Il progetto dei circuiti logici

M. Rebaudengo, M. Sonza Reorda, L. Sterpone

Politecnico di Torino
Dip. di Automatica e Informatica

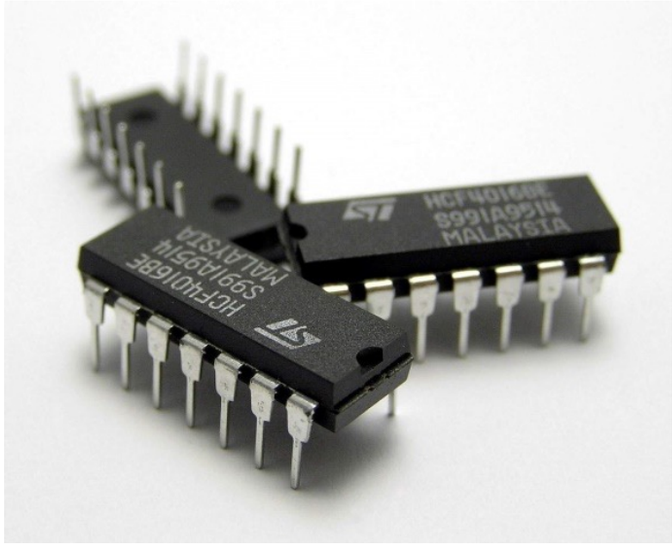
Introduzione

Un *sistema* è una collezione di oggetti (*componenti*) connessi a formare un'entità omogenea con una ben precisa funzionalità.

La funzione svolta dal sistema è determinata da:

- **funzioni svolte dai componenti**
- **modo in cui i componenti sono connessi (*topologia*).**

Un circuito è un particolare tipo di sistema.



Sistemi digitali

Verranno presi in considerazione i sistemi per l'elaborazione delle informazioni, che

- **agiscono su un insieme di dati A in ingresso**
- **producono un insieme di dati B in uscita.**

Si assume che i dati in A e in B siano quantità discrete (sistema *digitale*) corrispondenti a valori rappresentati su bit.

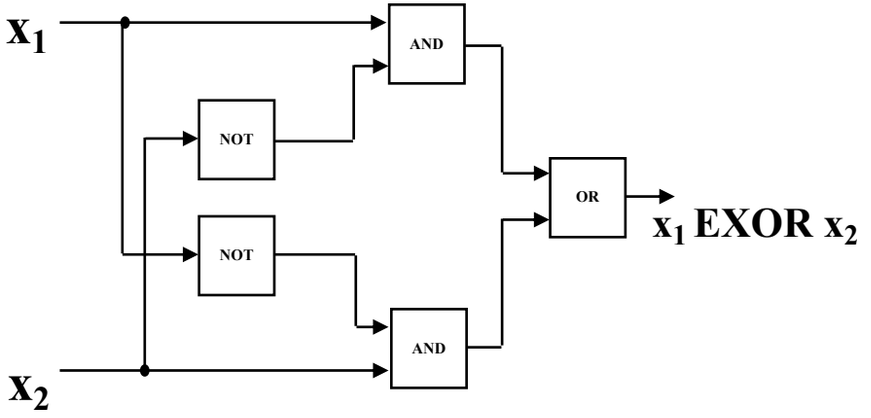
Esempi di sistemi digitali

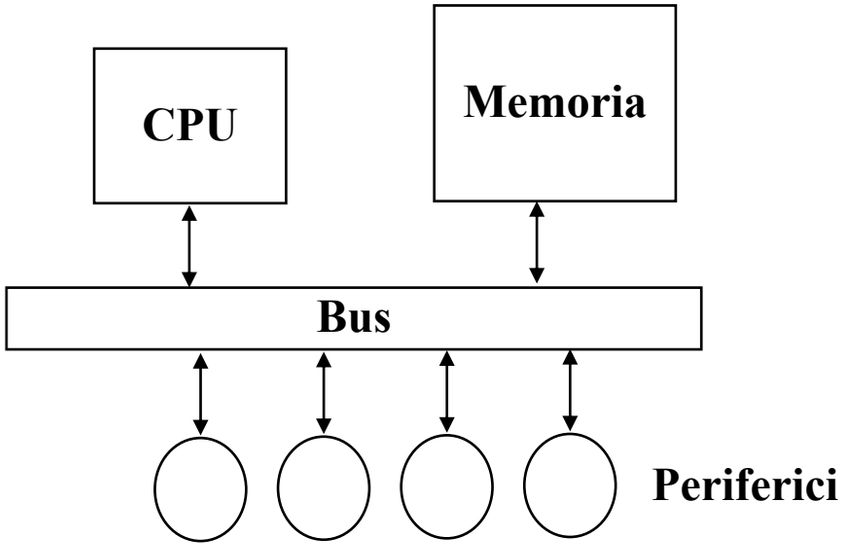
- Circuito digitale semplice (centinaia o migliaia di transistor)
  - Contatore
  - Sommatore
- Circuito digitale complesso (milioni di transistor)
  - Microcontrollore
  - Memoria
- System on Chip o SoC (miliardi di transistor)
- Scheda
  - Combinazione di diversi circuiti
- Sistema di elaborazione complesso
  - Automobile
- Centro di elaborazione dati

Diagramma a blocchi

Permette di descrivere in modo semplice la struttura di un sistema, modellandolo mediante un grafo:

- ***vertici*** $\Rightarrow$ **componenti**
- ***archi*** $\Rightarrow$ **linee per il trasferimento di dati tra componenti.**





Struttura / comportamento

Struttura **= Diagrammi a blocchi senza informazioni funzionali**

Comportamento **= Funzione di trasformazione dai dati in ingresso A a quelli in uscita B.**

Progetto

Siano dati:

- **un sistema da realizzare, di cui è noto il *comportamento* e una serie di altre specifiche**
- **una serie di *componenti* (di solito già disponibili), di cui è noto il comportamento e una serie di altre caratteristiche.**

L'attività di progetto consiste nell'individuare una connessione di componenti tale per cui

- **il comportamento globale sia quello desiderato**
- **le specifiche (tra cui il costo) siano soddisfatte.**

Specifiche

Possono includere specifici vincoli su parametri quali

- **la velocità**
- **il consumo di potenza**
- **l'affidabilità**
- **la durata**
- **ecc.**

Costo

Il costo che si cerca di minimizzare nel progetto è solitamente una combinazione pesata del costo per

- **il progetto**
- **la produzione**
- **la manutenzione.**

Ciclo di vita di un prodotto

Si compone tipicamente di 4 fasi:

- **Specifica**
  - **Si passa dall'idea ad un documento (o altro) che descrive il comportamento del sistema e i vincoli per il progetto**
- **Progetto**
  - **Si passa dalle specifiche ad un modello utilizzabile da chi produce il sistema**
- **Produzione**
  - **Si passa dal modello a un prodotto**
- **Operatività**
  - **Il prodotto è operativo.**

Ciclo di vita di un prodotto

Si compone tipicamente di 4 fasi:

- **Specifica**
  - **Si passa dall'idea ad un documento (o altro) che descrive il comportamento del sistema e i vincoli per il progetto**
- **Progetto**
  - **Si passa dalle specifiche ad un modello utilizzabile da chi produce il sistema**
- **Produzione**
  - **Si passa dal modello ad un prodott**
- **Operatività**
  - **Il prodotto è operativo.**

Prima di passare alla fase successiva è fondamentale verificare la correttezza di quanto sviluppato fino a quel punto.

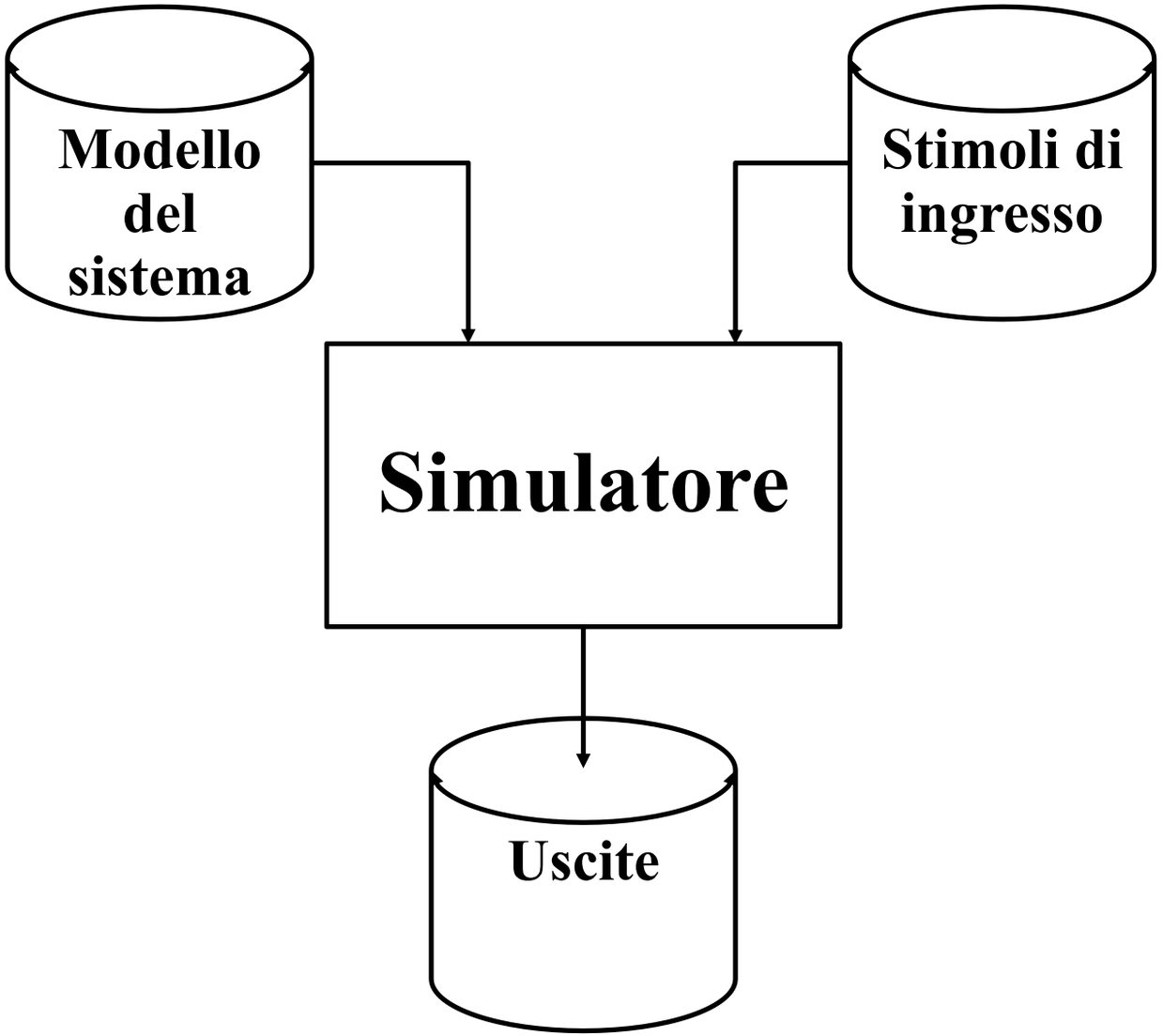
Più si va avanti, più il costo per riparare eventuali errori cresce.

Modello del sistema

L'attività di progetto è strettamente dipendente da quella di *modellamento*, attraverso cui ciò che si vuole realizzare viene descritto formalmente utilizzando un appropriato modello.

Sul modello è spesso possibile eseguire una serie di verifiche (ad esempio tramite *simulazione*) prima della realizzazione fisica.

Simulazione



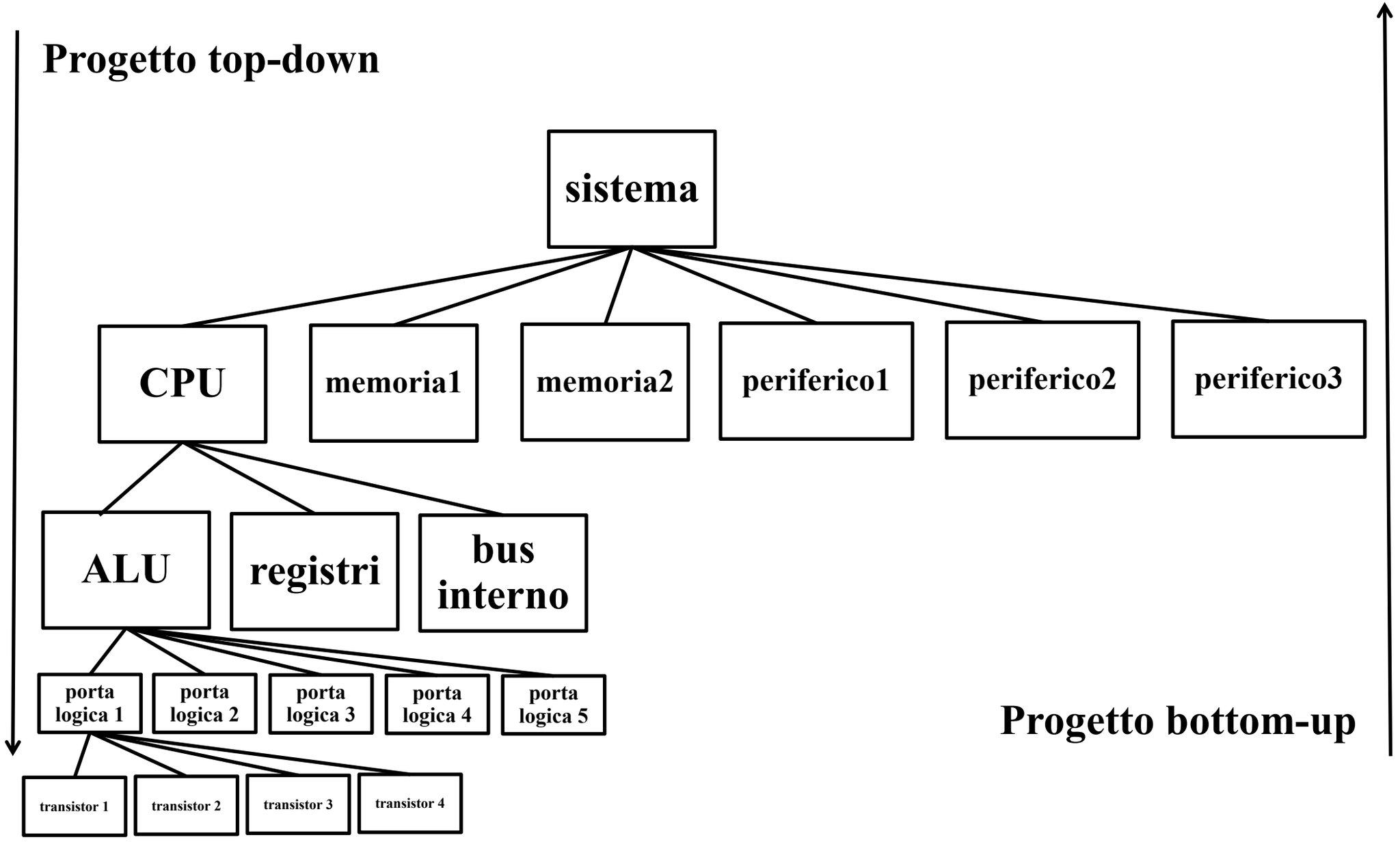
Progetto gerarchico

È di solito eseguito in maniera *top-down*. Nella pratica spesso si combinano tecniche top-down con altre *bottom-up*.

Linee guida:

- **i componenti possono corrispondere ad entità fisiche (circuiti integrati, schede, ecc.) o a moduli predefiniti (ad esempio appartenenti ad una libreria)**
- **ogni componente deve essere il più indipendente possibile, in modo da poter essere sviluppato o modificato in maniera indipendente**
- **in molti casi il progettista conosce le funzioni e le caratteristiche di un modulo, senza conoscere i dettagli sulla sua implementazione (*black box*).**

Esempio


Progetto top-down
sistema
CPU
memoria1
memoria2
periferico1
periferico2
periferico3
ALU
registri
bus interno
porta logica 1
porta logica 2
porta logica 3
porta logica 4
porta logica 5
transistor 1
transistor 2
transistor 3
transistor 4
Progetto bottom-up

Livello di progetto

È definito dal tipo dei componenti utilizzati in ciascuna fase del progetto.

Un componente ad un certo livello corrisponde ad un insieme di componenti al livello inferiore.

Il progetto dei sistemi elettronici può avvenire a diversi livelli:

- **elettrico**
- **transistor (o *switch*)**
- **porte logiche (o *gate*)**
- **registri (o *register transfer*, o RT)**
- **sistema.**

Livelli sistema, registri e porte logiche

Sono i livelli ai quali un sistema elettronico viene visto mano a mano che si procede nel flusso di progetto *top-down*.

Livello	Componenti	Unità di dato	Unità di tempo
Sistema	CPU, processori di IO, memorie	Blocchi di parole	$10^{-3} \div 10^{3}$s
Registri	Registri, reti combinatorie, Reti sequenziali semplici	Parole	$10^{-9} \div 10^{-6}$s
Porte logiche	Porte logiche, flip flop	Bit	$10^{-10} \div 10^{-8}$s

Flusso di progetto

Il progetto di un sistema avviene normalmente tramite l'iterazione ai vari livelli delle stesse operazioni:

- **definizione delle *specifiche* (tramite opportuno formalismo)**
- ***sintesi* (manuale o automatica)**
- ***verifica* (attraverso simulazione o altro).**

Se si segue l'approccio top-down, il risultato del progetto ad un livello spesso costituisce direttamente l'insieme delle specifiche per il livello inferiore.

Sistemi combinatori e sequenziali

Un sistema può essere:

- ***combinatorio*, se i valori delle sue uscite dipendono esclusivamente dai valori applicati sui suoi ingressi in quell'istante.**

  Esempio: sommatore

- ***sequenziale*, se i valori delle uscite dipendono sia dai valori correnti degli ingressi, sia dai valori applicati negli istanti precedenti.**

  Esempio: contatore

Sistemi combinatori

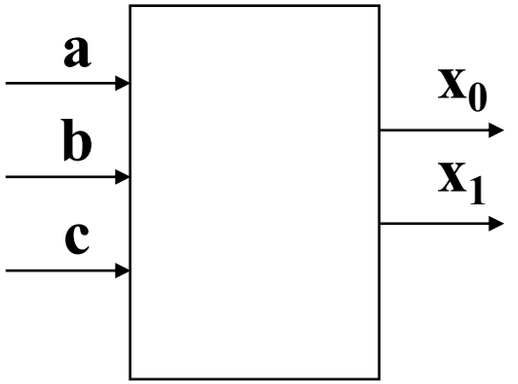
Il valore delle uscite a un certo istante dipende esclusivamente dal valore degli ingressi in quell'istante.

Il comportamento di un sistema combinatorio può essere descritto attraverso

- **una *tavola di verità*, che specifica per ogni combinazione di ingresso la corrispondente combinazione di uscita, oppure**
- **la *funzione booleana* implementata dalle uscite.**

Esempio

Si consideri il sistema che implementa un codificatore prioritario a 3 bit, in grado di generare sull'uscita `x` l'indice del bit di ingresso avente valore 1 e indice massimo:




```
if(a==1)
      x=3;
      else if(b==1)
            x=2;
            else if(c==1)
                  x=1;
                  else
                        x=0;
```

Esempio (II)

Il comportamento del sistema può essere descritto come tavola di verità o come funzione booleana:

a	b	c	x_1	x_0
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	1	0
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Tavola di verità

Esempio (II)

Il comportamento del sistema può essere descritto come tavola di verità o come funzione booleana:

a	*b*	*c*	x_1	x_0
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	1	0
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Tavola di verità

Funzione booleana

$$x_1 = a + b$$

$$x_0 = a + \bar{b}c$$

Sistemi sequenziali

La risposta di un sistema sequenziale in un qualsiasi istante dipende da

- **valori X applicati in quell'istante agli ingressi del sistema**
- **storia passata del sistema stesso, che viene di solito codificata dalla *variabile di stato* Y del sistema.**

In genere lo stato del sistema non è direttamente osservabile dall'esterno.

Formati di descrizione

La descrizione di un sistema sequenziale può assumere 3 forme:

- **tavola degli stati/uscite**
- **diagramma degli stati/uscite**
- **funzione di transizione degli stati e funzione delle uscite.**

Esempio

Si consideri un sistema avente un ingresso I e un'uscita O.

Il sistema campiona con una frequenza prefissata l'ingresso I.

