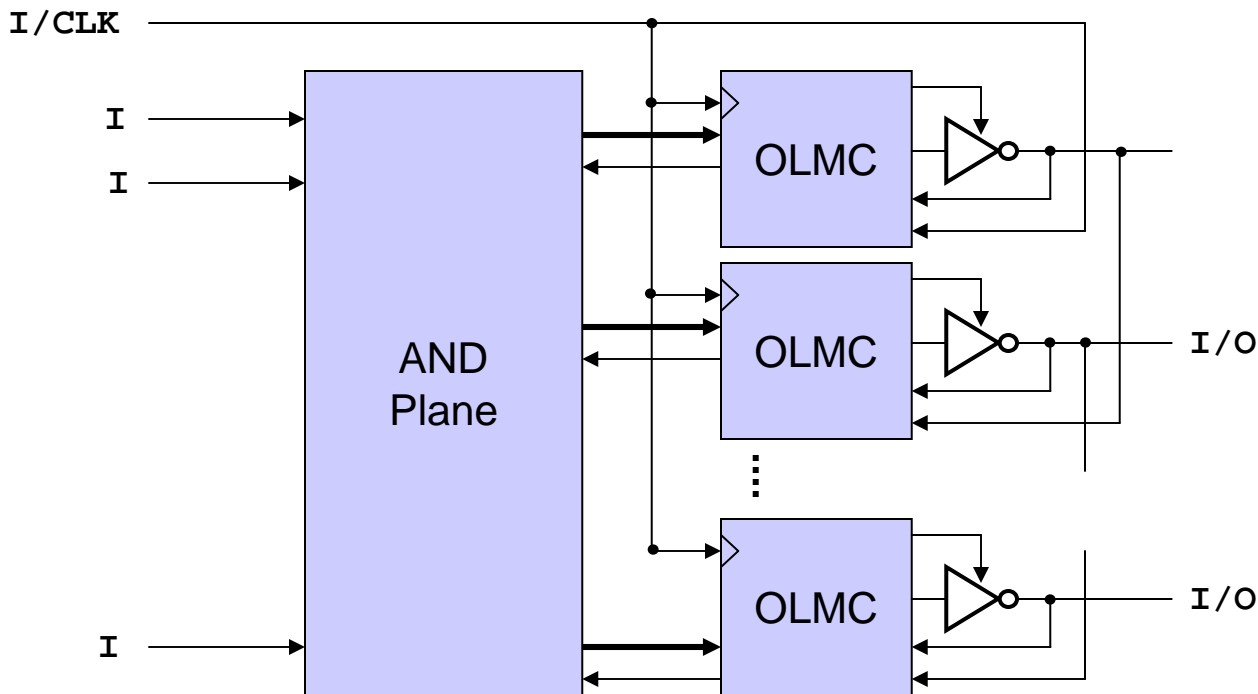




Generic Array Logic



- Le Generic Array Logic o GAL sono una generalizzazione delle logiche programmabili quali PAL e PLA
- I dispositivi più diffusi sono noti come **16V8** e **22V10**



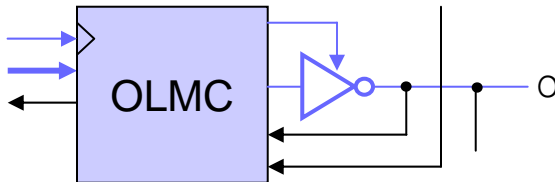


Generic Array Logic

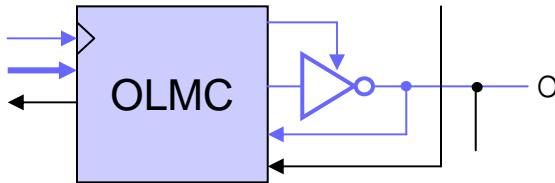


- Le celle di uscita OLMC o Output Logic MacroCell rendono questa architettura molto flessibile

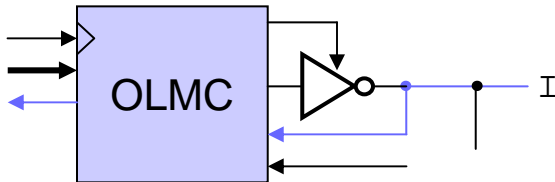
Uscita semplice



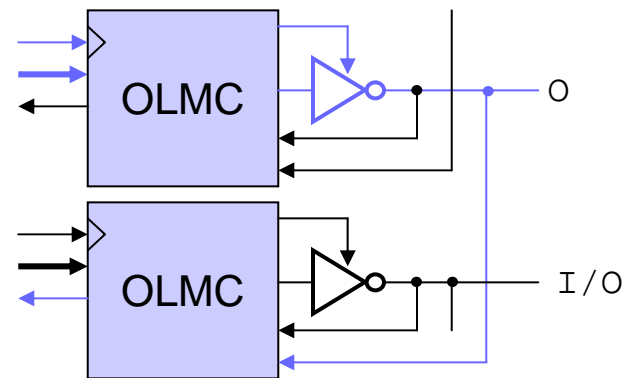
Uscita e retroazione interna



Ingresso



Uscita e retroazione esterna





Complex Programmable Logic Devices



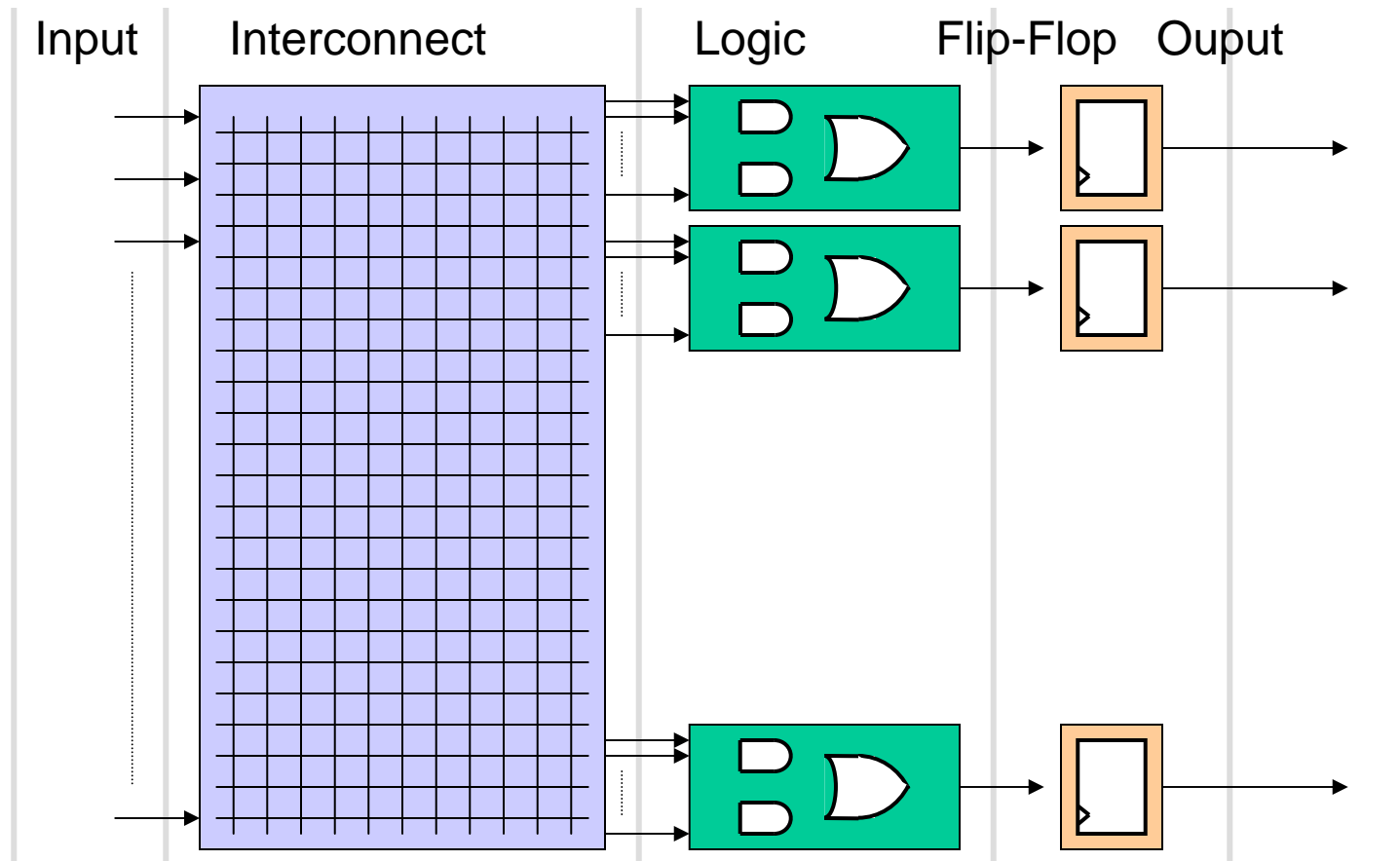
- I Complex Programmable Logic Devices o CPLD sono una evoluzione di dispositivi quali PLA, PAL e GAL
- Sono caratterizzati da
 - ▶ Connessioni globali
 - ▶ Logica concentrata
- Rispetto a PAL, PLA e GAL
 - ▶ Hanno dimensioni molto maggiori
 - ▶ Le celle disponibili sono più complesse
- Hanno diversi aspetti positivi
 - ▶ Dimensioni ridotte
 - ▶ Elevate velocità
 - ▶ Struttura regolare e facilmente programmabile