L'uscita O assume

- **il valore 1 se durante i 3 istanti di campionamento precedenti l'ingresso I ha assunto i valori 101**
- **il valore 0 diversamente.**

Variabile di stato

- **Poiché deve memorizzare quanto necessario della storia passata del sistema, in prima battuta potrebbe memorizzare i valori assunti da I negli ultimi 3 istanti di campionamento**
- **Quindi la variabile di stato può assumere 8 possibili valori**
- **È possibile dimostrare che sono sufficienti 5 valori.**

Diagramma degli stati/uscite

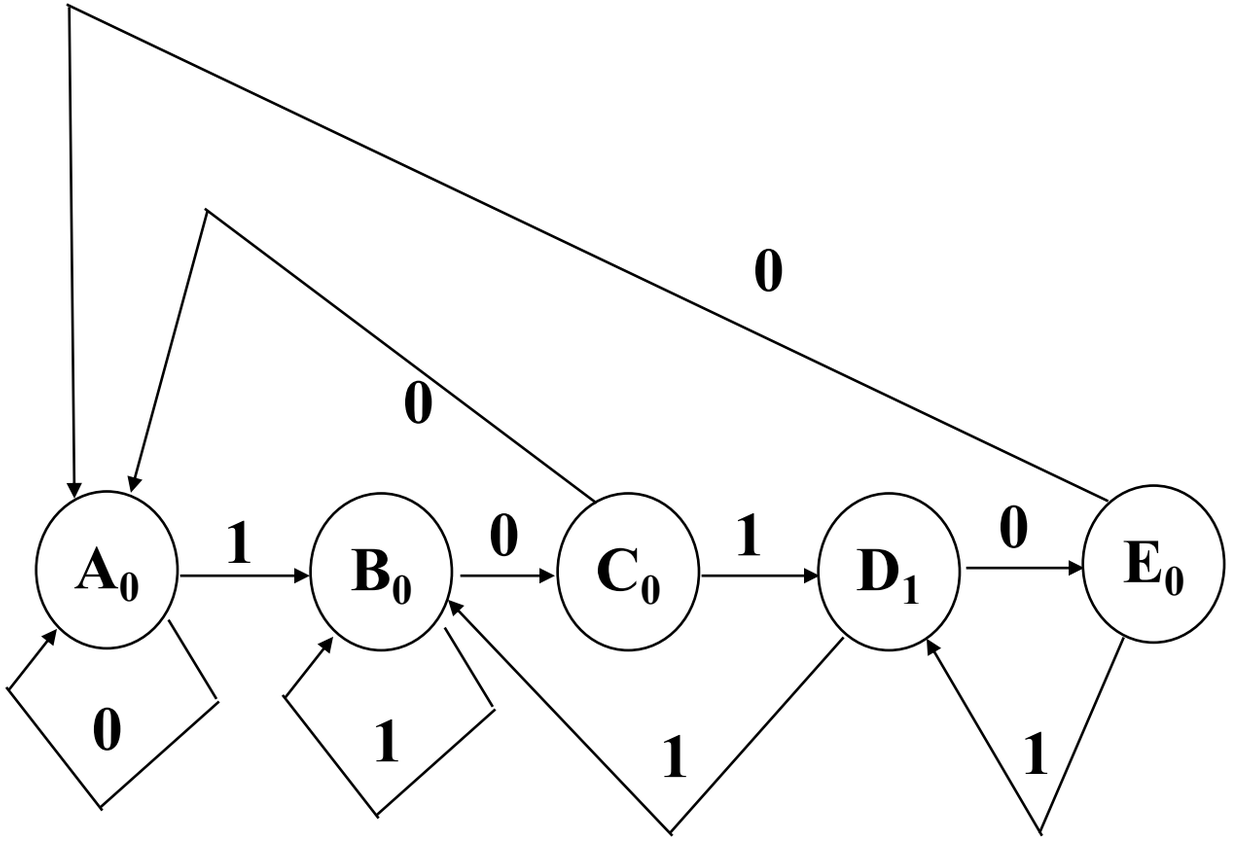


Diagramma degli stati/uscite

In questo caso l'uscita dipende esclusivamente dallo stato in cui si trova il sistema. Quindi è associata al vertice.

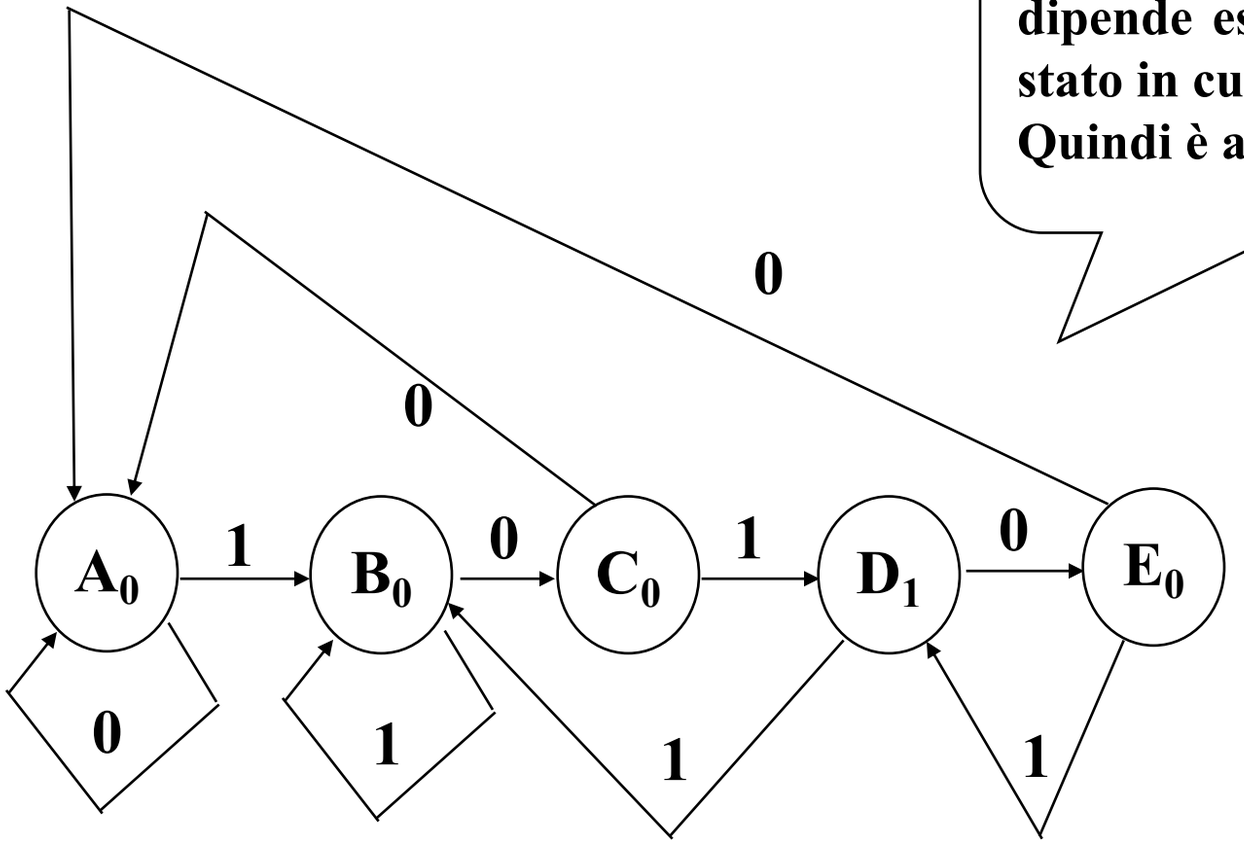


Diagramma degli stati/uscite

Nel caso più generale le uscite dipendono dallo stato in cui si trova il sistema e dal valore degli ingressi.
Quindi le uscite sono associate agli archi.

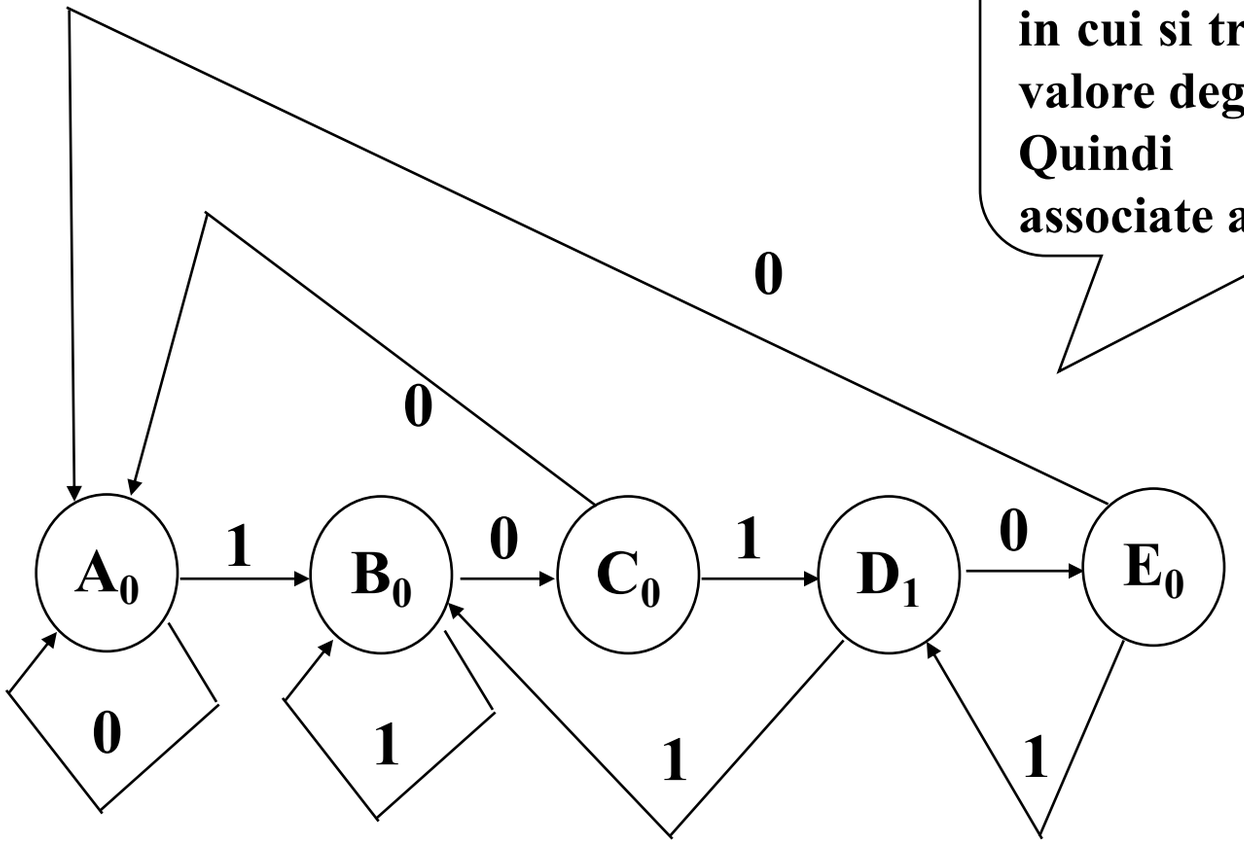


Tavola degli stati/uscite

Stato corrente	Ingresso	Stato futuro	Uscita
A	0	A	0
A	1	B	0
B	0	C	0
B	1	B	0
C	0	A	0
C	1	D	0
D	0	E	1
D	1	B	1
E	0	A	0
E	1	D	0

Funzione di transizione

Siano X, Y e Z tre spazi associati alle variabili di ingresso, di stato e di uscita, rispettivamente.

La *funzione di transizione* effettua la seguente trasformazione

$$f: X \times Y \rightarrow Y \times Z$$

Per poter esprimere f come funzione booleana è necessario eseguire preventivamente l'associazione tra i simboli che identificano gli stati e la rispettiva rappresentazione booleana.

Esempio: funzione di transizione in forma simbolica

$f(0,A) \rightarrow A,0$

$f(1,A) \rightarrow B,0$

$f(0,B) \rightarrow C,0$

$f(1,B) \rightarrow B,0$

$f(0,C) \rightarrow A,0$

$f(1,C) \rightarrow D,0$

$f(0,D) \rightarrow E,1$

$f(1,D) \rightarrow B,1$

$f(0,E) \rightarrow A,0$

$f(1,E) \rightarrow D,0$

Macchina a Stati Finiti

- **Un sistema sequenziale è anche noto come Macchina a Stati Finiti (*Finite State Machine* o *FSM*)**
- **Le FSM possono essere**
  - **FSM di *Moore*: il valore dell'uscita dipende esclusivamente dal valore corrente della variabile di stato**
  - **FSM di *Mealy*: il valore dell'uscita dipende dal valore corrente della variabile di stato e dal valore corrente dell'ingresso.**

Progetto a livello elettrico

Il sistema viene modellato come interconnessione di componenti (quali resistori, condensatori, induttori, ecc.) connessi tra loro in serie o parallelo.

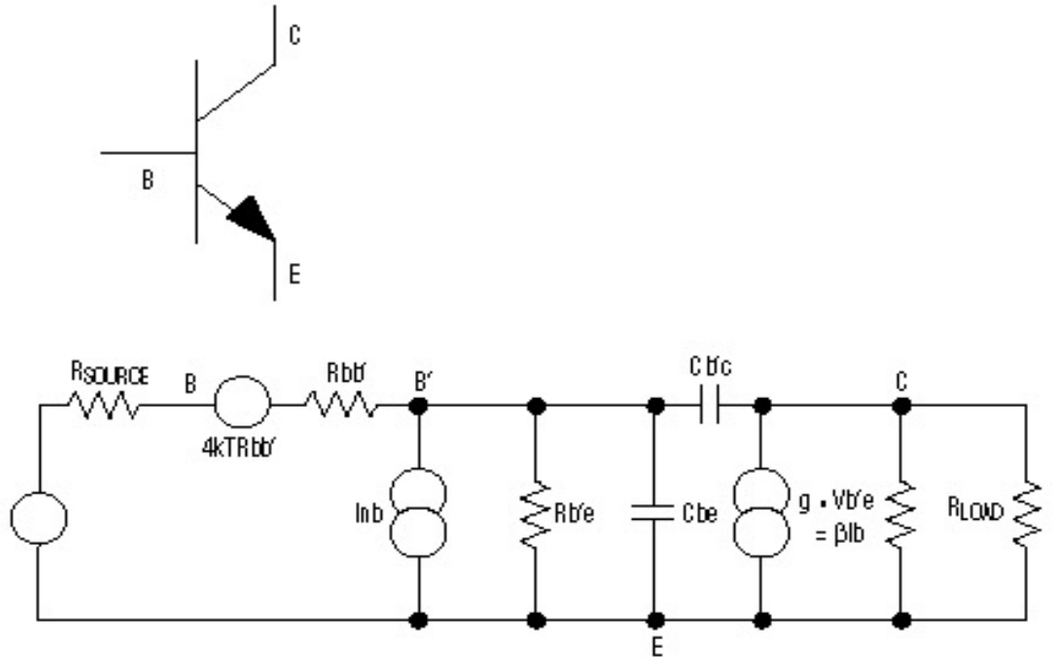
Il calcolo dei valori di corrente e tensione nei diversi punti (a regime o nel transitorio) comporta la risoluzione di un sistema di equazioni differenziali.

Sul mercato esistono vari pacchetti SW (ad es. *SPICE*) in grado di simulare circuiti di dimensioni comunque ridotte.

Sta al progettista trovare un soddisfacente compromesso tra precisione e tempo di calcolo.

Esempio

È possibile modellare il comportamento di un transistor attraverso opportuni modelli, tra cui il seguente (noto come *modello di Giacoletto*):



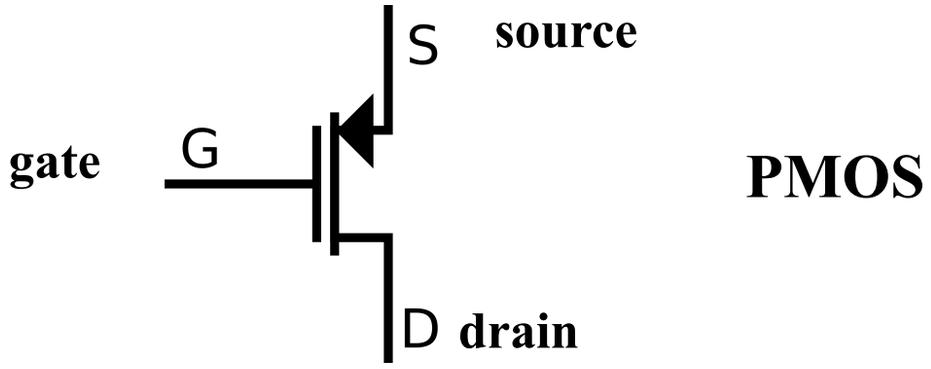
Progetto a livello transistor

Il sistema viene visto come un'interconnessione di *transistor*, considerati come interruttori pilotati dal segnale di base.

Il transistor Metal-Oxide-Semiconductor Field-Effect Transostor (MOSFET) è un transistor ad effetto di campo largamente usato in elettronica digitale

A seconda della tecnologia utilizzata (PMOS o NMOS) il transistor è chiuso/aperto quando il segnala di base ha una tensione inferiore/maggiore ad una certa soglia.

In ogni caso, la velocità di commutazione di un transistor è dell'ordine dei nanosecondi.

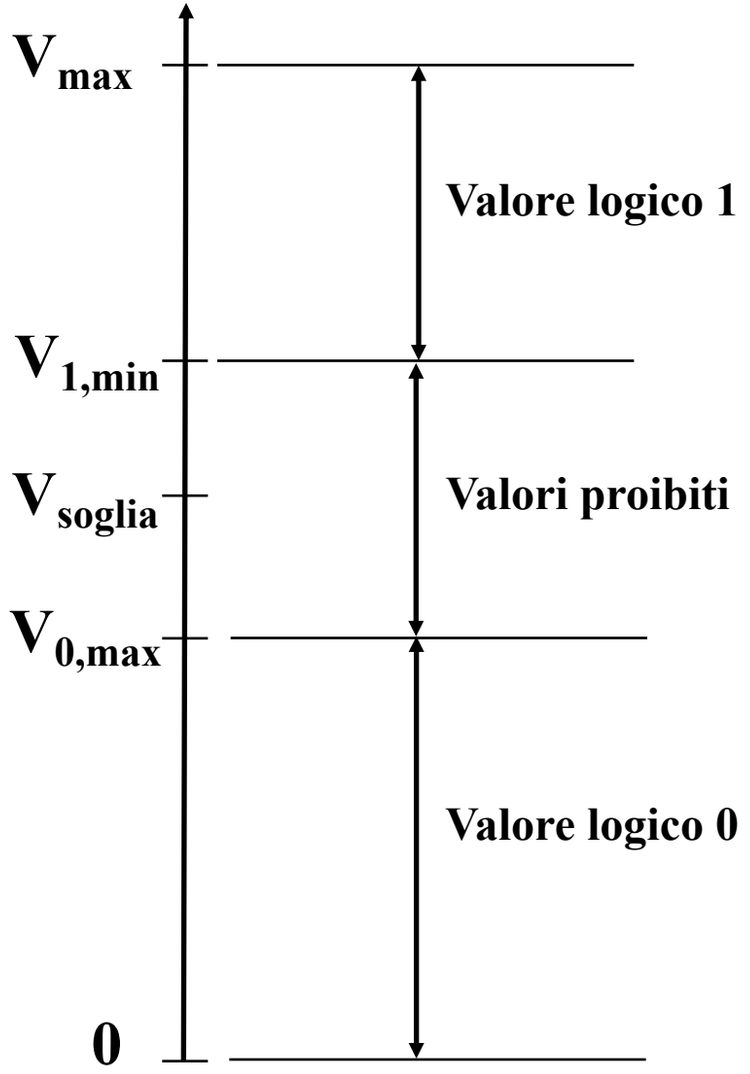


Tensioni/valori logici

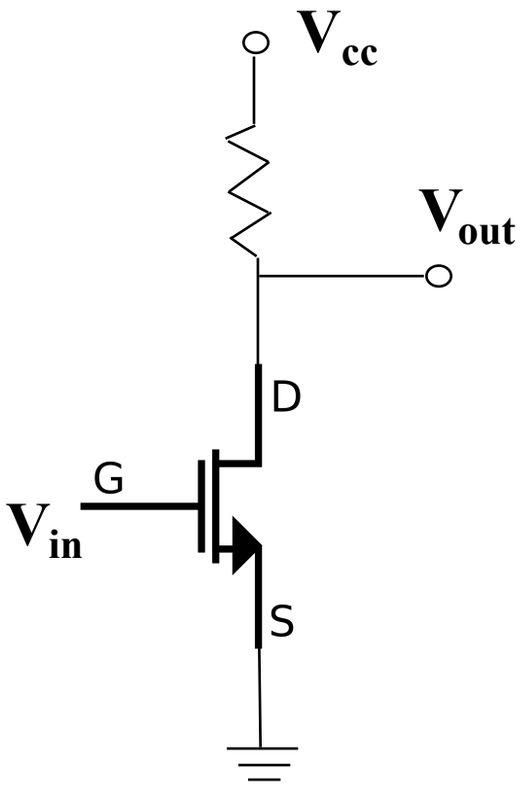
In un circuito elettronico è comune associare i valori logici (0/1) alle tensioni (alta/bassa).

Tale operazione richiede di fissare

- **una soglia per discriminare le tensioni alte da quelle basse**
- **una fascia di tensioni proibita, per evitare gli effetti del rumore.**



Inverter NMOS



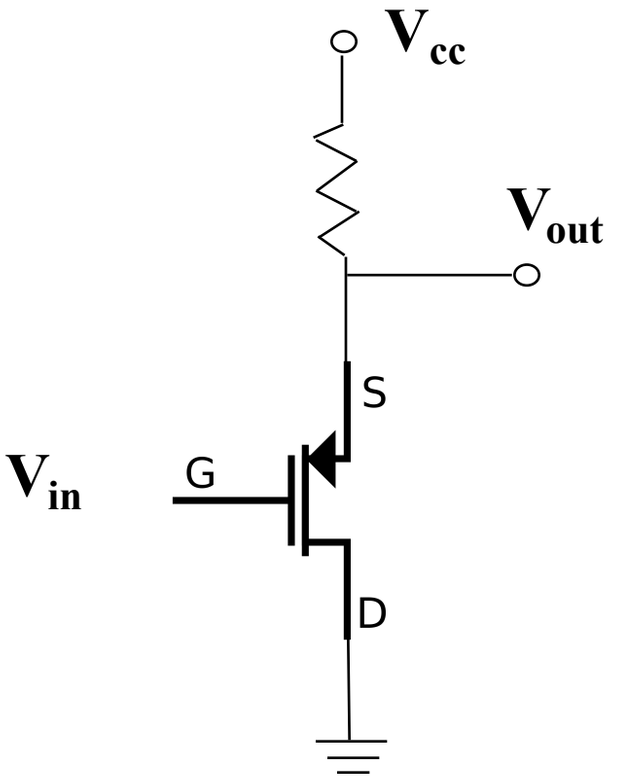
Se $V_{in} < V_{soglia}$:

$V_{out} = V_{cc}$

Se $V_{in} > V_{soglia}$:

$V_{out} = V_{ground}$

Buffer PMOS



Se $V_{in} > V_{soglia}$:

$V_{out} = V_{cc}$

Se $V_{in} < V_{soglia}$:

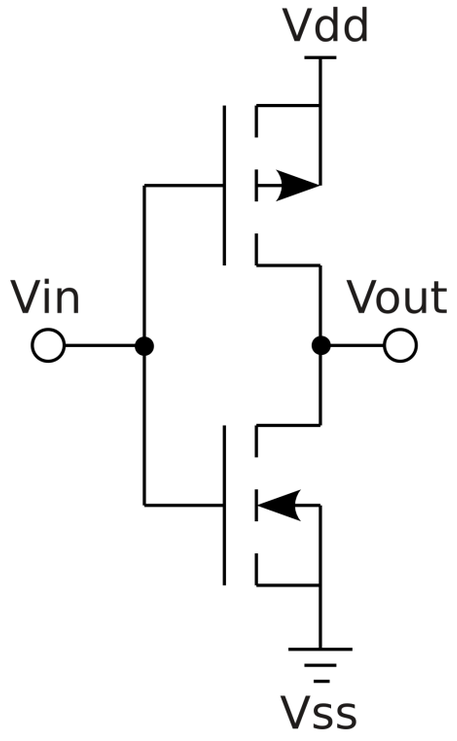
$V_{out} = V_{ground}$

Inverter CMOS

Integra la tecnologia NMOS con quella PMOS.

C’è sempre un transistor aperto e uno chiuso.

In questo modo non c’è mai un collegamento tra alimentazione e massa, e si riduce il consumo.



Porte logiche

Utilizzando i transistor è possibile implementare una qualsiasi porta logica.

È quindi possibile costruire una *libreria* di porte logiche utilizzabili per il progetto al livello superiore.

Progetto a livello porte logiche

A questo livello i componenti sono le porte logiche (*gate*).

Per questa ragione è anche noto come *livello logico*.

Le porte logiche sono elementi operanti su variabili binarie che possono assumere i due valori 0 e 1.

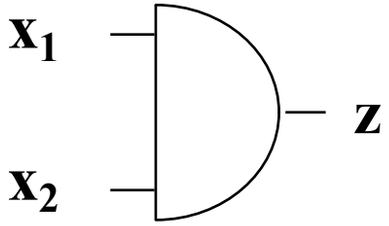
Le informazioni trattate a questo livello sono *segnali* binari; idealmente, ogni linea del circuito può cioè assumere 2 soli valori di tensione, corrispondenti ai 2 valori logici 0 e 1.

Porte logiche (I)

Compongono i circuiti combinatori.

L'insieme delle porte logiche utilizzate è formato dai seguenti componenti

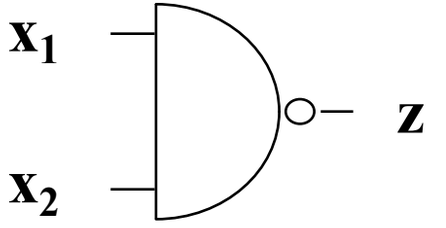
AND



$$Z = X_1 X_2 = X_1 \wedge X_2$$

0	0	0
0	1	0
1	0	0
1	1	1

NAND

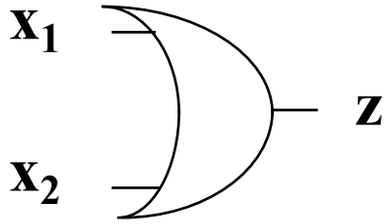


$$Z = \overline{0_1 0_2} = \overline{X_1 \wedge X_2}$$

0	0	1
0	1	1
1	0	1
1	1	0

Porte logiche (II)

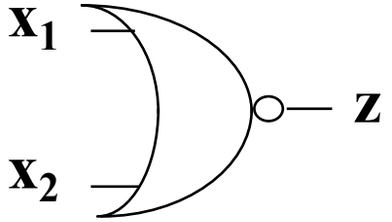
OR



$$z = x_1 + x_2 = x_1 \vee x_2$$

0	0	0
0	1	1
1	0	1
1	1	1

NOR

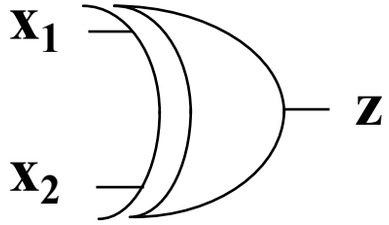


$$z = \overline{x_1 + x_2} = \overline{x_1 \vee x_2}$$

0	0	1
0	1	0
1	0	0
1	1	0

Porte logiche (III)

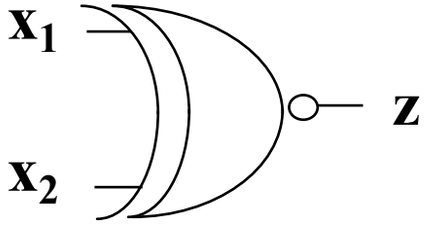
EXOR



$$Z = X_1 \oplus X_2$$

0	0	0
0	1	1
1	0	1
1	1	0

EXNOR

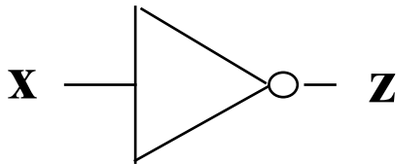


$$Z = \overline{X_1 \oplus X_2}$$

0	0	1
0	1	0
1	0	0
1	1	1

Porte logiche (IV)

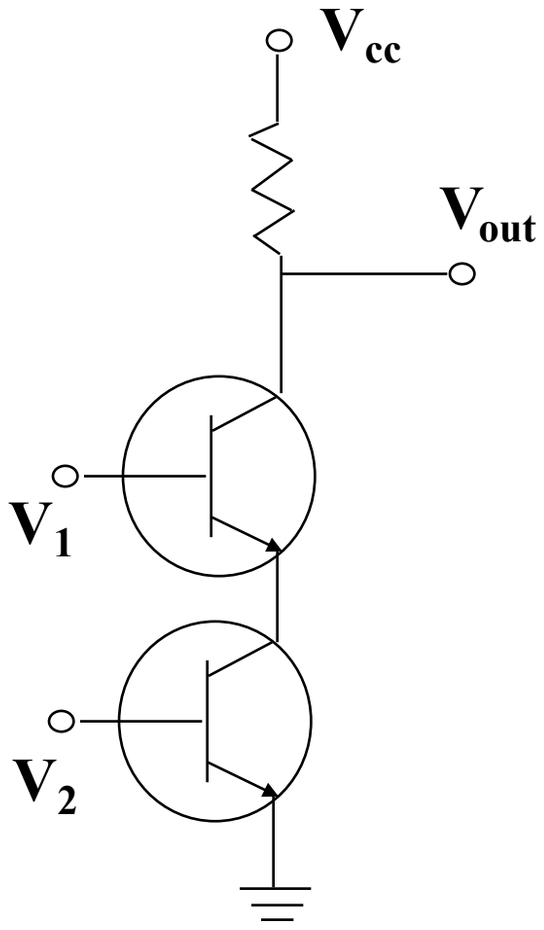
NOT



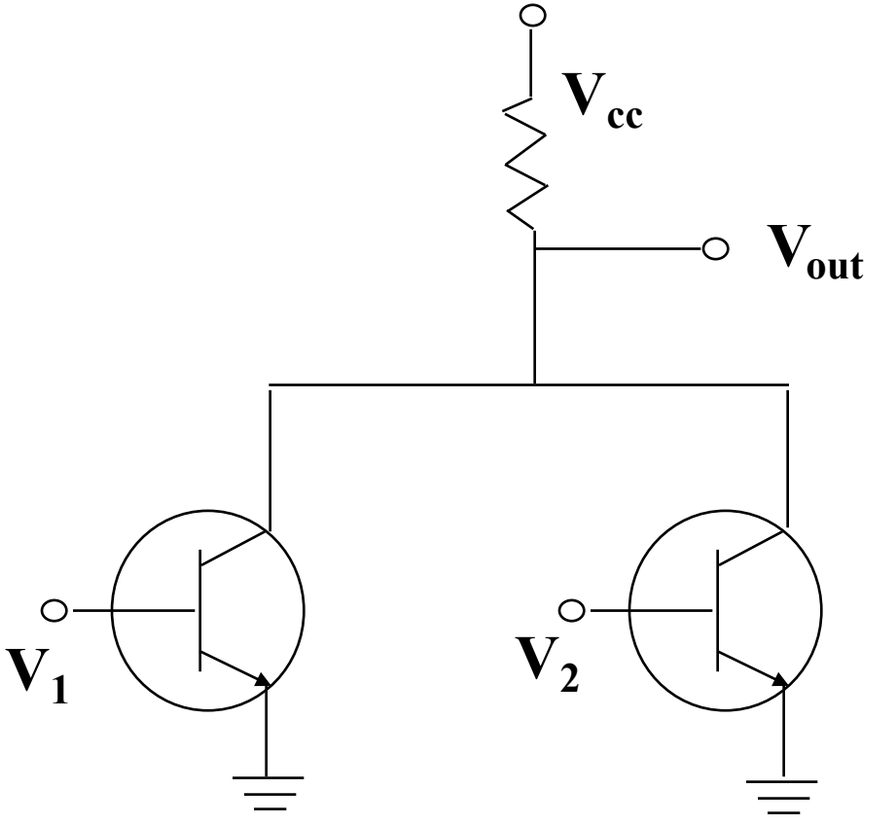
$$Z = \overline{X}$$

0	**1**
1	**0**

Implementazione delle porte logiche tramite transistor NMOS



Porta NAND



Porta NOR

Insieme di porte completo

Un insieme di tipi di porte logiche si dice *completo* se utilizzando i soli tipi di porte in esso contenuti si può realizzare qualsivoglia funzione combinatoria.

Sono insiemi completi:

- **{NAND}**
- **{NOR}**
- **{AND, NOT}**
- **{OR, NOT}**
- **{AND, OR, NOT}**

Nella pratica sono frequenti i circuiti che implementano una funzione combinatoria utilizzando le sole porte NAND (o NOR).

Circuiti combinatori

Implementano funzioni combinatorie.

Una *funzione combinatoria* è una trasformazione

$$z: B^n \rightarrow B,$$

dove B={0,1}.

Le funzioni combinatorie non coinvolgono il tempo (a differenza di quelle sequenziali) e possono essere descritte (se il numero di variabili è piccolo) attraverso *tavole di verità*.

I circuiti combinatori possono essere implementati attraverso opportune combinazioni di porte logiche.

Circuiti combinatori
ben formati

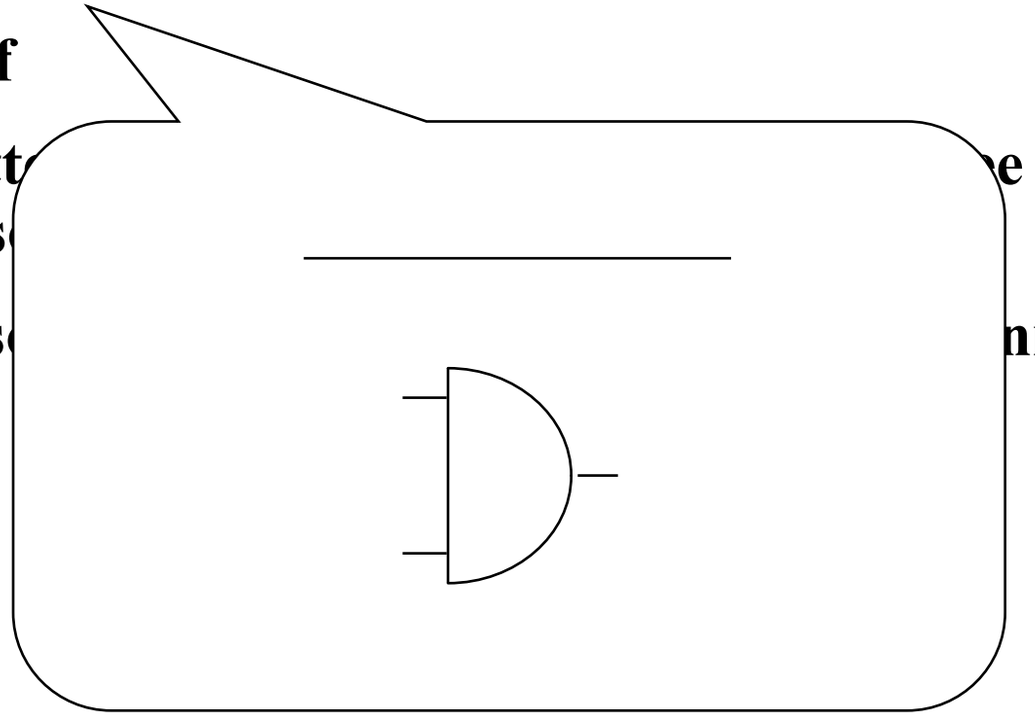
Si dicono *Circuiti Combinatori Ben Formati* (ccbf) i circuiti combinatori che soddisfano le seguenti regole:

- **una singola linea o una singola porta è un ccbf**
- **la giustapposizione di 2 ccbf è un ccbf**
- **se C_1 e C_2 sono due ccbf, il circuito ottenuto connettendo un insieme di linee di uscita di C_1 ad un insieme di linee di ingresso di C_2 è un ccbf**
- **se x_i ed x_j sono due linee di ingresso ad un ccbf, il circuito ottenuto connettendo insieme x_i e x_j è un ccbf**
- **non sono presenti cicli.**

Circuiti combinatori
ben formati

Si dicono *Circuiti Combinatori Ben Formati* (ccbf) i circuiti combinatori che soddisfano le seguenti regole:

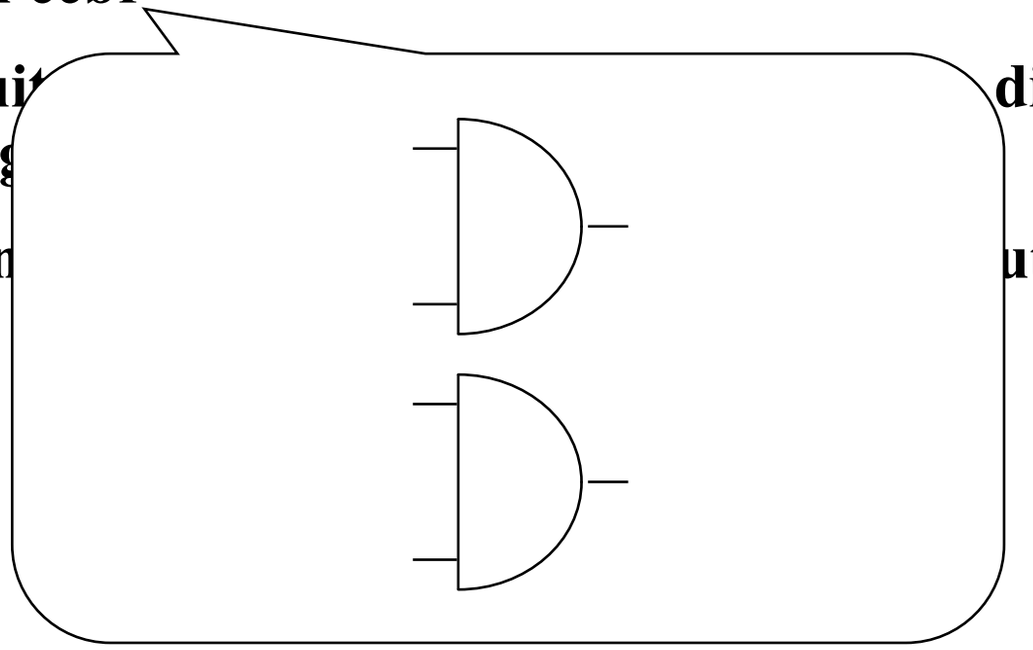
- **una singola linea o una singola porta è un ccbf**
- **la giustapposizione di 2 ccbf è un ccbf**
- **se C_1 e C_2 sono due ccbf, il circuito ott e di uscita di C_1 ad un insieme di linee di ingress**
- **se x_i ed x_j sono due linee di ingress nnettendo insieme x_i ed x_j è un ccbf**
- **non sono presenti cicli.**



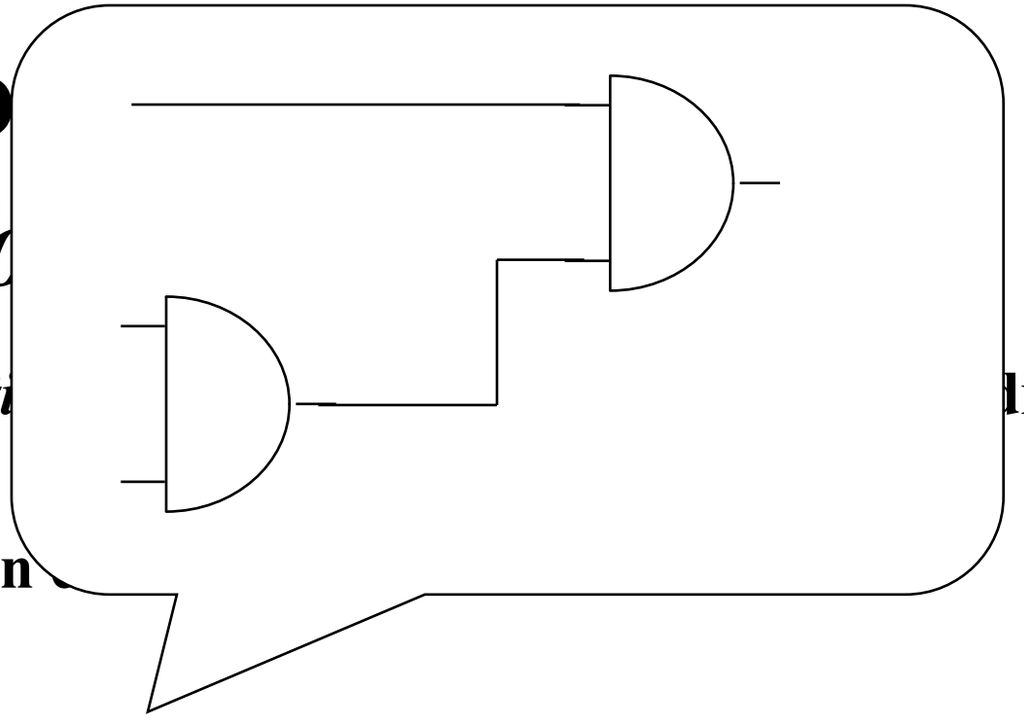
Circuiti combinatori
ben formati

Si dicono *Circuiti Combinatori Ben Formati* (ccbf) i circuiti combinatori che soddisfano le seguenti regole:

- **una singola linea o una singola porta è un ccbf**
- **la giustapposizione di 2 ccbf è un ccbf**
- **se C_1 e C_2 sono due ccbf, il circuit… di linee di uscita di C_1 ad un insieme di linee di ing…**
- **se x_i ed x_j sono due linee di in… uto connettendo insieme x_i ed x_j è un ccbf**
- **non sono presenti cicli.**