Complex Programmable Logic Devices



- L'architettura generale è la seguente

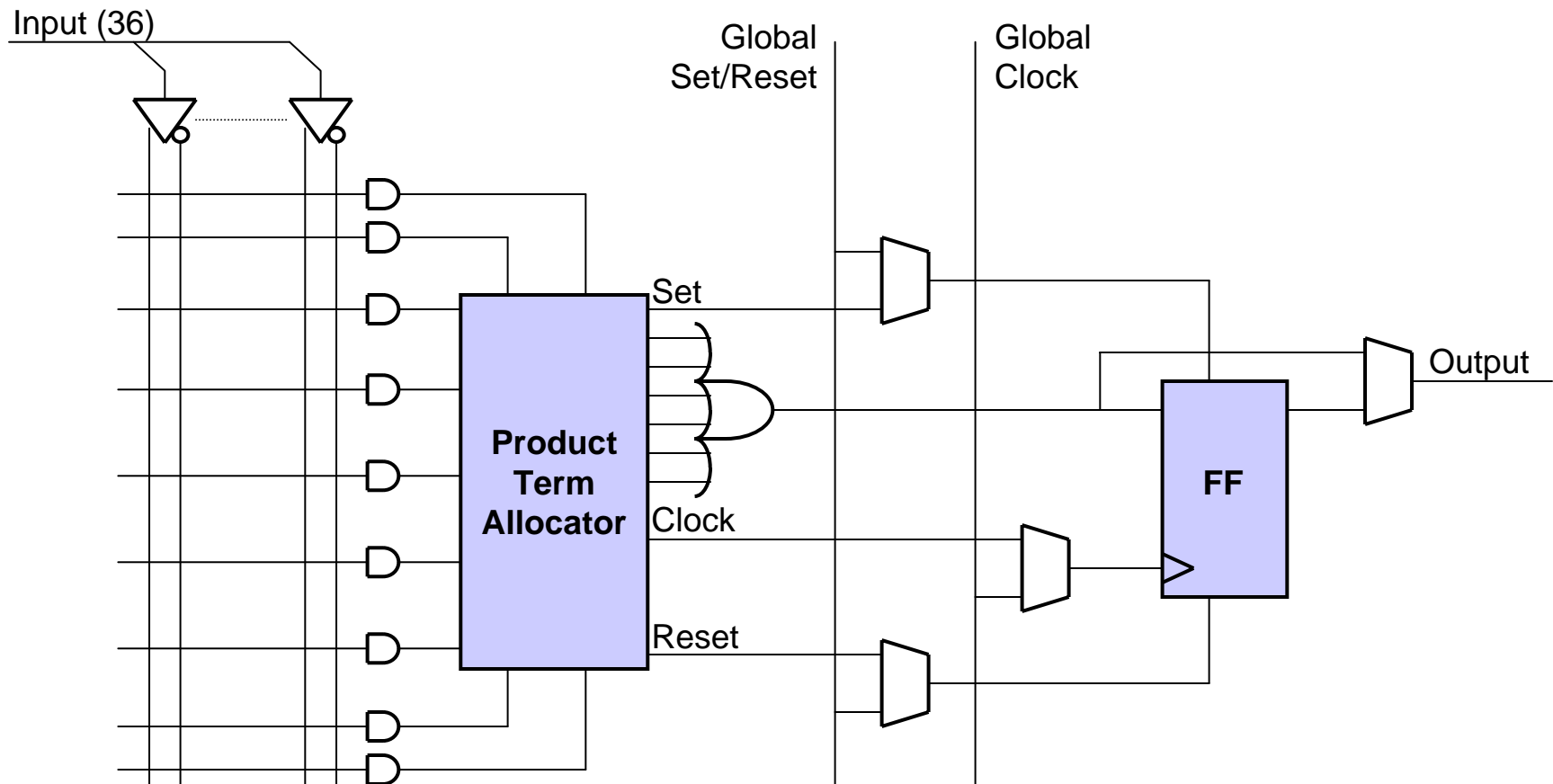




Complex Programmable Logic Devices



- Xilinx XC9500 Logic Cell





Field Programmable Gate Array



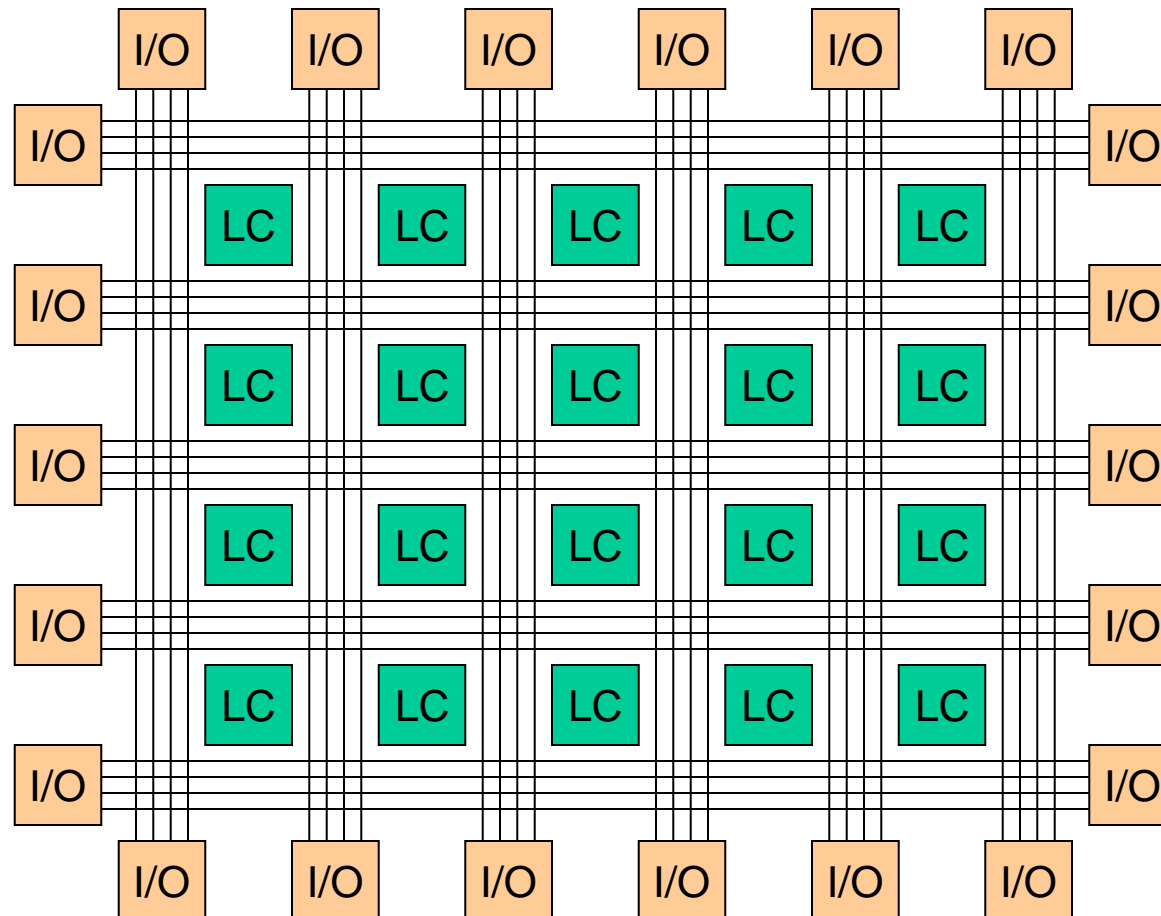
- I dispositivi noti come Field Programmable Gate Arrays o FPGA sono i dispositivi programmabili più complessi attualmente disponibili
- Sono caratterizzati da
 - ▶ Connessioni locali
 - ▶ Logica distribuita
- Rispetto a PAL, PLA, GAL e CPLD
 - ▶ Hanno dimensioni molto maggiori
 - ▶ Le celle disponibili hanno complessità variabile
 - ▶ Sono estremamente flessibili
 - ▶ Possono disporre di componenti complessi integrati