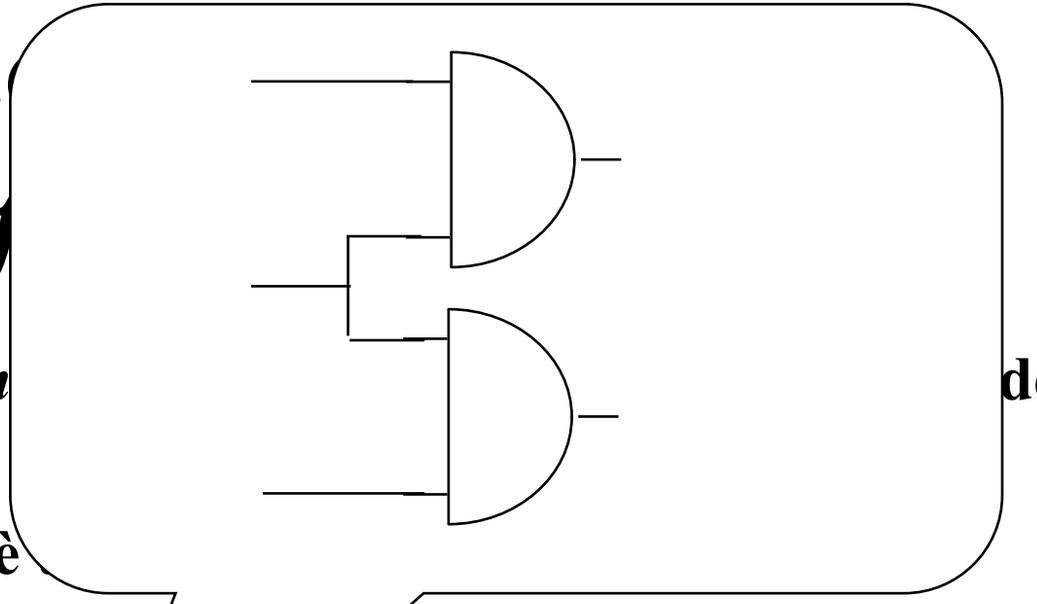
Circuiti co
ben fo



Si dicono *Circuiti Combinatori Ben Formati* (ccbf) quei circuiti che soddisfano le seguenti regole:

- **una singola linea o una singola porta è un ccbf**
- **la giustapposizione di 2 ccbf è un ccbf**
- **se C_1 e C_2 sono due ccbf, il circuito ottenuto connettendo un insieme di linee di uscita di C_1 ad un insieme di linee di ingresso di C_2 è un ccbf**
- **se x_i ed x_j sono due linee di ingresso ad un ccbf, il circuito ottenuto connettendo insieme x_i e x_j è un ccbf**
- **non sono presenti cicli.**

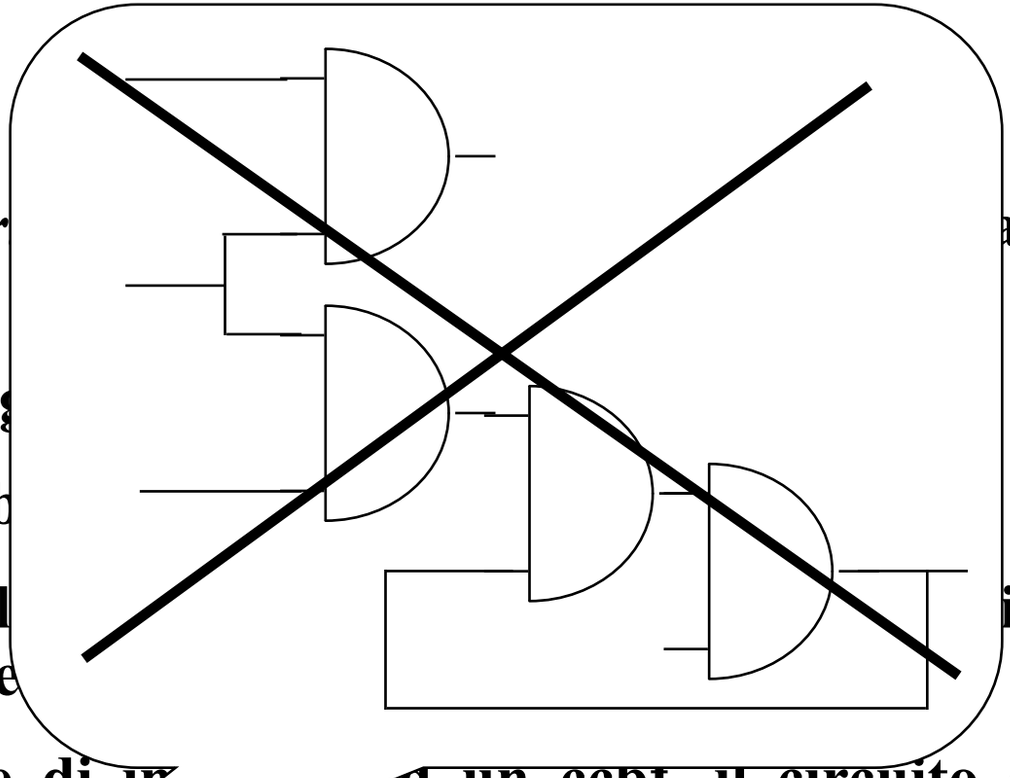
Circuiti c
ben f



Si dicono *Circuiti Combinatori Ben Form* ddisfano le seguenti regole:

- **una singola linea o una singola porta è**
- **la giustapposizione di 2 ccbf è un ccbf**
- **se C_1 e C_2 sono due ccbf, il circuito ott o connettendo un insieme di linee di uscita di C_1 ad un insieme di linee di ingr so di C_2 è un ccbf**
- **se x_i ed x_j sono due linee di ingresso ad un ccbf, il circuito ottenuto connettendo insieme x_i e x_j è un ccbf**
- **non sono presenti cicli.**

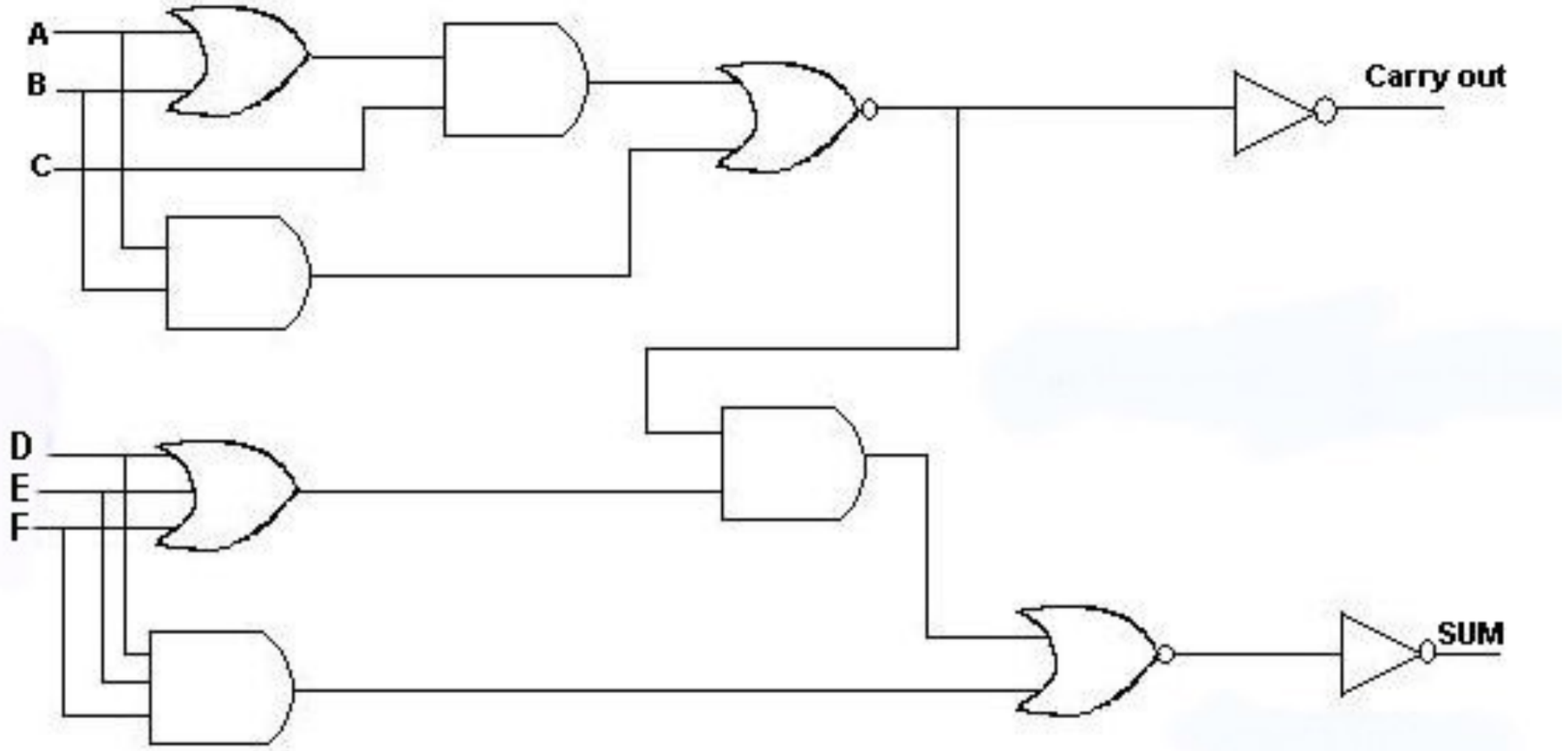
Circuiti combinatori



Si dicono *Circuiti Combinator* atori che soddisfano le seguenti regole:

- **una singola linea o una sing**
- **la giustapposizione di 2 ccb**
- **se C_1 e C_2 sono due ccbf, il ieme di linee di uscita di C_1 ad un insieme di linee**
- **se x_i ed x_j sono due linee di i ad un ccbf, il circuito ottenuto connettendo insieme x_i ed x_j è un ccbf**
- **non sono presenti cicli.**

Esempio di circuito ben formato



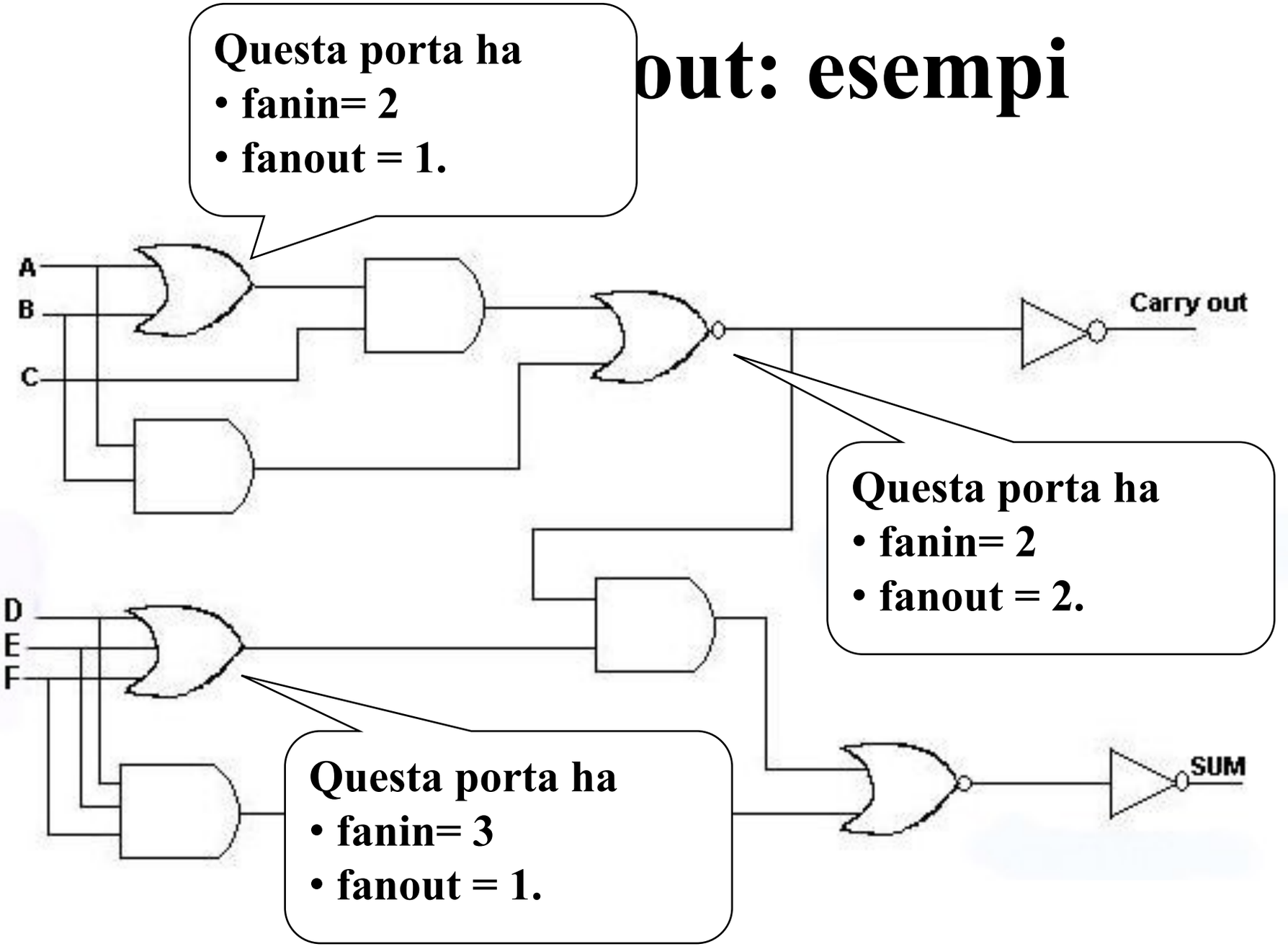
Fanin

È il numero di segnali in ingresso ad una porta.

Fanout

È il numero di altre porte pilotate dall'uscita di una porta.

out: esempi


Questa porta ha
• fanin= 2
• fanout = 1.
A
B
C
Carry out
Questa porta ha
• fanin= 2
• fanout = 2.
D
E
F
Questa porta ha
• fanin= 3
• fanout = 1.
SUM

Progetto di circuiti combinatori

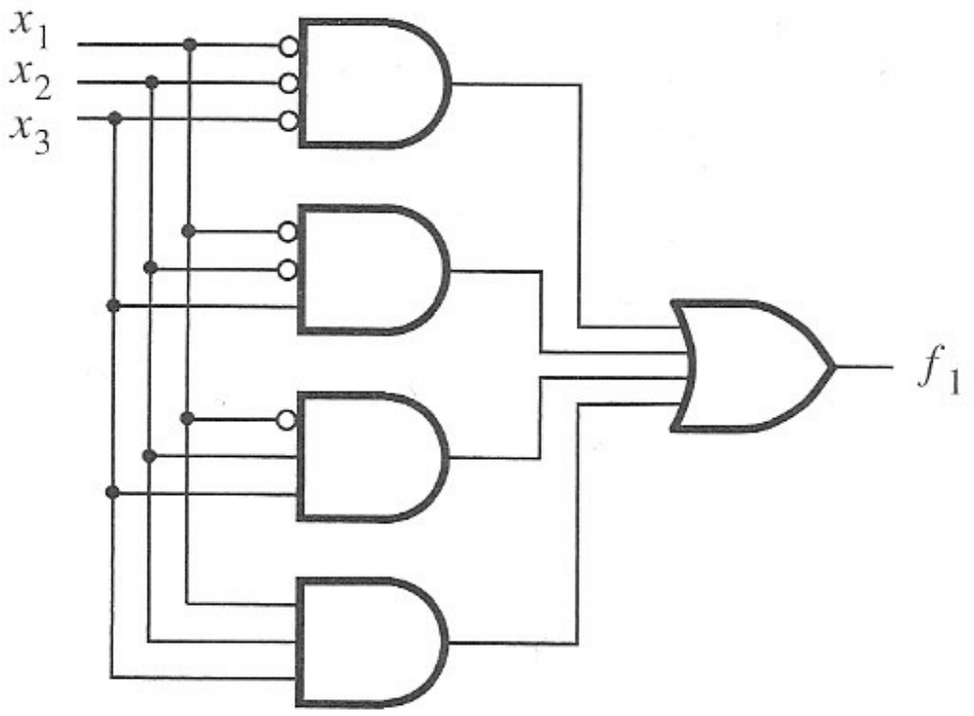
- **Può essere eseguito partendo dalla tabella di verità**
- **Per ogni riga corrispondente a un valore 1 in uscita si inserisce nel circuito una porta AND**
- **Gli ingressi della porta sono affermati o negati a seconda del valore nella riga**
- **Le uscite della porta AND alimentano una porta OR.**

Esempio

x_1	x_2	x_3	f_1
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Esempio

x_1	x_2	x_3	f_1
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1



$$f_1 = \overline{x_1}\,\overline{x_2}\,\overline{x_3} + \overline{x_1}\,\overline{x_2}x_3 + \overline{x_1}x_2\,x_3 + x_1x_2x_3$$

Equivalenza di reti logiche

- **È possibile che alla stessa tabella di verità corrisponda più di un circuito (e più di una espressione Booleana)**
- **In tal caso**
  - **il comportamento dei vari circuiti è lo stesso**
  - **i circuiti (e le funzioni) si dicono *equivalenti*.**

Esempio (II)

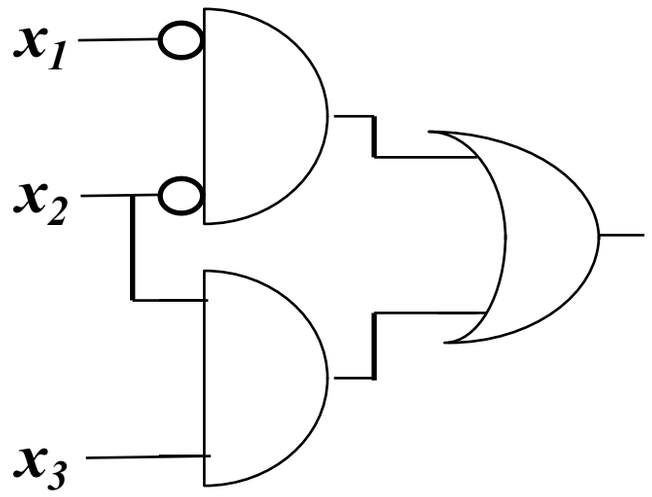
La derivazione di una forma dall'altra può essere ottenuta applicando le regole dell'algebra booleana.

$$
\begin{aligned}
f_1 &= \overline{x}_1\overline{x}_2\overline{x}_3 + \overline{x}_1\overline{x}_2x_3 + \overline{x}_1x_2x_3 + x_1x_2x_3 \\
&= \overline{x}_1\overline{x}_2(\overline{x}_3 + x_3) + (\overline{x}_1 + x_1)x_2x_3 \\
&= \overline{x}_1\overline{x}_2 \cdot 1 + 1 \cdot x_2x_3 \\
&= \overline{x}_1\overline{x}_2 + x_2x_3
\end{aligned}
$$

Esempio (II)

La derivazione di una forma dall'altra può essere ottenuta applicando le regole dell'algebra booleana.

$$
\begin{aligned}
f_1 &= \overline{x}_1\overline{x}_2\overline{x}_3 + \overline{x}_1\overline{x}_2x_3 + \overline{x}_1x_2x_3 + x_1x_2x_3 \\
&= \overline{x}_1\overline{x}_2(\overline{x}_3 + x_3) + (\overline{x}_1 + x_1)x_2x_3 \\
&= \overline{x}_1\overline{x}_2 \cdot 1 + 1 \cdot x_2x_3 \\
&= \overline{x}_1\overline{x}_2 + x_2x_3
\end{aligned}
$$



Algebra booleana

È definita sui seguenti elementi ed operatori:

- ***elementi*: 0, 1**
- ***operatori*: AND (·), OR (+), NOT (¯)**
- ***assiomi*: (Postulati di Huntington)** **K={0,1}**
  - **chiusura:** $Ï \in Ç, Ī \in Ç \Rightarrow Ï Ī \in Ç, Ï + Ī \in Ç$
  - **identità: a+0=a, a · 1=a**
  - **commutatività: a+b=b+a, a · b=b · a**
  - **distributività: a(b+c)=a · b+a · c, a+(b · c)=(a+b) ·(a+c)**
  - **inverso:** $a\bar{a} = 0, a + \bar{a} = 1$

Leggi dell'algebra booleana

- **Associatività:**

$$a+(b+c)=(a+b)+c$$
$$a(bc)=(ab)c$$

- **Idempotenza:**

$$a+a=a$$
$$aa=a$$

- **De Morgan:**

$$\overline{I + \bar{I}} = \bar{I}\,\bar{\bar{I}}$$

$$\bar{I}\,\bar{\bar{I}} = \bar{I} + \bar{\bar{I}}$$

- **Involuzione:**

$$\bar{\bar{\bar{I}}} = \bar{I}$$

Minimizzazione

Dovendo implementare il circuito combinatorio corrispondente ad una tavola di verità data, è preferibile identificare quello con costo minimo, ad esempio in termini di minimo numero di porte.

Circuito combinatorio minimo

Un circuito combinatorio si dice *minimo* se:

- **realizza la funzione booleana corrispondente alla tavola di verità data**
- **soddisfa determinati vincoli quali:**
  - **la profondità del circuito è minore di un certo valore**
  - **il numero di porte utilizzate è minimo**
  - **il *fanin* massimo è minore di un certo valore**
  - **il *fanout* massimo è minore di un certo valore.**

Circuiti a 2 livelli

Una soluzione comune al problema del progetto dei circuiti combinatori minimi è rappresentata dai circuiti a 2 livelli.

Il metodo di Quine-McCluskey permette di passare dalla tavola di verità a un circuito minimo a 2 livelli.

Esso consiste nel trasformare la tavola di verità in una espressione booleana di uno dei 2 tipi seguenti:

- **somma di prodotti**

$$I(x_1, x_2, \ldots, x_n) = \sum_{i} x_{i1}\, x_{i2} \ldots x_{in}$$

- **prodotto di somme**

$$I(x_1, x_2, \ldots, x_n) = \prod_{i} x_{i1} + x_{i2} + \ldots + x_{in}$$

Circuiti a 2 livelli (II)

Se tutte le variabili di ingresso sono disponibili affermate o negate, tali espressioni sono direttamente trasformabili in circuiti a 2 livelli composti esclusivamente da porte AND e OR.

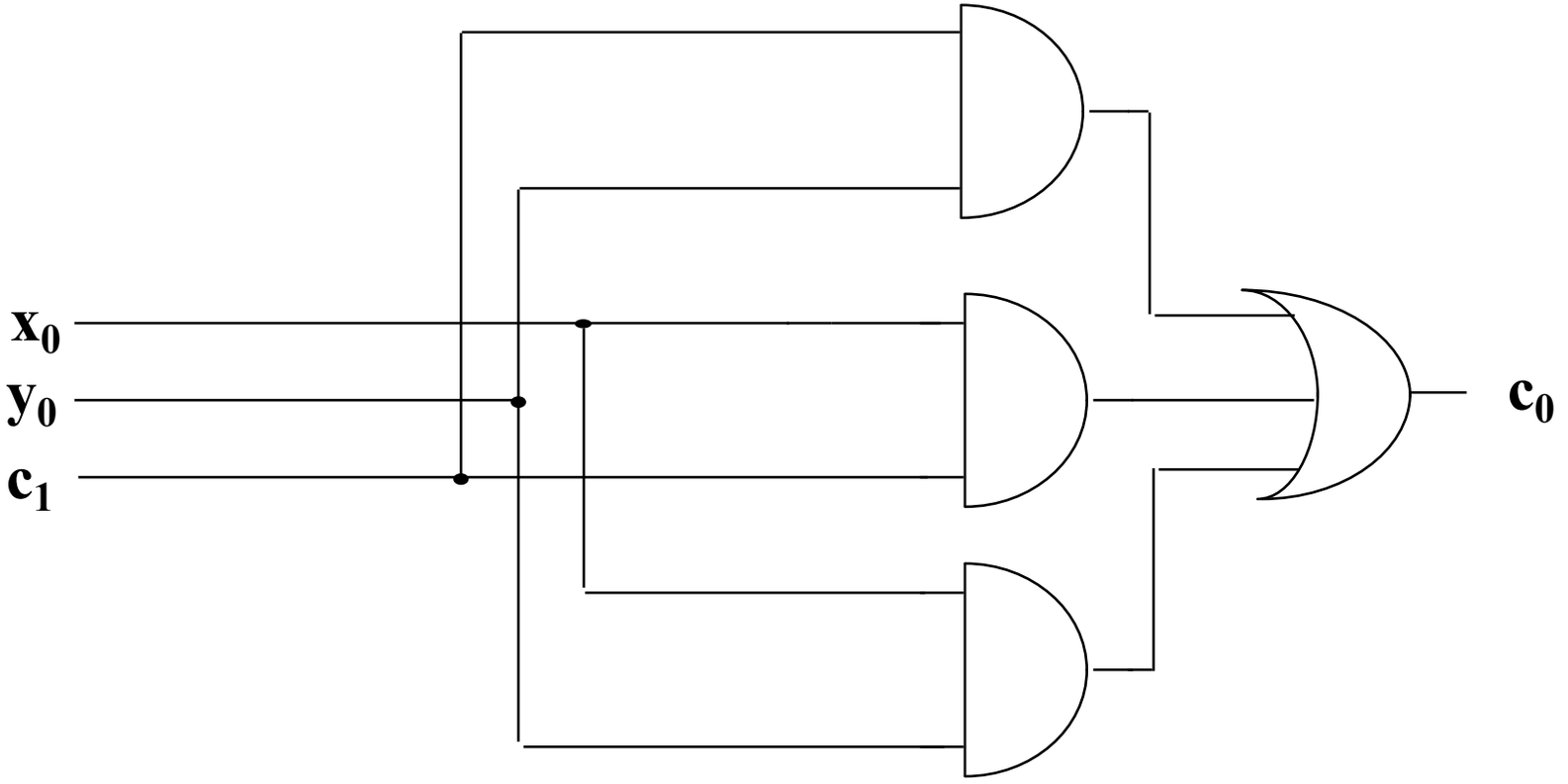
Esempio

La funzione svolta da un sommatore a 2 bit può essere scritta come

$$c_0 = x_0 c_1 + y_0 c_1 + x_0 y_0$$

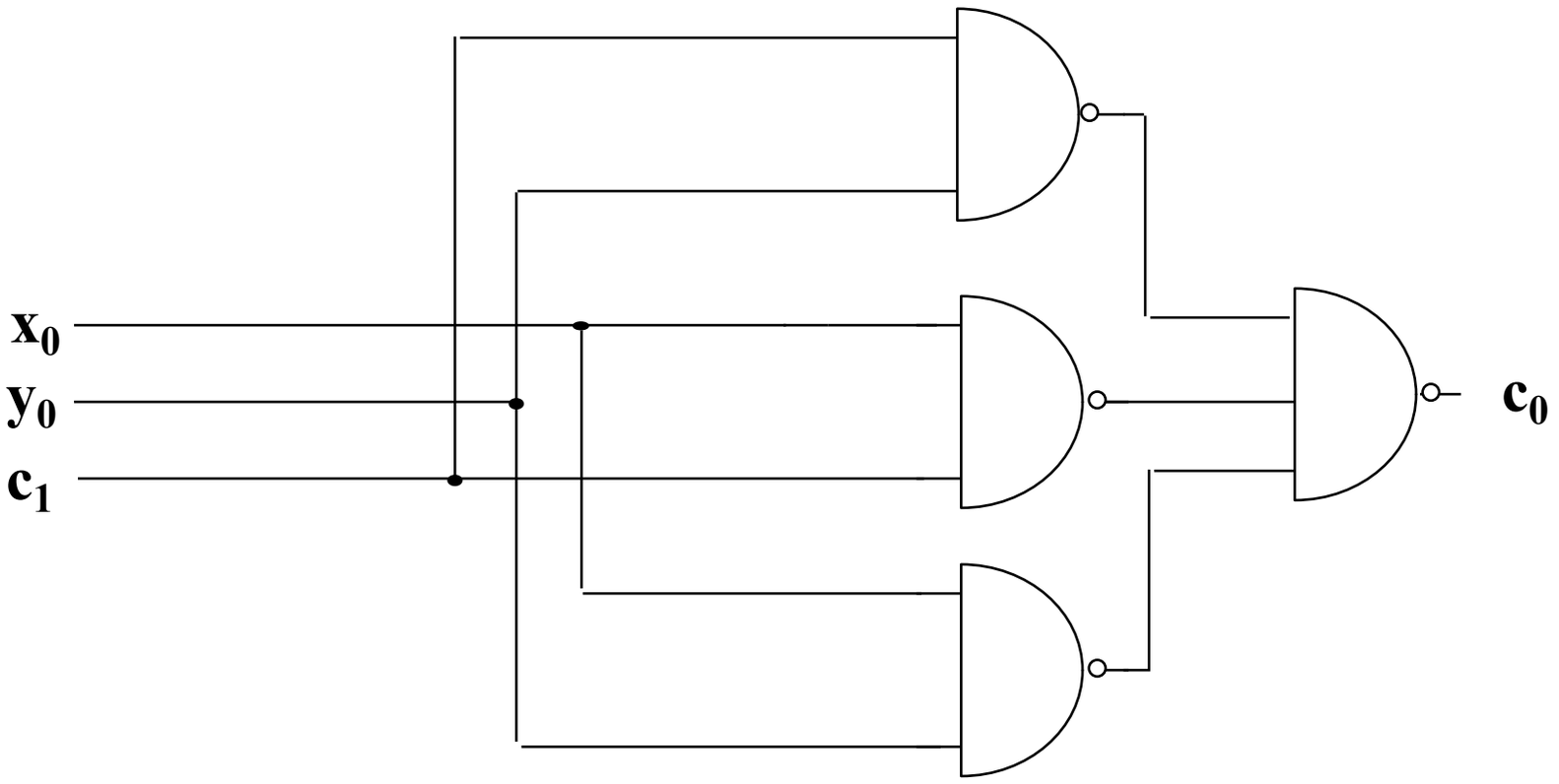
che si può trasformare nel circuito seguente:

Circuiti a 2 livelli (III)



$$c_0 = x_0 c_1 + y_0 c_1 + x_0 y_0$$

Circuiti a 2 livelli (IV)



Utilizzando la legge di De Morgan:

Canonicità

Qualsiasi funzione booleana possiede due possibili forme canoniche:

- **come *somma di prodotti*, ove tutti i prodotti sono cubi**
- **come *prodotto di somme*.**

Costruzione circuito minimo

Un circuito combinatorio si dice *minimo* se è vera una delle seguenti condizioni (tra loro equivalenti):

- **il circuito è composto dal minimo numero possibile di porte logiche**
- **il circuito implementa una funzione espressa in forma di somma di prodotti, nella quale**
  - **il numero di prodotti è minimo**
  - **nessun letterale può esssere cancellato da un prodotto senza cambiare la funzione.**

Valgono analoghe condizioni relative alla forma prodotto di somme.

Costruzione circuito minimo (II)

Il problema della costruzione del circuito combinatorio minimo a partire dalle specifiche ha complessità NP.

In pratica può essere risolto con strumenti automatici (ad es. *ESPRESSO*), con tempi di calcolo anche significativi.

Per circuiti combinatori di piccole dimensioni (fino a 5 variabili di ingresso) il metodo delle *Mappe di Karnaugh* permette di passare dalla tavola della verità al circuito minimo.

Mappe di Karnaugh

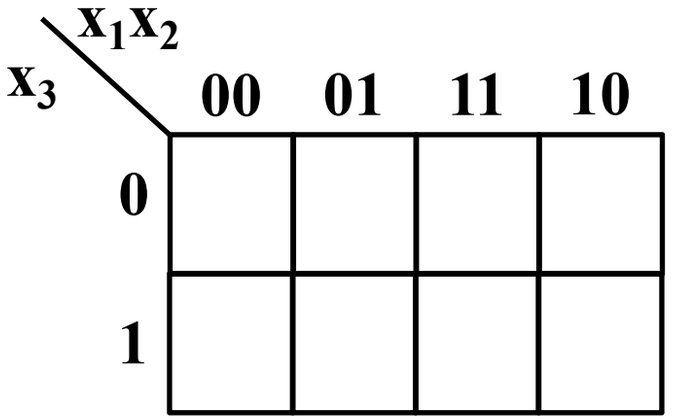
- **Una mappa di Karnaugh è una rappresentazione grafica del comportamento di un sistema combinatorio**
- **Tale rappresentazione è utilizzabile per funzioni con una sola uscita e un numero di ingressi fino a 4**
- **Utilizzando le mappe di Karnaugh è possibile identificare la funzione booleana minima per il sistema, e di qui realizzare il circuito minimo corrispondente.**

Mappe di Karnaugh - Struttura

Esempio

x_1	x_2	x_3	f_1
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Tavola di verità

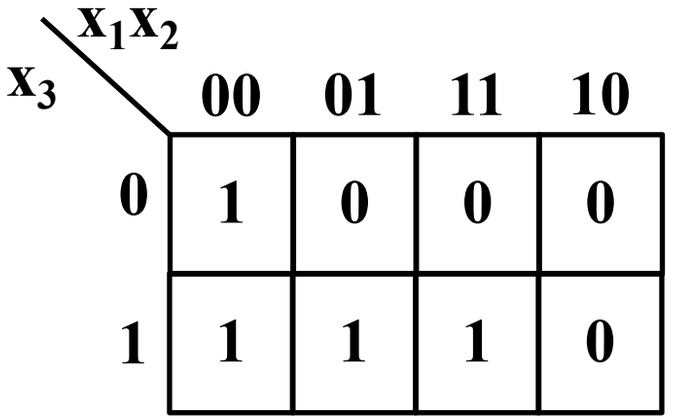


Mappa di Karnaugh

Esempio

x_1	x_2	x_3	f_1
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Tavola di verità

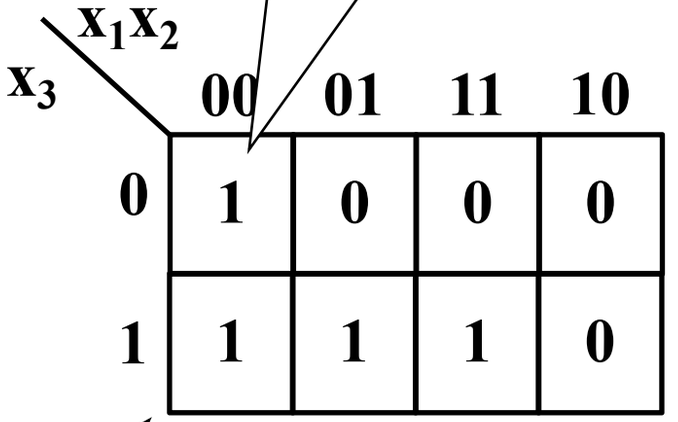


Mappa di Karnaugh

Esempio

x_1	x_2	x_3	f_1
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Tavola d ap



Ogni casella a 1 corrisponde a un minterm

Caselle a 1 adiacenti corrispondono ad un cubo

L'ordine delle colonne e delle righe non è casuale!

Esempio

Ogni casella a 1 corrisponde ad un minterm

$\overline{x_1}\ \overline{x_2}\ \overline{x_3}$

			f_1
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

x_3 \ x_1x_2	00	01	11	10
0	1	0	0	0
1	1	1	1	0

T

$\overline{x_1}\ \overline{x_2}x_3$

$\overline{x_1}x_2x_3$

li

$x_1x_2x_3$

Esem

Ogni coppia di caselle a 1 adiacenti corrisponde ad un cubo

$\overline{x_1}\,\overline{x_2}$

			f_1
0	0	0	
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

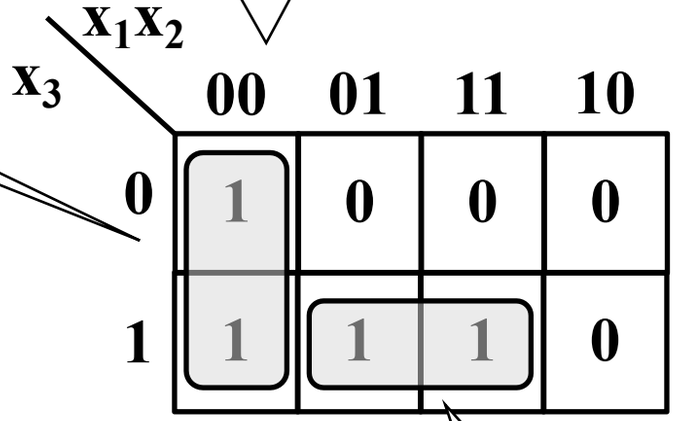


Tavola di verità

Mappa di

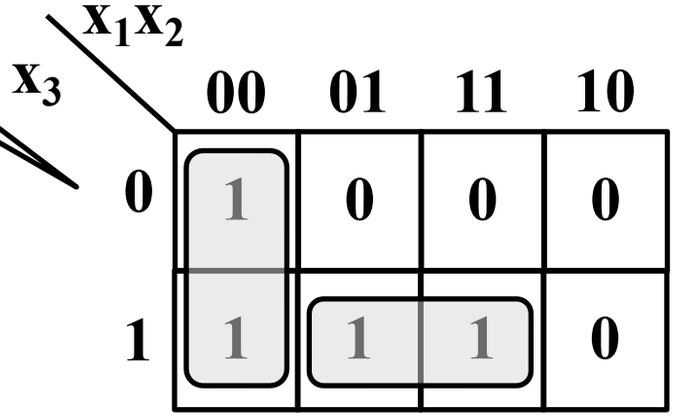
x_2x_3

Esempio

$$f_1 = \overline{x_1}\,\overline{x_2} + x_2 x_3$$

x_1	x_2	x_3	f_1
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Tavola di verità

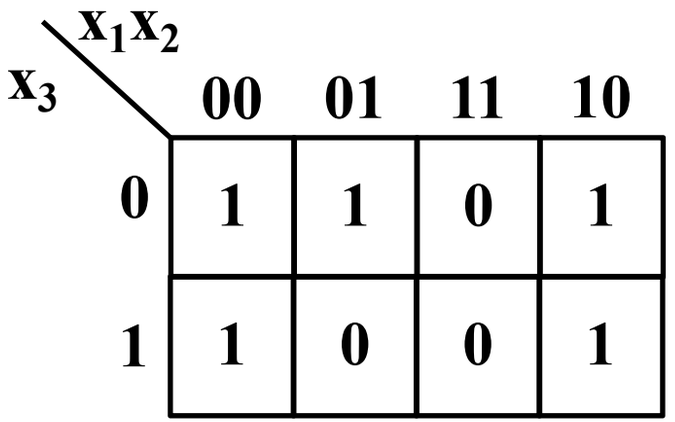


Mappa di Karnaugh

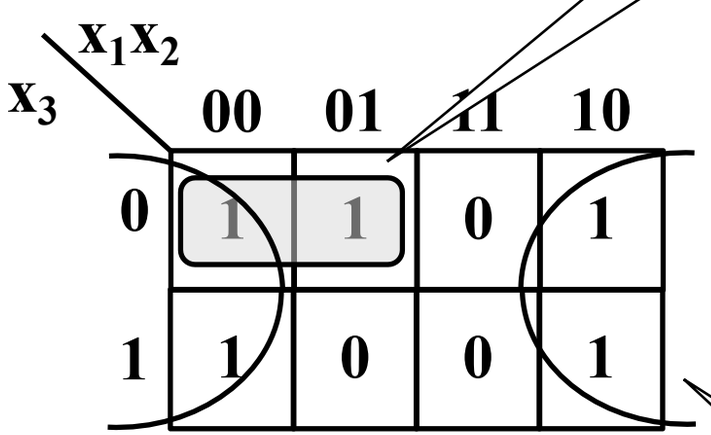
Procedura di sintesi

- **Una volta costruita la mappa di Karnaugh corrispondente alla tabella di verità, si deve**
  - **Identificare il minimo insieme di cubi che coprono tutti gli 1 nella mappa di Karnaugh, scegliendo quelli di dimensione massima; i cubi possono eventualmente sovrapporsi**
  - **Trasformare i cubi nella corrispondente espressione in forma di somma di prodotti**
  - **Costruire il corrispondente circuito.**

Esempio



Esempio



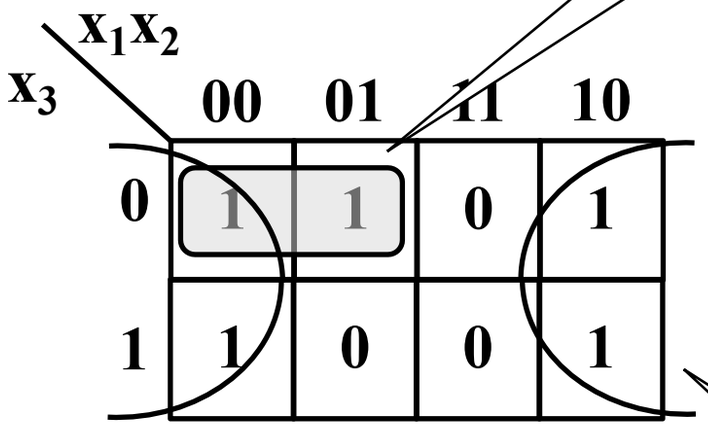
$\overline{x_1 x_3}$

$\overline{x_2}$

Esempio

$f_1 = \overline{x_1}\,\overline{x_3} + \overline{x_2}$

$\overline{x_1}\,\overline{x_3}$



$\overline{x_2}$

Mappe di Karnaugh con 4 ingressi

- **Sono composte da 16 caselle**
- **In esse possono essere individuati cubi da 1, 2, 4, 8 elementi.**