Field Programmable Gate Array



- L'architettura generale di una FPGA è la seguente





Field Programmable Gate Array

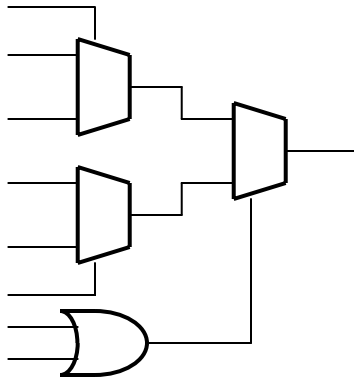


- Esistono due classi differenti di FPGA che si distinguono per la complessità delle celle o Logic Cell (LC)
 - ▶ Celle semplici
 - Flessibilità
 - Possibilità di sfruttare appieno le risorse
 - Complessità della rete di interconnessione
 - Problemi di routing
 - ▶ Celle complesse
 - Funzioni complesse in una singola cella o in poche celle
 - Spreco di risorse
 - Rete di interconnessione semplificata
 - Routing semplice

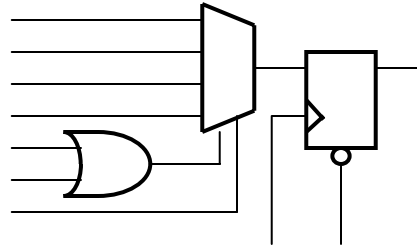


- Esempi di celle semplici: ACTEL 40MX, 42MX Family

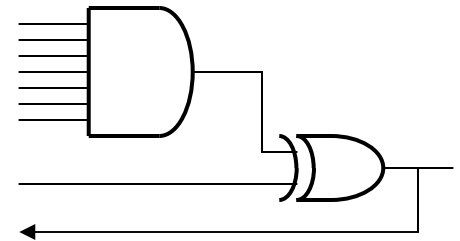
40MX C-Module



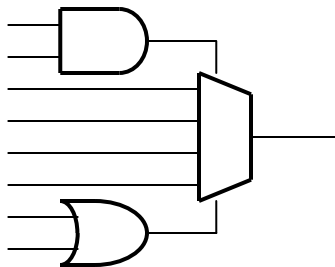
40/42MX S-Module



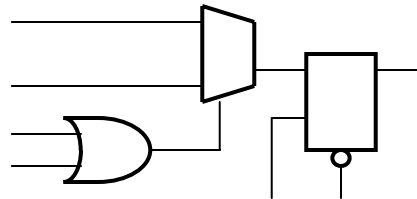
40/42MX D-Module



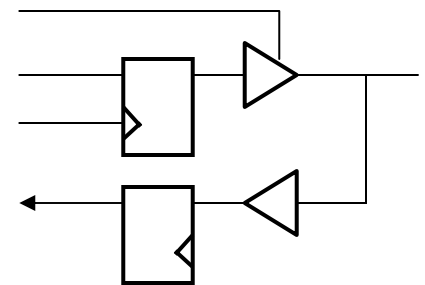
42MX C-Module



40/42MX S-Module



40/42MX I/O-Module





- Mediante multiplexer a due ingressi è possibile realizzare le funzioni logiche elementari:
 - ▶ NOT
 - ▶ AND
 - ▶ OR
- Avendo a disposizione un inverter (o utilizzando un altro multiplexer) si possono anche realizzare le funzioni:
 - ▶ NAND
 - ▶ NOR
 - ▶ XOR
- Inoltre il multiplexer può essere utilizzato come tale

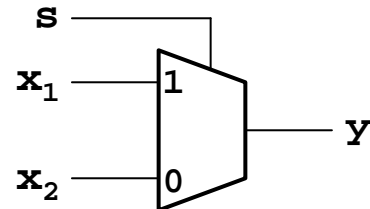


Celle semplici

[3]



$$y = x_1 s + x_2 s'$$

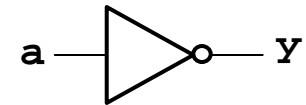
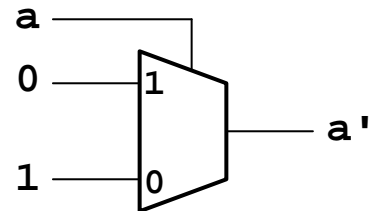


$$s = a$$

$$x_1 = 0$$

$$x_2 = 1$$

$$y = (0)a + (1)a' = a'$$

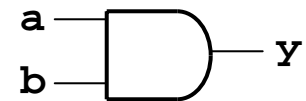
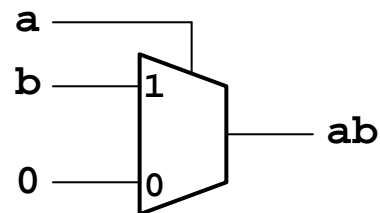


$$s = a$$

$$x_1 = b$$

$$x_2 = 0$$

$$y = ba + (0)a' = ab$$

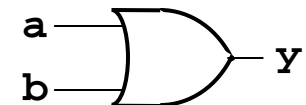
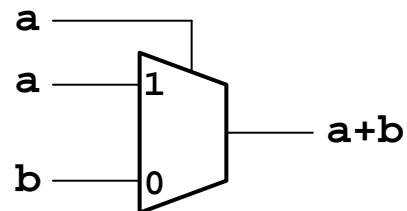