Mappe di Karnaugh con 4 ingressi
Struttura

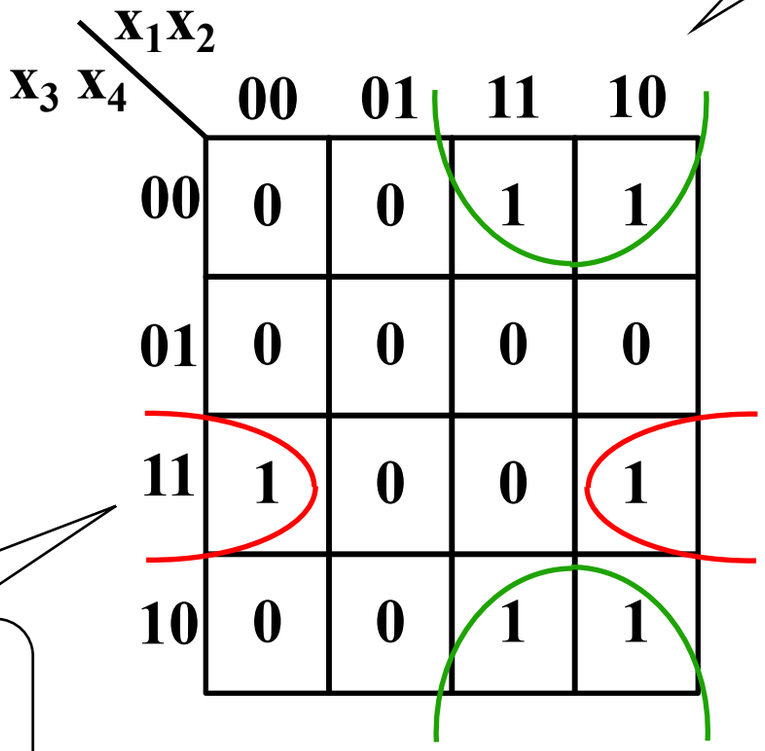
Esempio

$x_3 x_4$ \ $x_1 x_2$	00	01	11	10
00	0	0	1	1
01	0	0	0	0
11	1	0	0	1
10	0	0	1	1

Esempio

$f_1 = x_1\,\overline{x_4} + \overline{x_2}x_3x_4$

$x_1\overline{x_4}$



$\overline{x_2}x_3x_4$

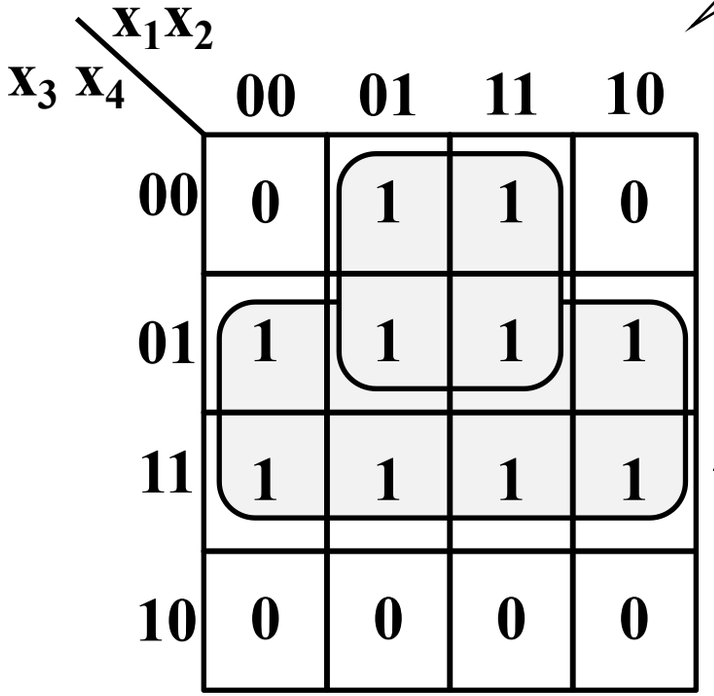
Esempio 2

$x_3 x_4$ \ $x_1 x_2$	00	01	11	10
00	0	1	1	0
01	1	1	1	1
11	1	1	1	1
10	0	0	0	0

Esempio 2

$f_1 = x_2\overline{x_3} + x_4$

$x_2\overline{x_3}$



x_4

Don’t care

- **In alcuni casi le specifiche per un sistema combinatorio possono non prevedere alcun valore in uscita in corrispondenza di alcune combinazioni di ingresso, ad esempio perché queste non si presentano mai**
- **In tal caso si parla di valori «don’t care»**
- **Il processo di sintesi può sfruttare questi valori per minimizzare il circuito.**

Esempio

- **Consideriamo una funzione che riceve in ingresso i 4 bit corrispondenti a una cifra BCD e ritorna il valore 1 se e solo se tale cifra corrisponde a un multiplo di 3:**

cifra decimale	codifica binaria #	b_3	b_2	b_1	b_0	f
0	0	0	0	0	0	0
1	1	0	0	0	1	0
2	2	0	0	1	0	0
3	3	0	0	1	1	1
4	4	0	1	0	0	0
5	5	0	1	0	1	0
6	6	0	1	1	0	1
7	7	0	1	1	1	0
8	8	1	0	0	0	0
9	9	1	0	0	1	1
non usate	10	1	0	1	0	x
	11	1	0	1	1	x
	12	1	1	0	0	x
	13	1	1	0	1	x
	14	1	1	1	0	x
	15	1	1	1	1	x

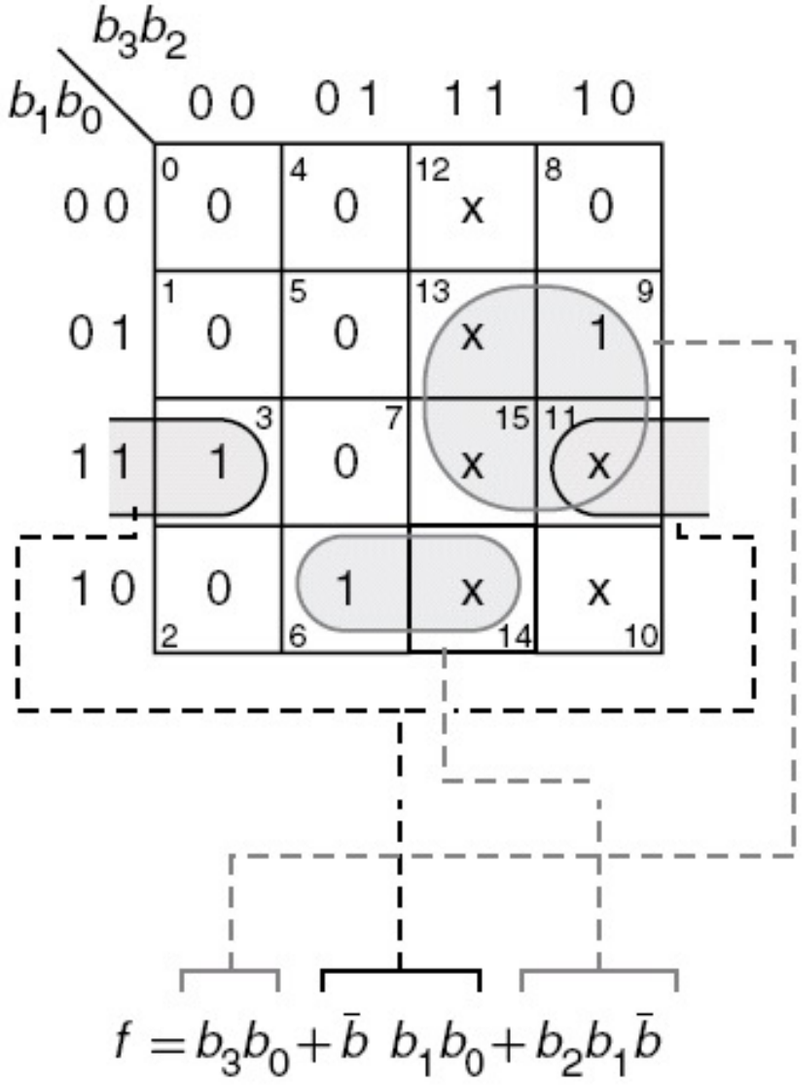
Esempio

- **Consideriamo una funzione che riceve in ingresso i 4 bit corrispondenti a una cifra BCD e ritorna il valore 1 se e solo se tale cifra corrisponde a un multiplo di 3:**

cifra decimale	codifica binaria #	b_3	b_2	b_1	b_0	f
0	0	0	0	0	0	0
1	1	0	0	0	1	0
2	2	0	0	1	0	0
3	3	0	0	1	1	1
4	4	0	1	0	0	0
5	5	0	1	0	1	0
6	6	0	1	1	0	1
7	7	0	1	1	1	0
8	8	1	0	0	0	0
9	9	1	0	0	1	1
non usate	10	1	0	1	0	x
	11	1	0	1	1	x
	12	1	1	0	0	x
	13	1	1	0	1	x
	14	1	1	1	0	x
	15	1	1	1	1	x

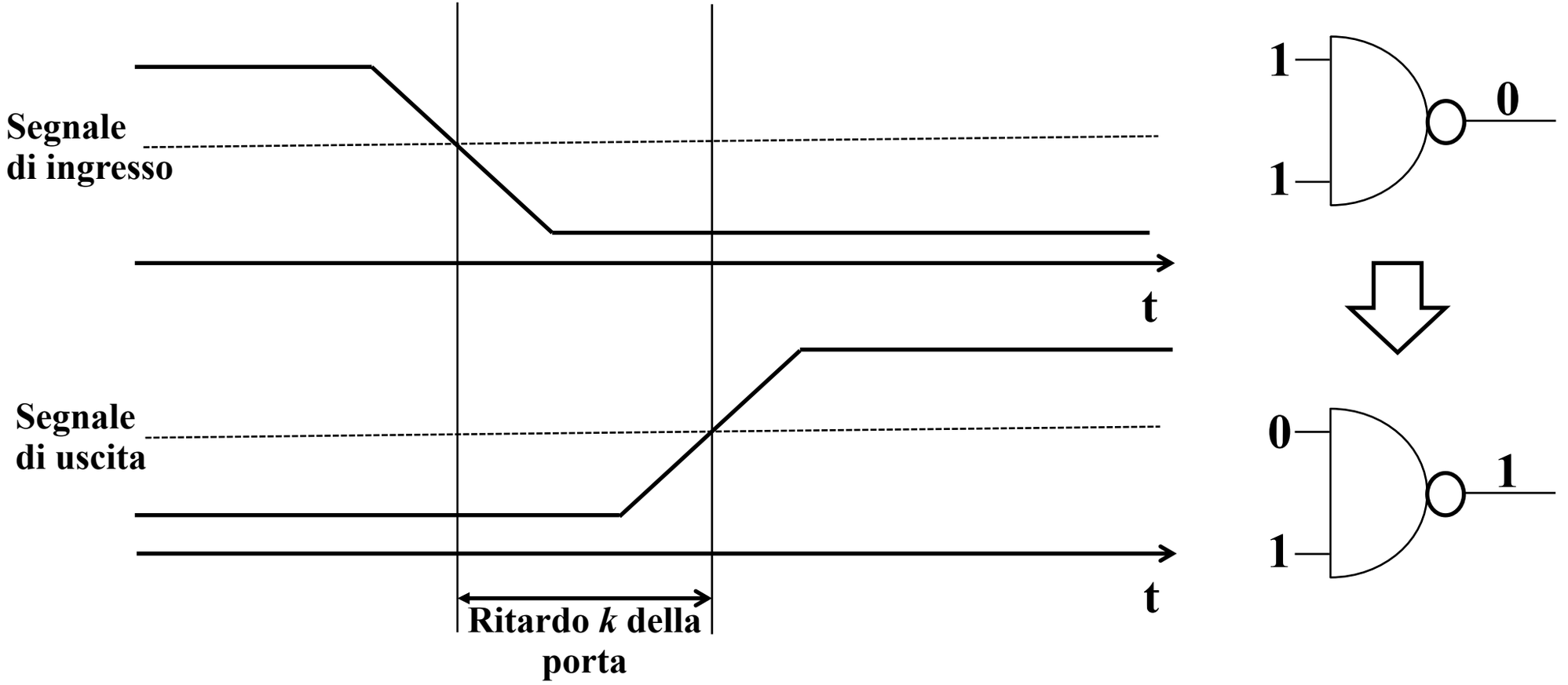
Esempio

- **La corrispondente mappa di Karnaugh include 6 caselle con il valore don’t care**
  - **Esse possono essere considerate come se contenessero il valore 0 o 1, a seconda di cosa è più conveniente per il progetto.**



Ritardi

L'uscita di una porta logica assume il valore corretto solo dopo che è trascorso un certo tempo *k* dalla stabilizzazione dei segnali di ingresso.



Ritardo associato a un circuito

- **Dato un circuito combinatorio, a fronte di un cambiamento nel valore degli ingressi esso richiede un certo tempo prima di produrre il valore di uscita corrispondente**
- **È buona norma non applicare un nuovo cambiamento sino a che la propagazione degli effetti del precedente non si è completamente esaurita**
- **La conoscenza del tempo massimo necessario al circuito per assumere la nuova configurazione di uscita è cruciale per sapere con quale frequenza massima si possono applicare i vettori agli ingressi del circuito.**

Ritardo associato a un circuito

- **Conoscendo i ritardi associati alle porte logiche componenti è possibile calcolare il tempo massimo di risposta di un circuito combinatorio**
- **Tale tempo dipende dal numero massimo di porte appartenenti ad uno stesso cammino dagli ingressi alle uscite che devono commutare al mutare del vettore di ingresso**
- **Assumendo che i ritardi delle porte siano uguali, tale tempo è proporzionale alla *profondità* del circuito, ossia al massimo numero di porte che si incontrano lungo un qualsiasi cammino da un ingresso a un'uscita.**

Cammino critico

Il cammino lungo il quale il ritardo con il quale una variazione in ingresso si propaga sulle uscite è massimo si definisce *cammino critico* (*critical path*).

La lunghezza del cammino critico determina la massima frequenza con cui si possono applicare i vettori di ingresso al circuito.

Ridurre la lunghezza del cammino critico permette quindi di migliorare le prestazioni (in termini di velocità) dei circuiti.

Cammino critico

Tale lunghezza è pari alla somma dei ritardi associati alle porte che compongono il cammino

Il cammino lungo il quale il ritar[illegible]ngresso si propaga sulle uscite è massimo [illegible]misce *cammino critico* (*critical path*).

La lunghezza del cammino critico determina la massima frequenza con cui si possono applicare i vettori di ingresso al circuito.

Ridurre la lunghezza del cammino critico permette quindi di migliorare le prestazioni (in termini di velocità) dei circuiti.

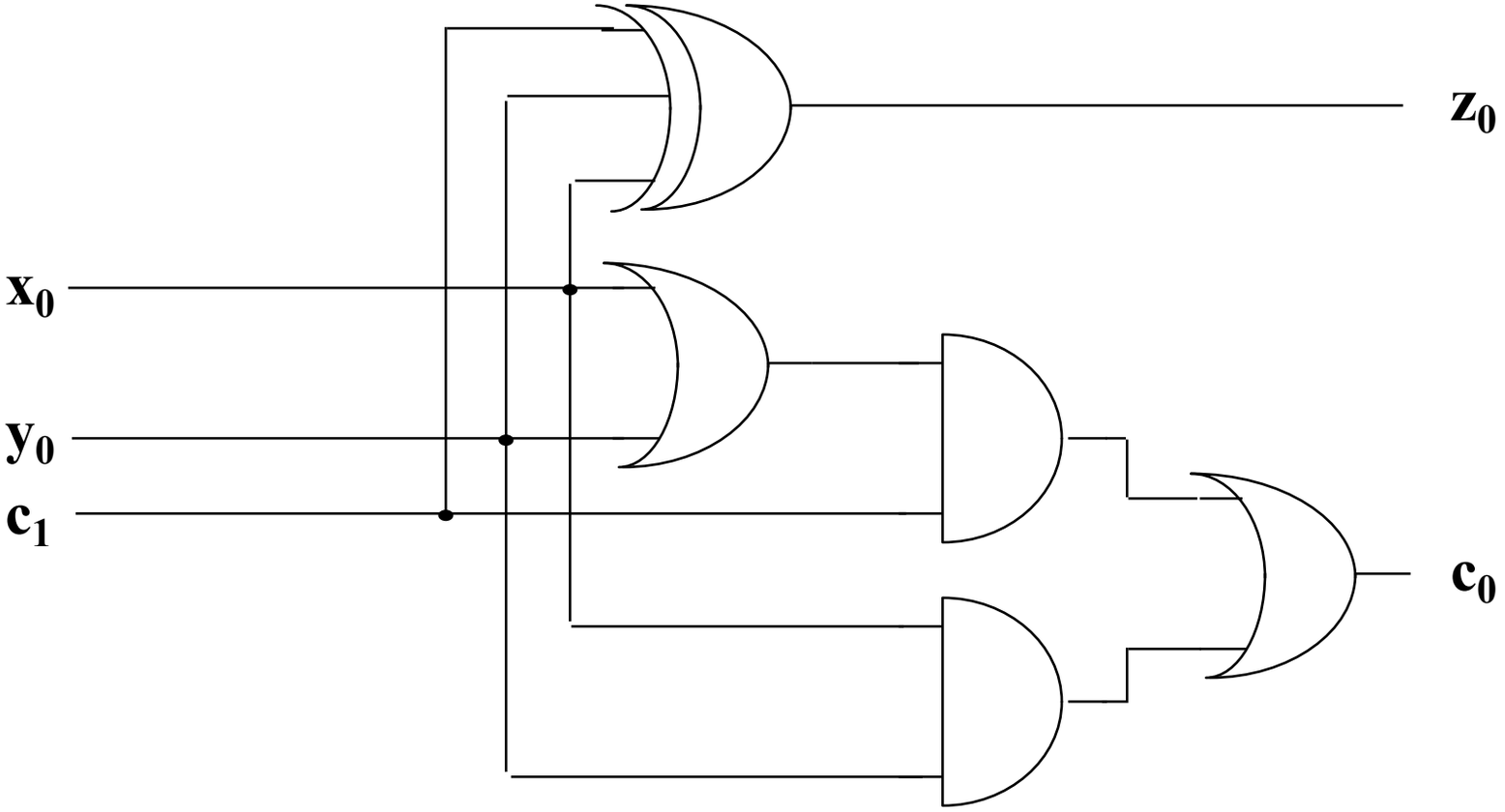
Livelli

Ad ogni porta logica in un circuito può essere assegnato un *livello* nella seguente maniera:

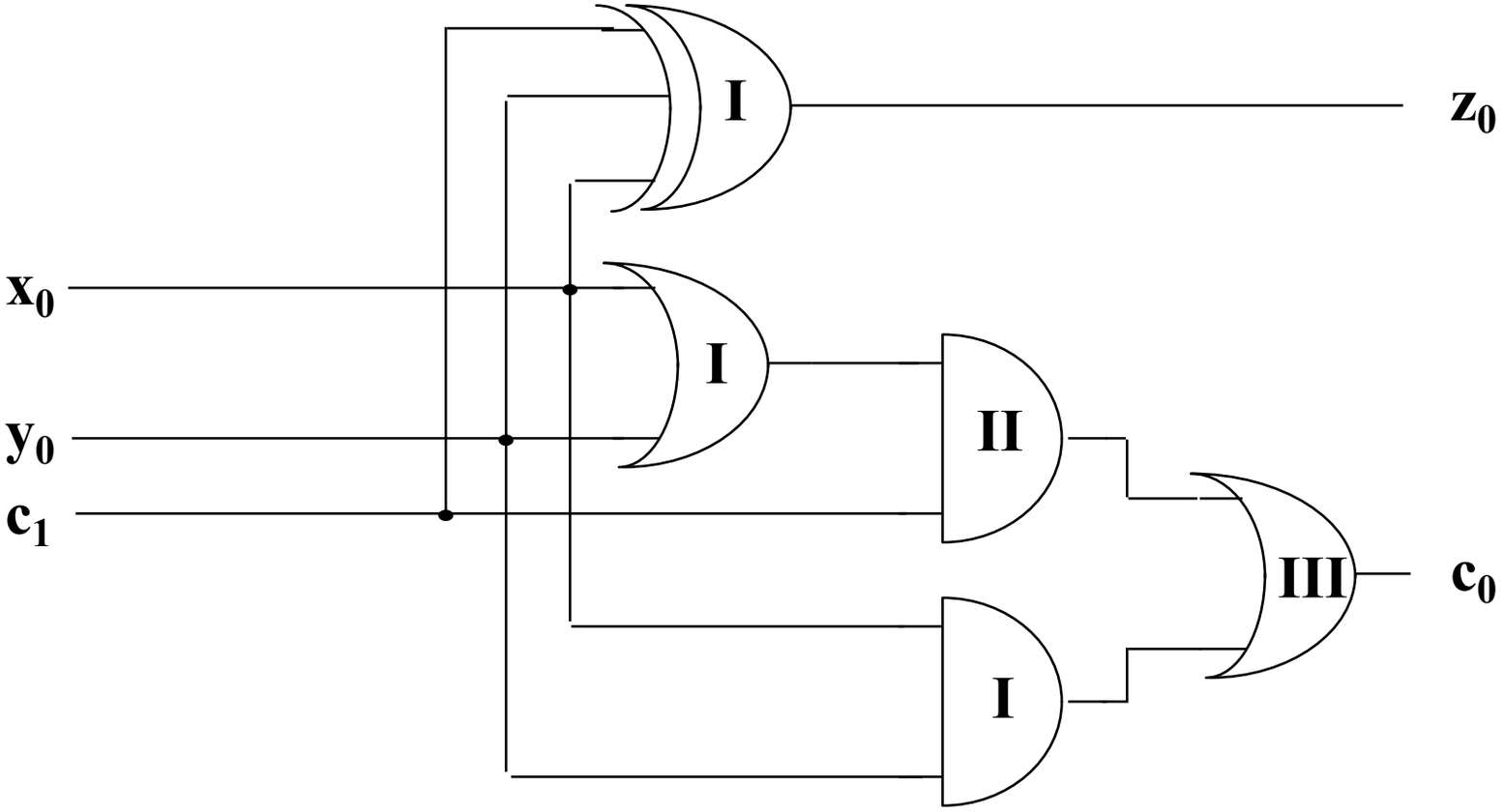
- **il livello delle porte che hanno come ingressi solo linee di ingresso è 1**
- **il livello delle altre porte è pari al livello della porta di ingresso avente il livello massimo, più 1.**

La *profondità* di un circuito è pari al livello della porta di livello massimo. Questa alimenterà sicuramente una linea di uscita.