$$s = a$$

$$x_1 = a$$

$$x_2 = b$$

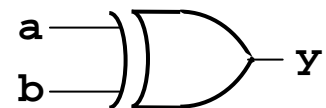
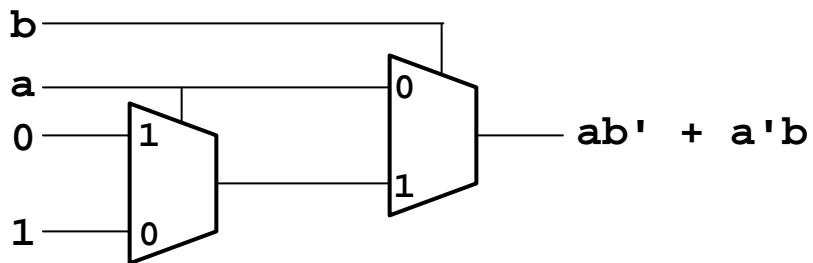
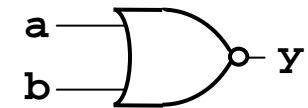
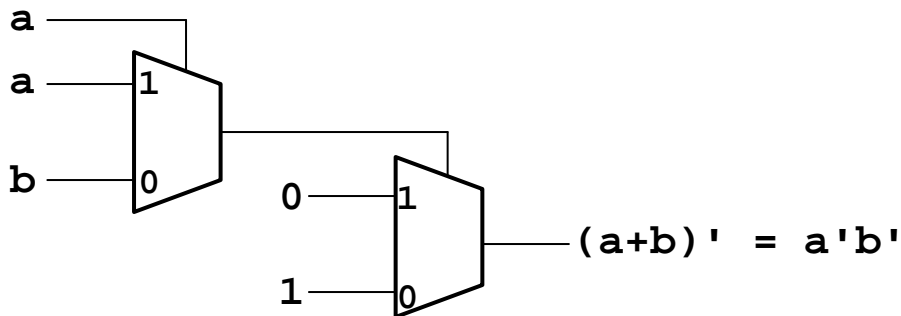
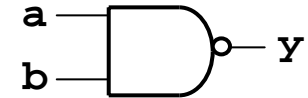
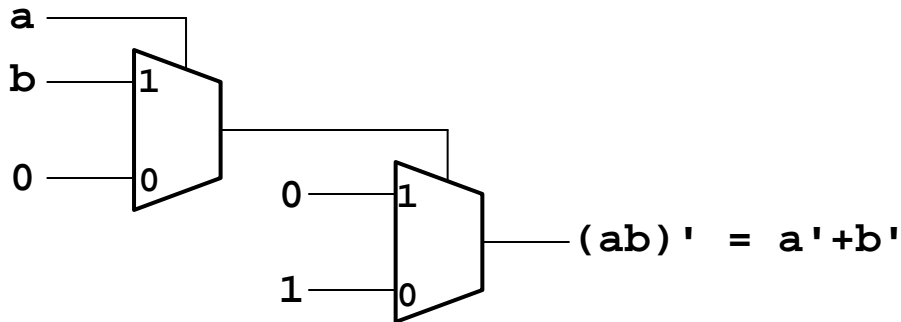
$$\begin{aligned} y &= aa + ba' = a + a'b \\ &= a + b \end{aligned}$$





Celle semplici

[4]

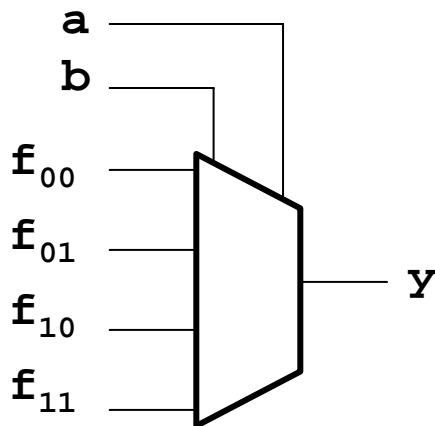




- Mediante multiplexer a quattro ingressi è possibile realizzare qualsiasi funzione di due variabili, infatti:

$$y = a'b'f_{00} + a'bf_{01} + ab'f_{10} + abf_{11}$$

- Un multiplexer è usato come look-up table
 - ▶ Gli ingressi dati sono i valori della funzione
 - ▶ Gli ingressi di selezione sono le variabili

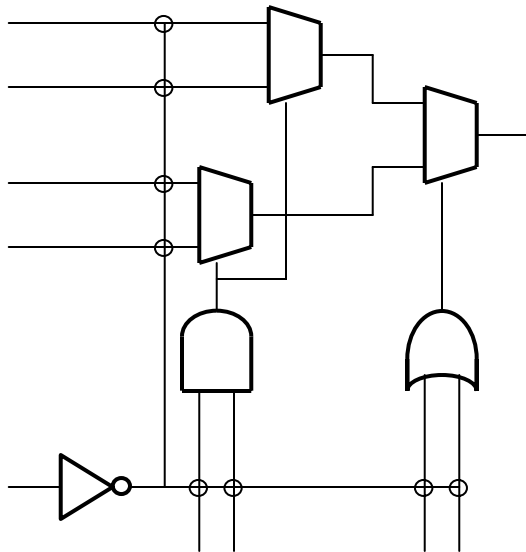


a	b	y
0	0	f_{00}
0	1	f_{01}
1	0	f_{10}
1	1	f_{11}

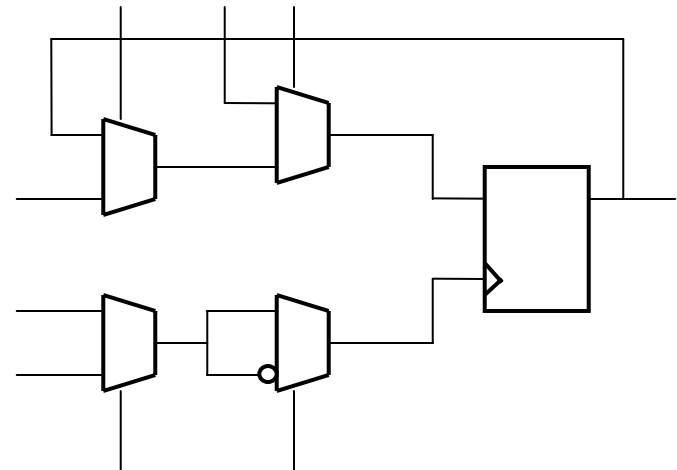


- Alcune celle possono essere leggermente più complesse
- Buon compromesso tra celle semplici e celle complesse
- Esempio: Actel SX-A Family