Livelli: esempio



Livelli: esempio

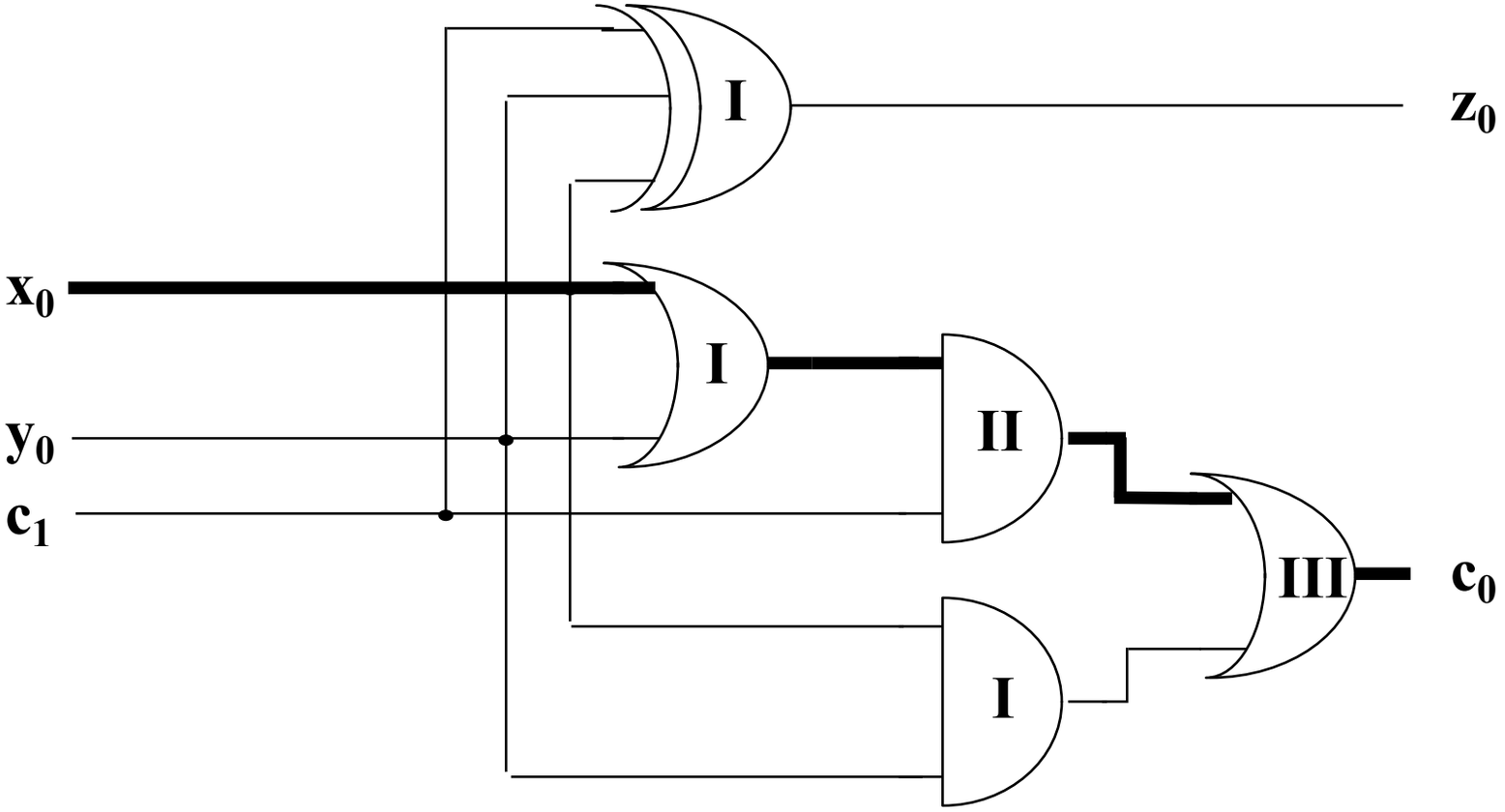


Calcolo del cammino critico

Se le porte hanno tutte lo stesso ritardo, il cammino da un ingresso a un'uscita composto dal massimo numero di porte è il cammino critico.

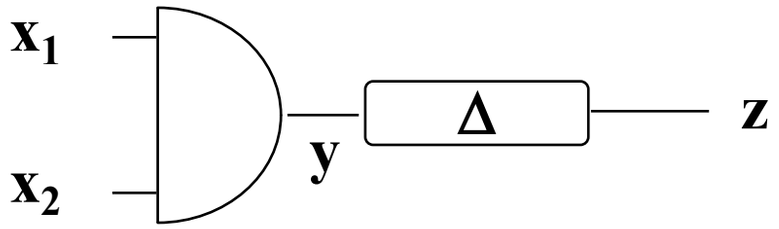
Se le porte hanno ritardi diversi il calcolo del cammino critico è più complesso.

Cammino critico



Circuiti sequenziali

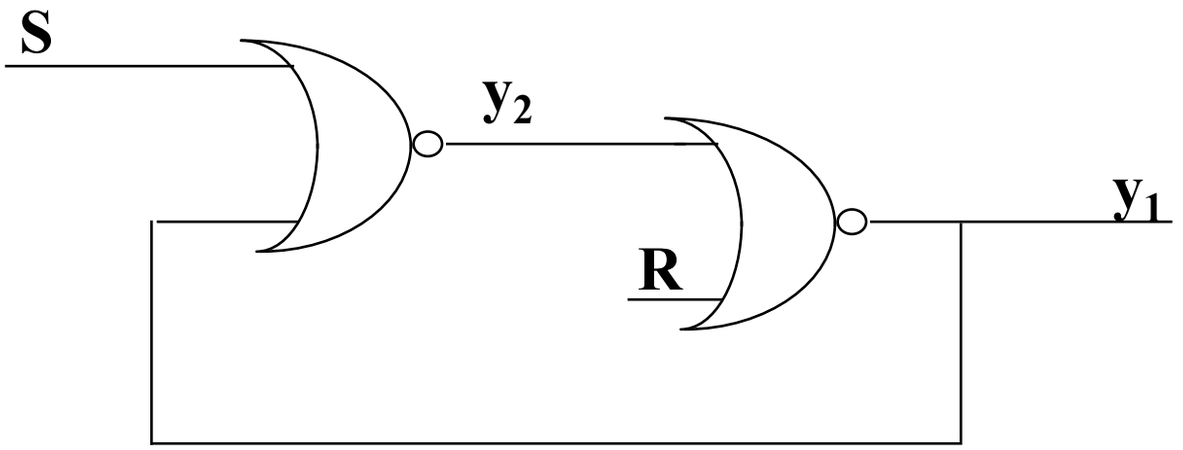
- **Implementano funzioni dipendenti dal tempo**
- **Sono in grado di memorizzare informazioni**
- **Sfruttano i ritardi delle porte:**



$$z(t+\Delta)=x_1(t)x_2(t)$$

Flip-Flop

È in grado di mantenere l'informazione per un periodo illimitato di tempo.

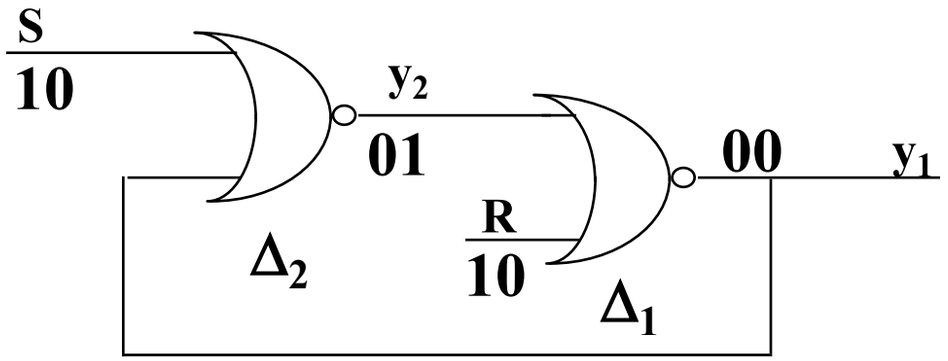


Comportamento:

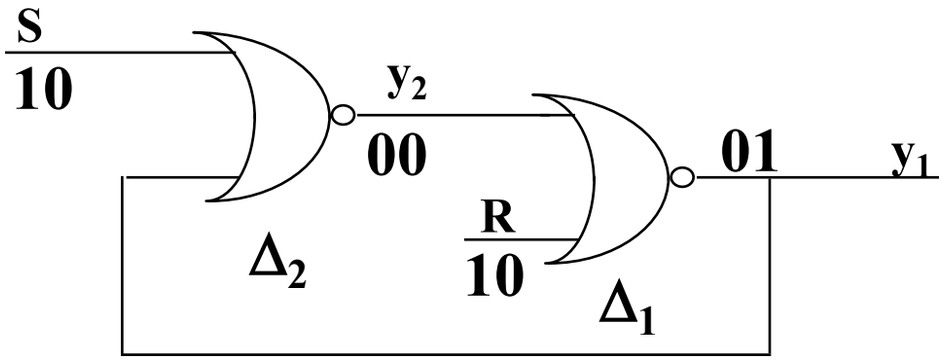
- **S=R=0: mantiene il valore su y_1**
- **S=0, R=1: forza uno 0 su y_1**
- **S=1, R=0: forza un 1 su y_1**
- **S=1, R=1: configurazione vietata**

Transizione S=R=1⇒ S=R=0

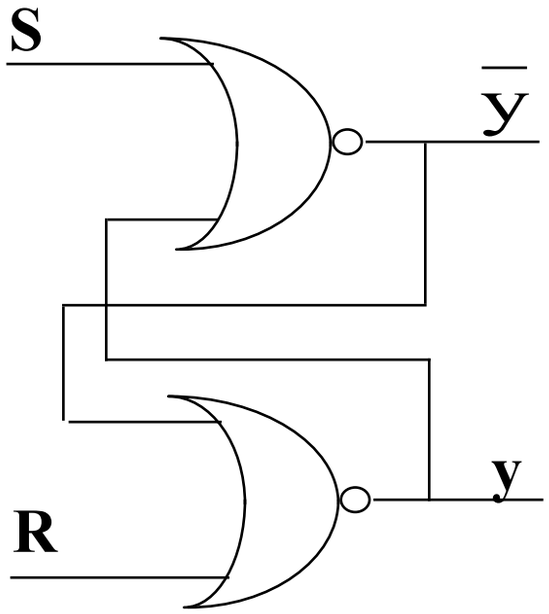
$\Delta_2 < \Delta_1$



$\Delta_2 > \Delta_1$

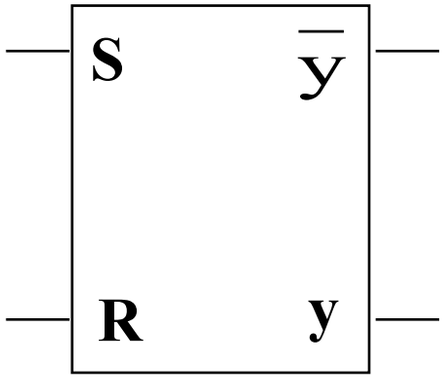


Flip-Flop SR asincrono



Schema

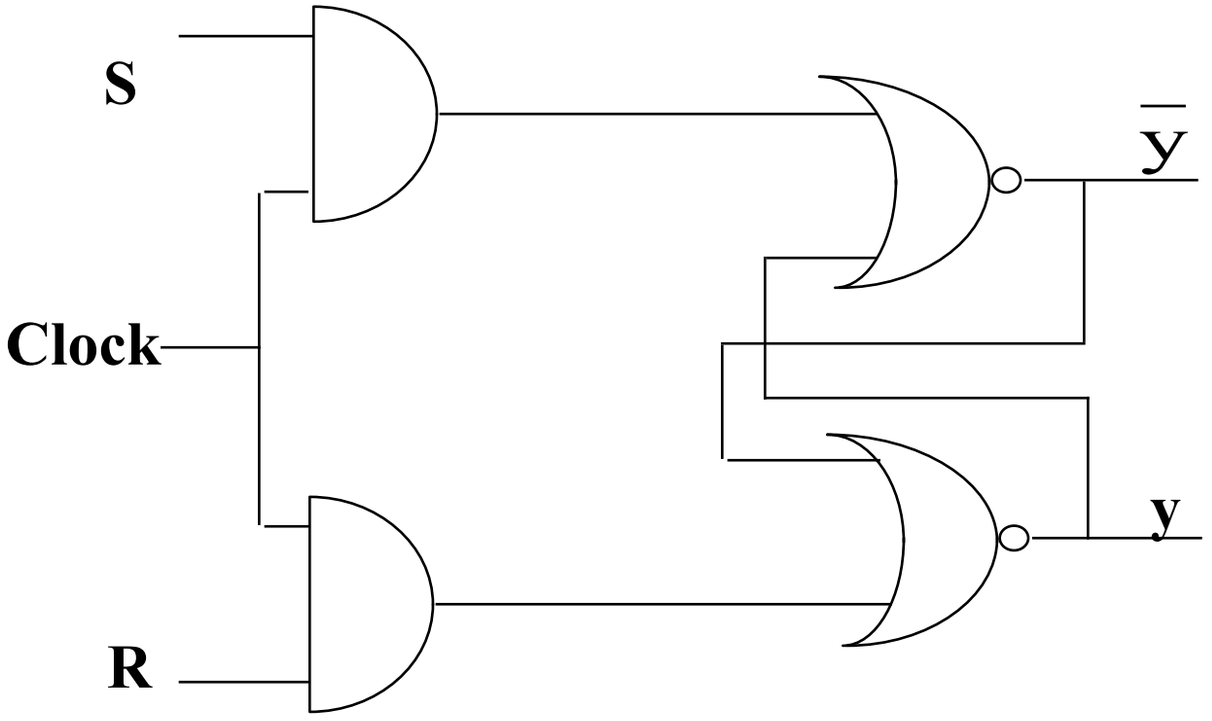
Simbolo



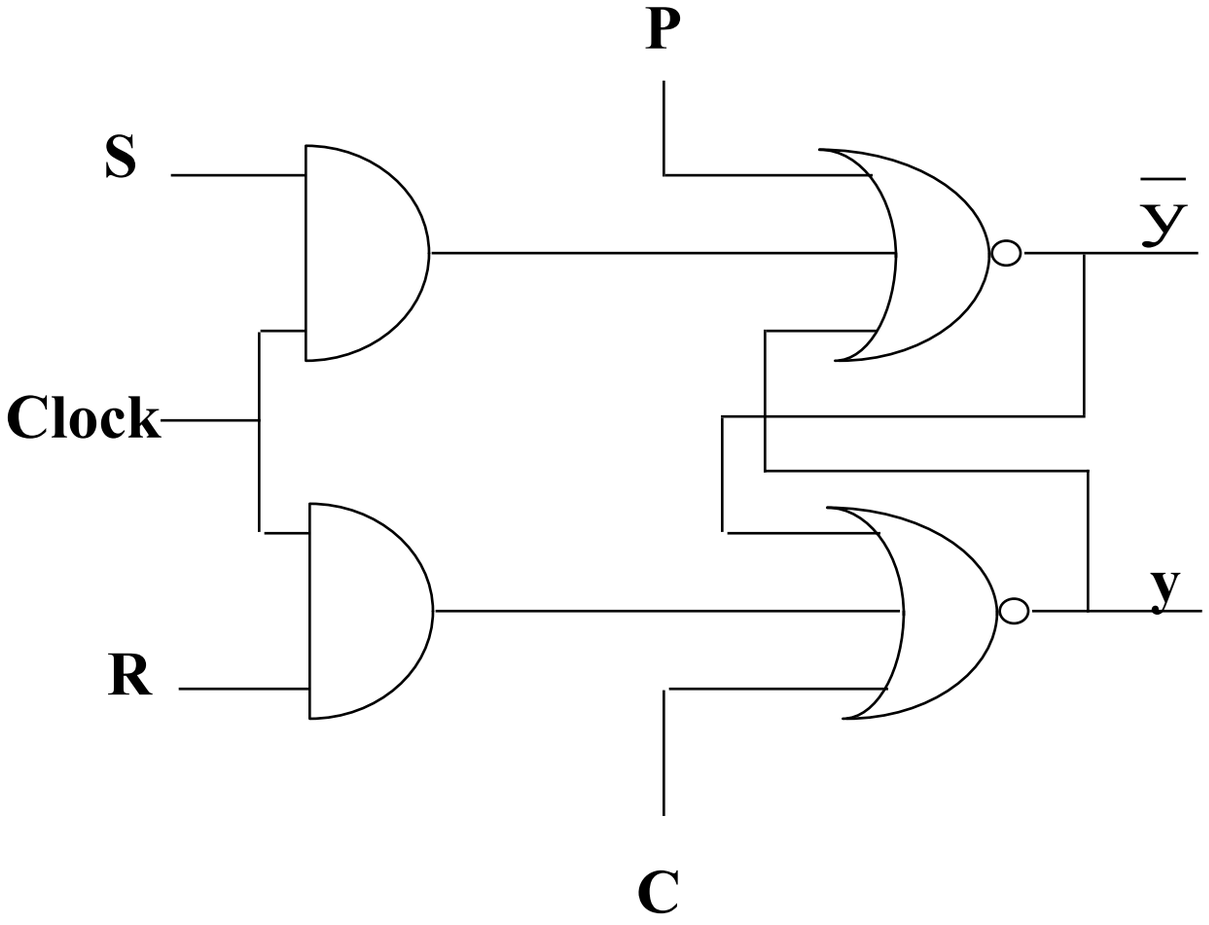
y \ SR	00	01	10
0	0	0	1
1	1	0	1

Tavola degli stati

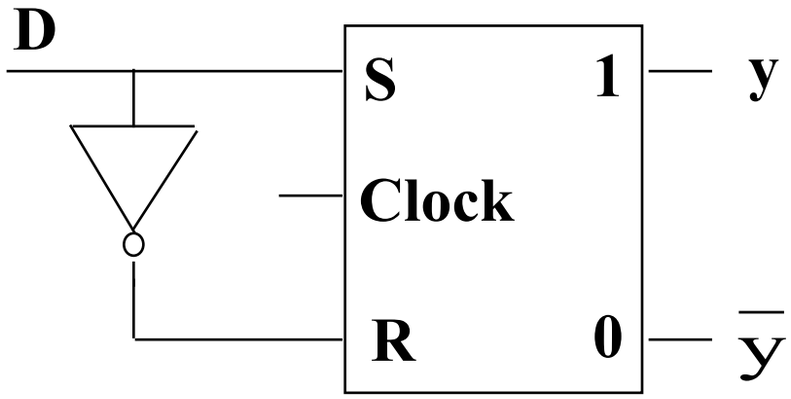
Flip-Flop sincrono



Flip-Flop SR con Clock, Preset e Clear asincroni



Flip-Flop D



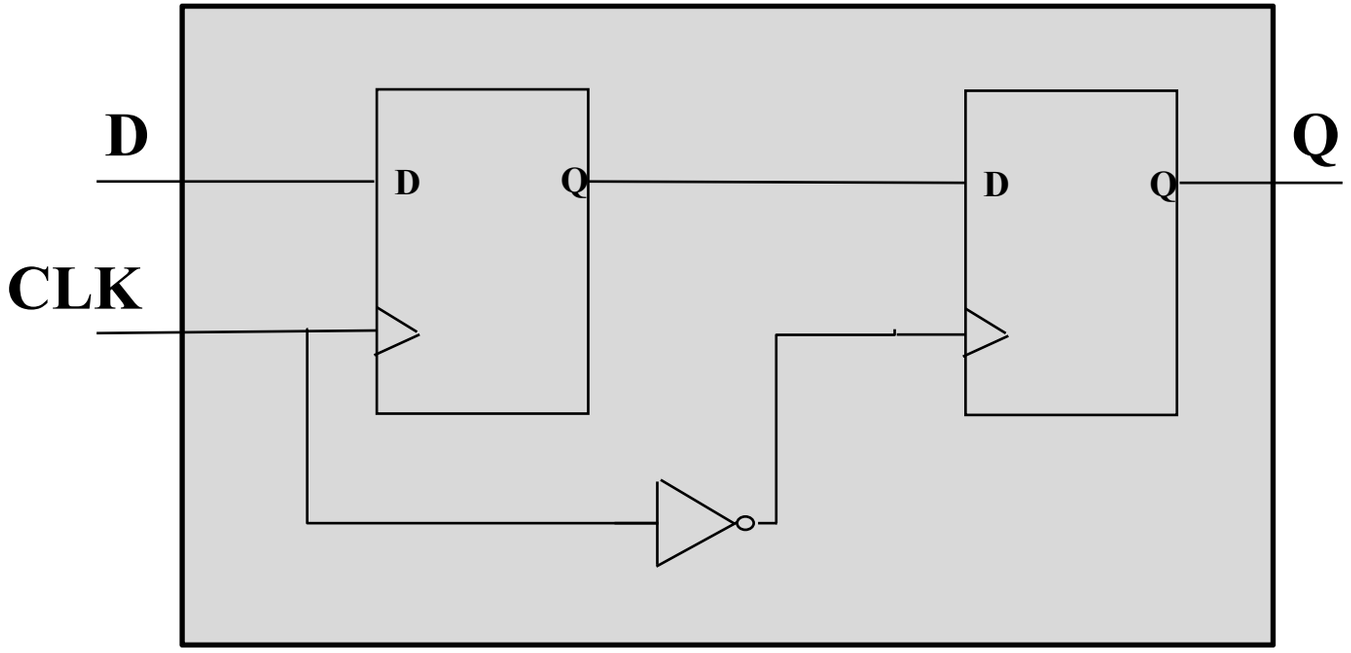
y \ D	0	1
0	0	1
1	0	1

Tavola degli stati

Quando Clock=1 viene memorizzato il valore di D.