SX-A C-Module

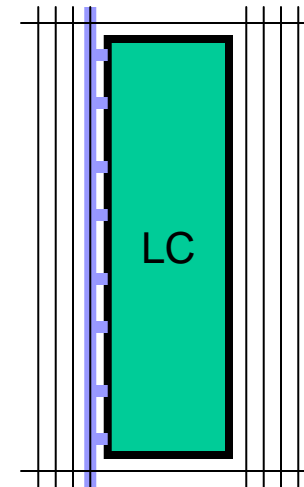
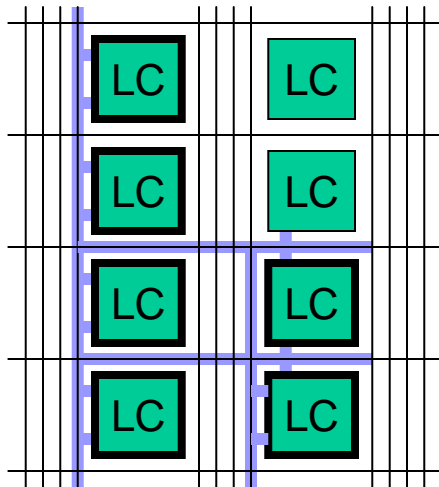


SX-A C-Module





- Molti dispositivi FPGA dispongono di celle complesse
 - ▶ Numero elevato di ingressi/uscite
 - ▶ Elementi sequenziali
 - ▶ Funzioni complesse in una sola cella
 - Riduzione dei problemi di routing
 - Maggiore velocità grazie alla località



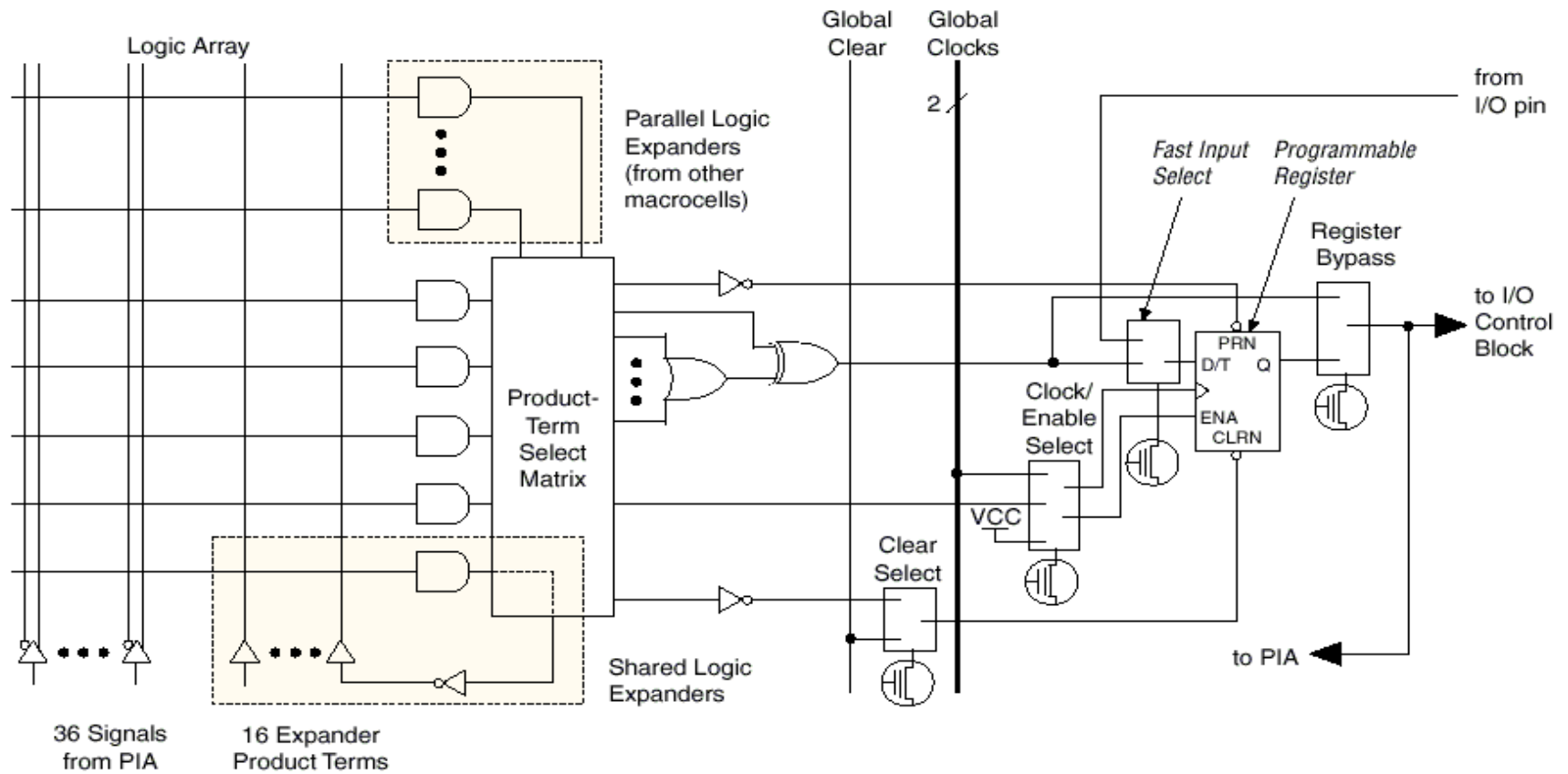


Celle complesse

[2]



Altera MAX 7000 Combinatorial Cell



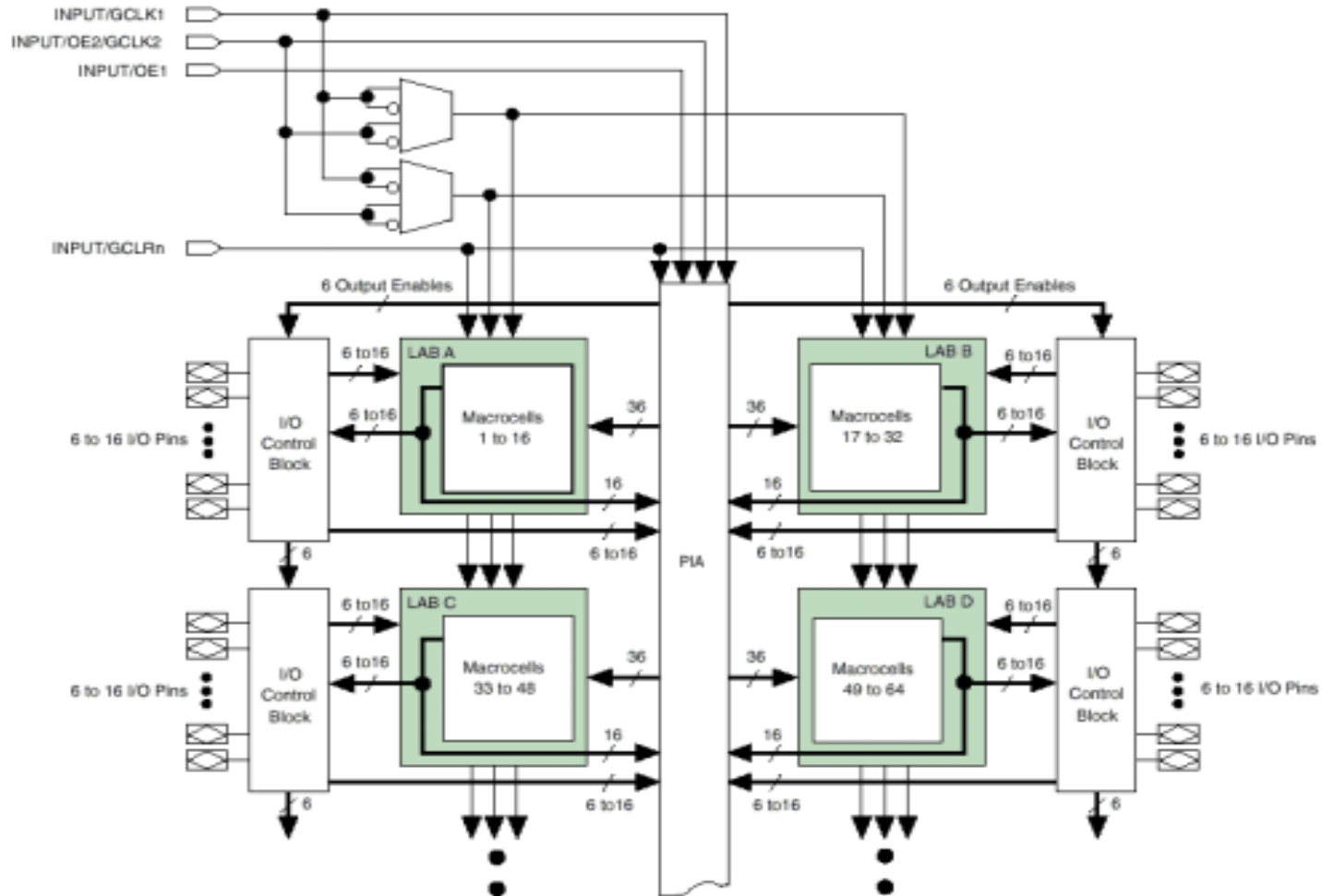


Celle complessa

[3]



Altera MAX 7000 Architecture



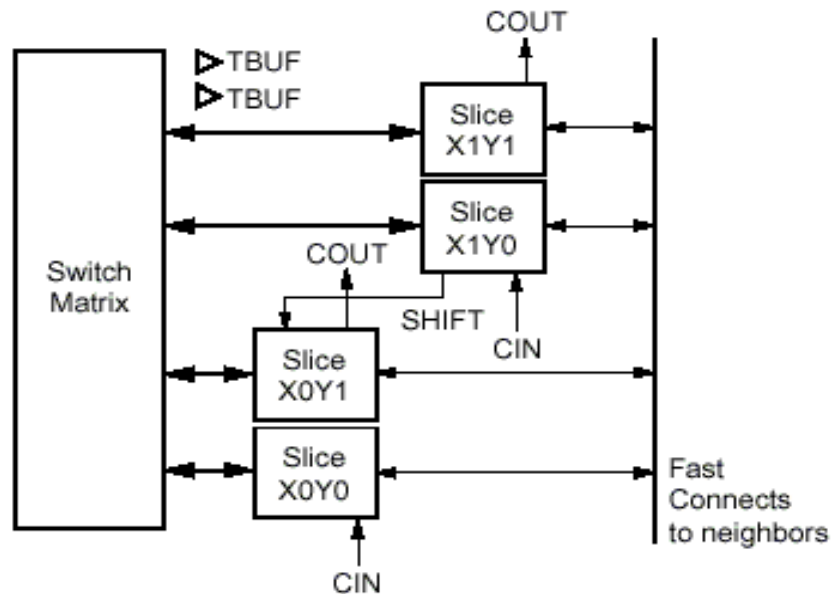


Celle complexe

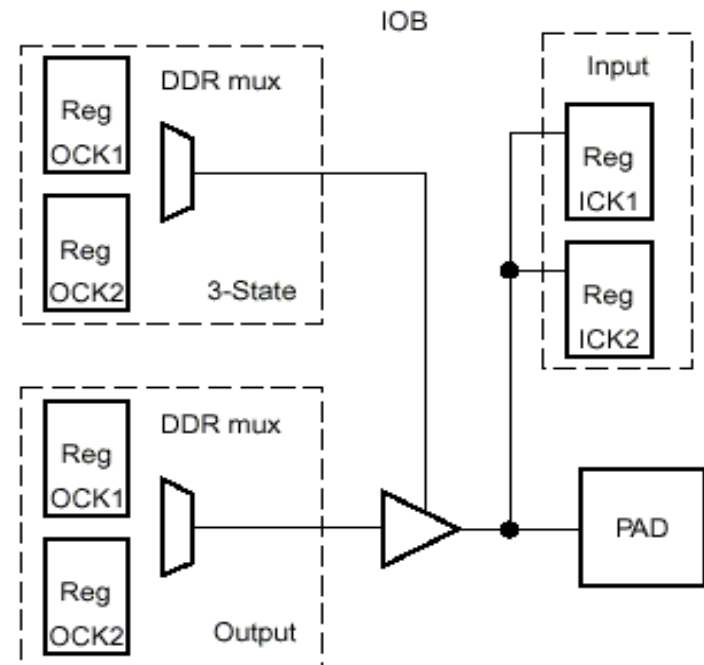
[5]



Xilinx Virtex II Combinatorial Cell



Xilinx Virtex II Input/Output



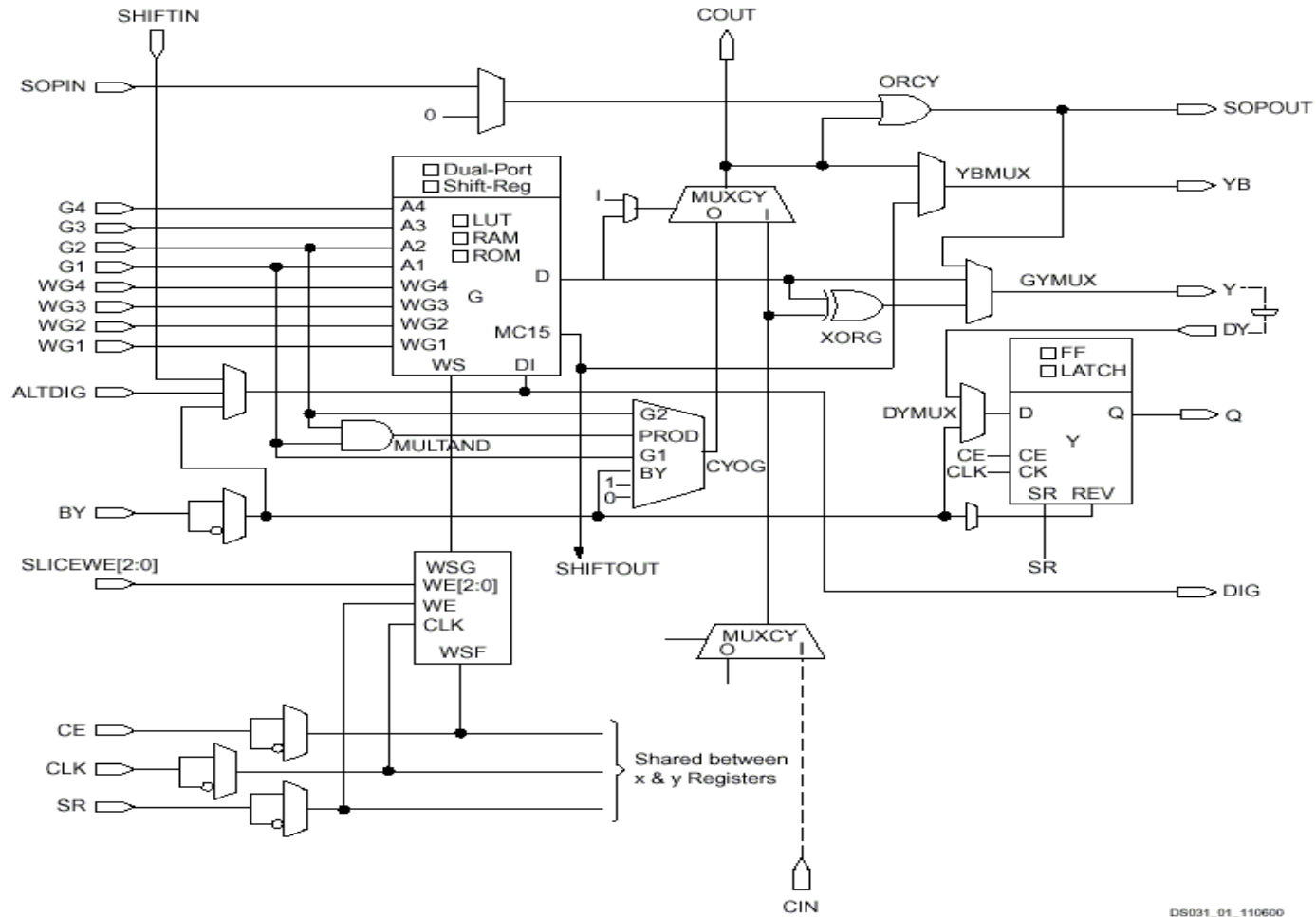


Celle complexe

[6]



Xilinx Virtex II Combinatorial Cell internals



DS031_01_110600