Flip-Flop Master-Slave

In alcuni casi (quando non si può controllare il duty cycle di Clock) è utile utilizzare una versione di Flip-Flop in cui la transizione può avvenire solo sul fronte di salita (o di discesa) di Clock.

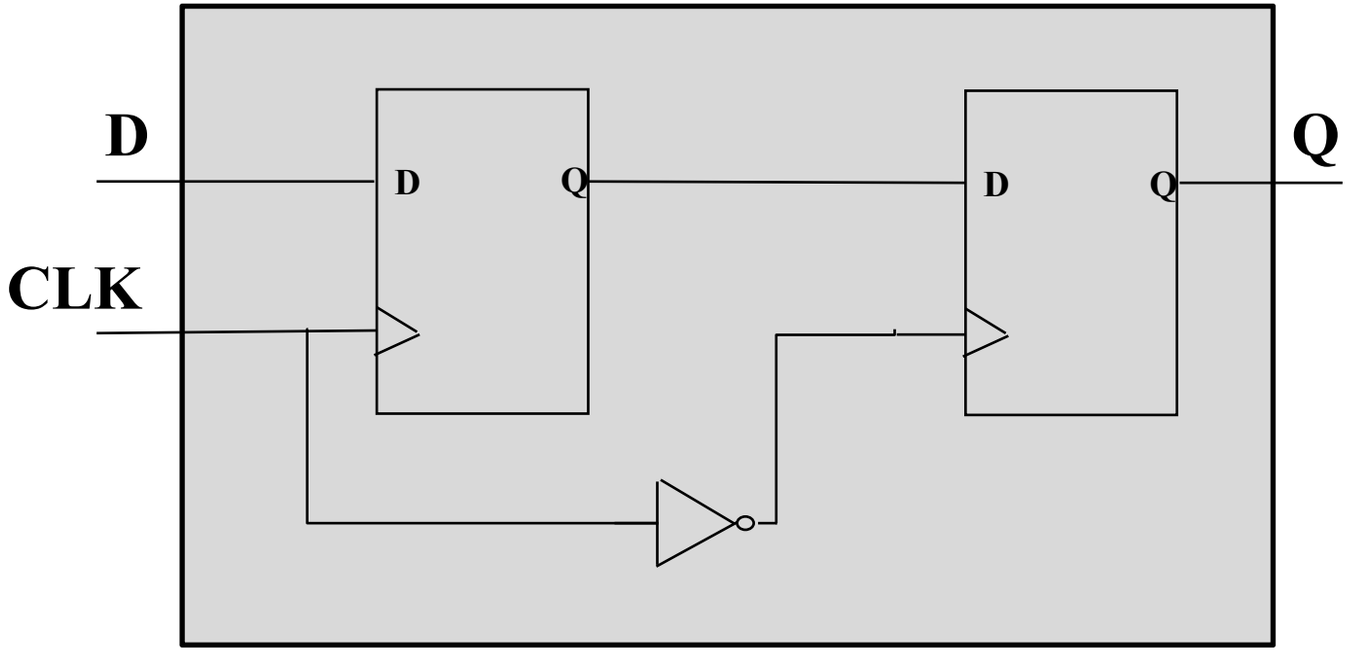


Flip-Flop

L'uscita del circuito in figura commuta sul fronte di discesa di Clock.
Collegando l'ingresso di Clock dei due Flip-Flop a CLK in modo opposto si può ottenere la versione che commuta sul fronte di salita.

In alcu[...] è utile utilizza[...]op in cui la transizione può avvenire solo sul fronte [...]lock.

I due Flip-Flop prendono il nome di Master e Slave, rispettivamente.



Circuiti sincroni

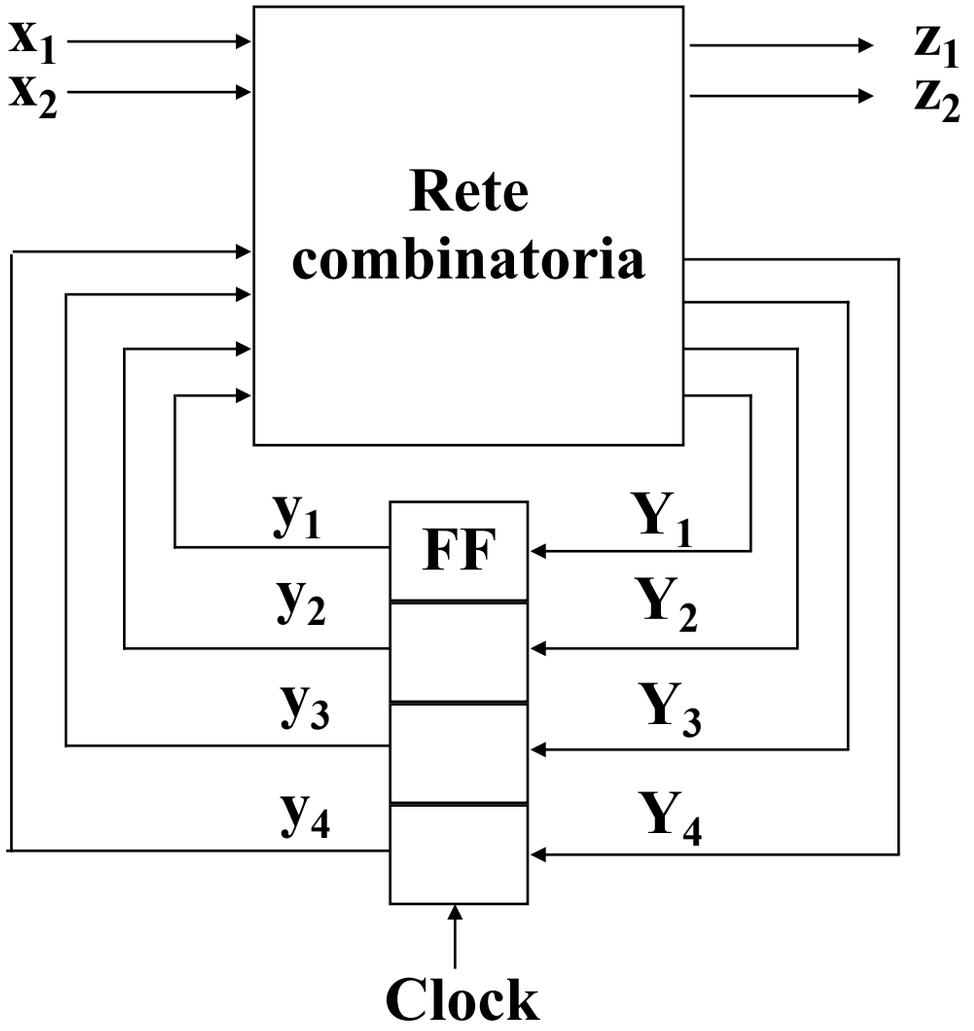
Il funzionamento di un circuito sequenziale dipende dai valori relativi dei ritardi delle porte logiche nella rete combinatoria (*corse* o *race*).

Questo può essere evitato facendo in modo che la memorizzazione dei valori in ingresso avvenga per tutti i Flip-Flop in uno stesso istante, determinato dal valore di un segnale comune (*clock*).

Tale segnale ha un andamento ad onda quadra con una certa frequenza.

I circuiti dotati di clock si dicono *sincroni*.

Modello di Huffman



Frequenza di funzionamento

Per garantire il corretto funzionamento di un circuito sequenziale sincrono è necessario che il periodo del segnale di clock sia maggiore del ritardo massimo attraverso la rete combinatoria.

Tabella degli stati

Un circuito sequenziale sincrono evolve attraverso *stati*, determinati dai valori presenti nei Flip-Flop.

Il suo comportamento può quindi essere descritto attraverso una *tabella degli stati*, che descrive le transizioni tra stati in funzione del valore sugli ingressi.

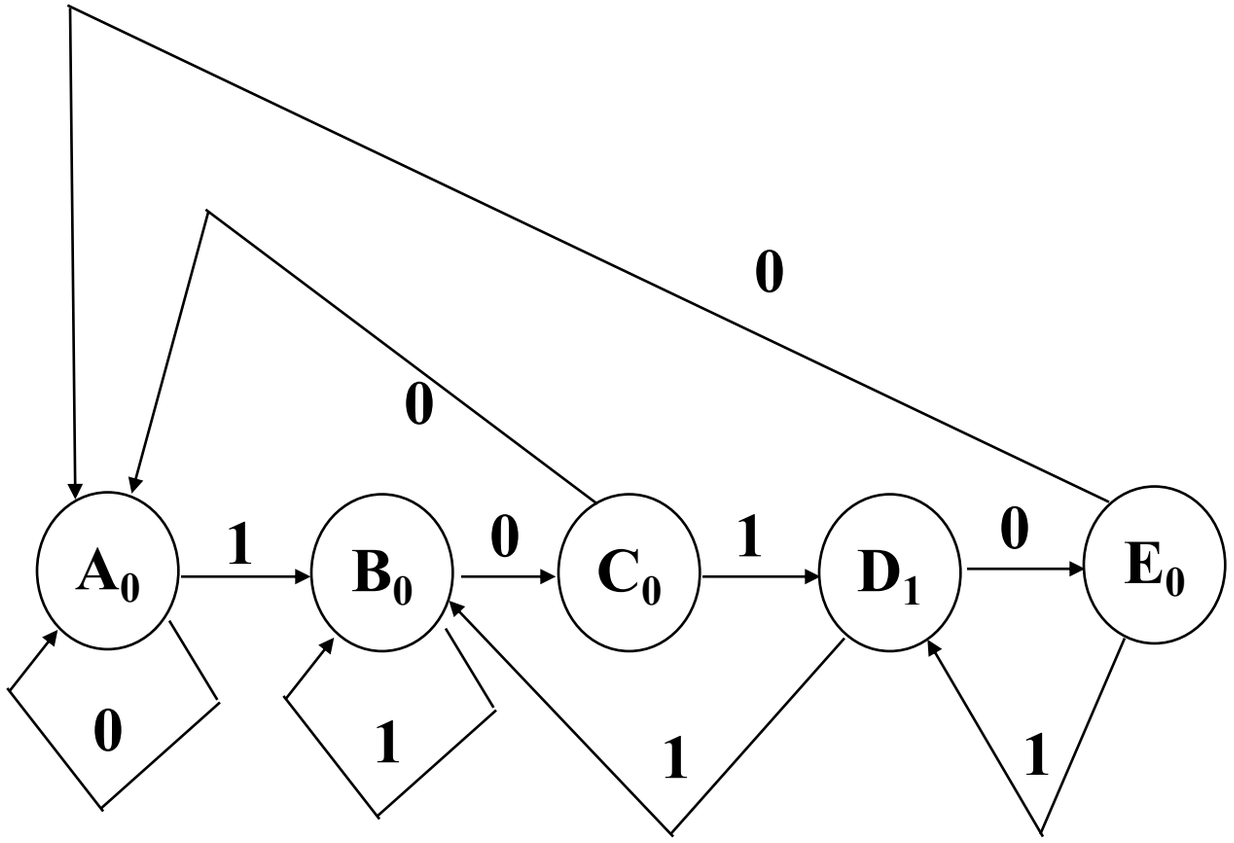
Progetto di circuiti sequenziali sincroni

- **Costruzione della tavola degli stati**
- **Minimizzazione (eliminazione degli stati equivalenti)**
- **Assegnazione degli stati**
- **Costruzione della tavola della verità della rete combinatoria**
- **Sintesi della rete combinatoria.**

Esempio

Consideriamo il riconoscitore della sequenza 1-0-1.

Il diagramma degli stati è



Svolgimento

Consideriamo il riconoscitore della sequenza 1-0-1.

Il diagramma degli stati:

Svolgimento

Consideriamo il riconoscitore della sequenza 1-0-1.

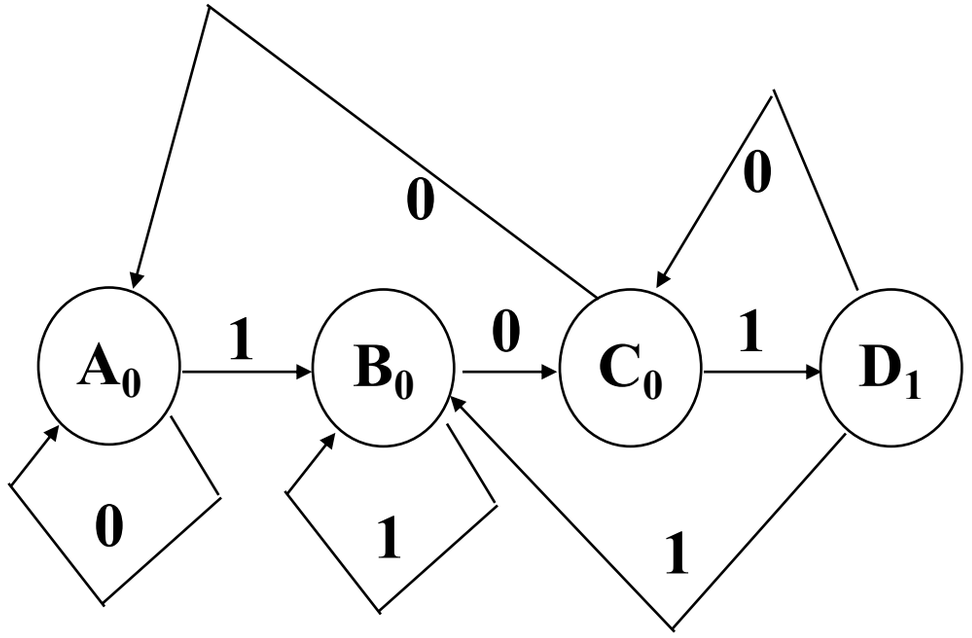
La tabella di transizione degli stati e delle uscite:

Svolgimento

La realizzazione del circuito:

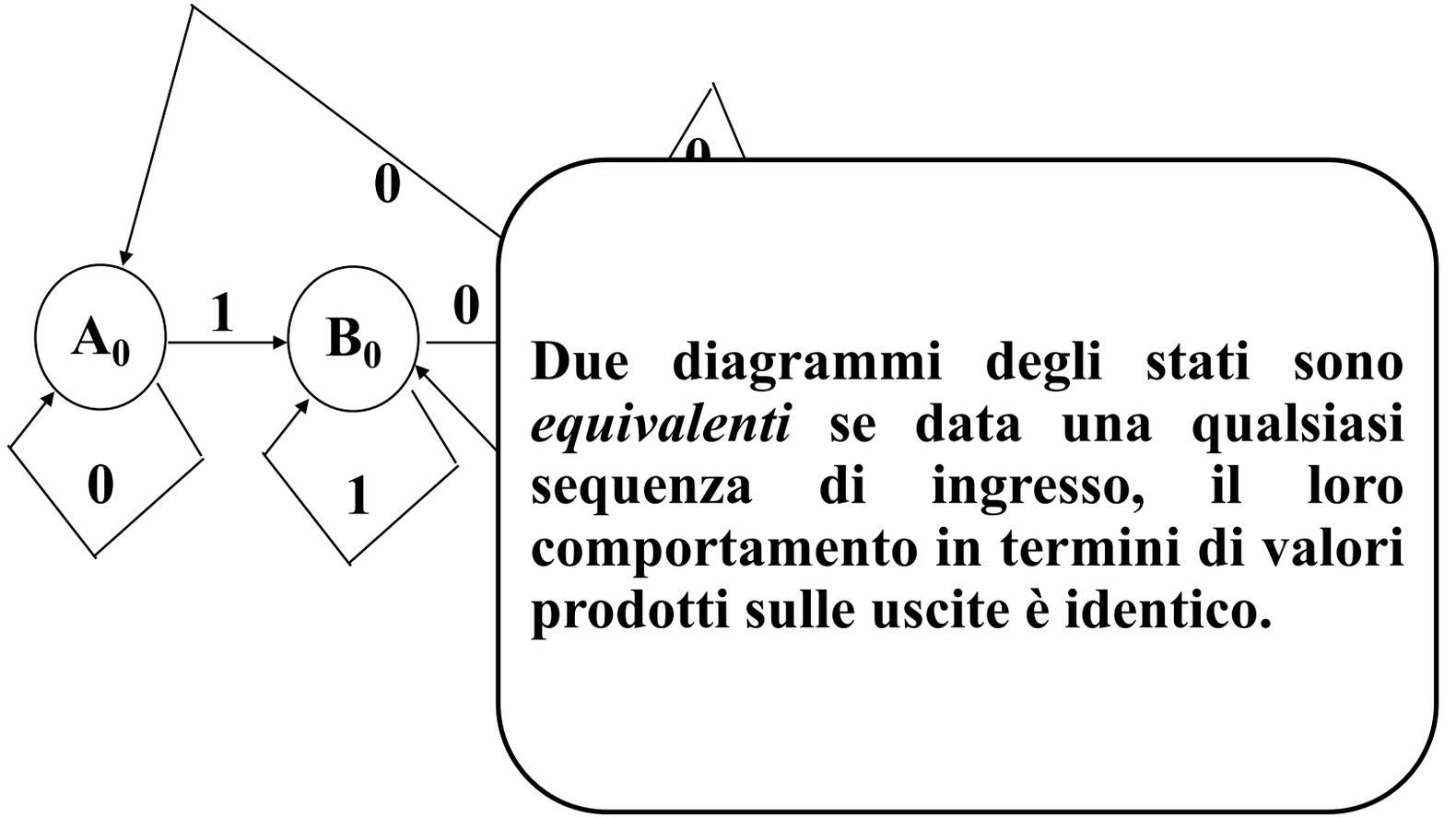
Esempio (II)

Lo stato E è equivalente allo stato C; il diagramma può quindi essere così semplificato:



Esempio (II)

Lo stato E è equivalente allo stato C; il diagramma può quindi essere così semplificato:



Due diagrammi degli stati sono *equivalenti* se data una qualsiasi sequenza di ingresso, il loro comportamento in termini di valori prodotti sulle uscite è identico.

Esempio (III)

Una possibile codifica degli stati è:

	Y_1	Y_2
A	**0**	**0**
B	**0**	**1**
C	**1**	**0**
D	**1**	**1**

Esempio (IV)

La tavola di verità della funzione di transizione degli stati è:

$i\ y_1\ y_2 \rightarrow Y_1\ \ Y_2$

0 00 → 00
1 00 → 01
0 01 → 10
1 01 → 01
0 10 → 00
1 10 → 11
0 11 → 10
111 → 01

Il valore dell'uscita z può essere facilmente generato tenendo conto che esso è 1 solo quando il circuito è nello stato 11:

$i\ y1\ y2 \rightarrow z$

- 00 → 0
- 01 → 0
- 10 → 0
- 11 → 1

Esempio (V)

Si possono ora sintetizzare le 3 funzioni Y_1, Y_2 e z, ottenendo:

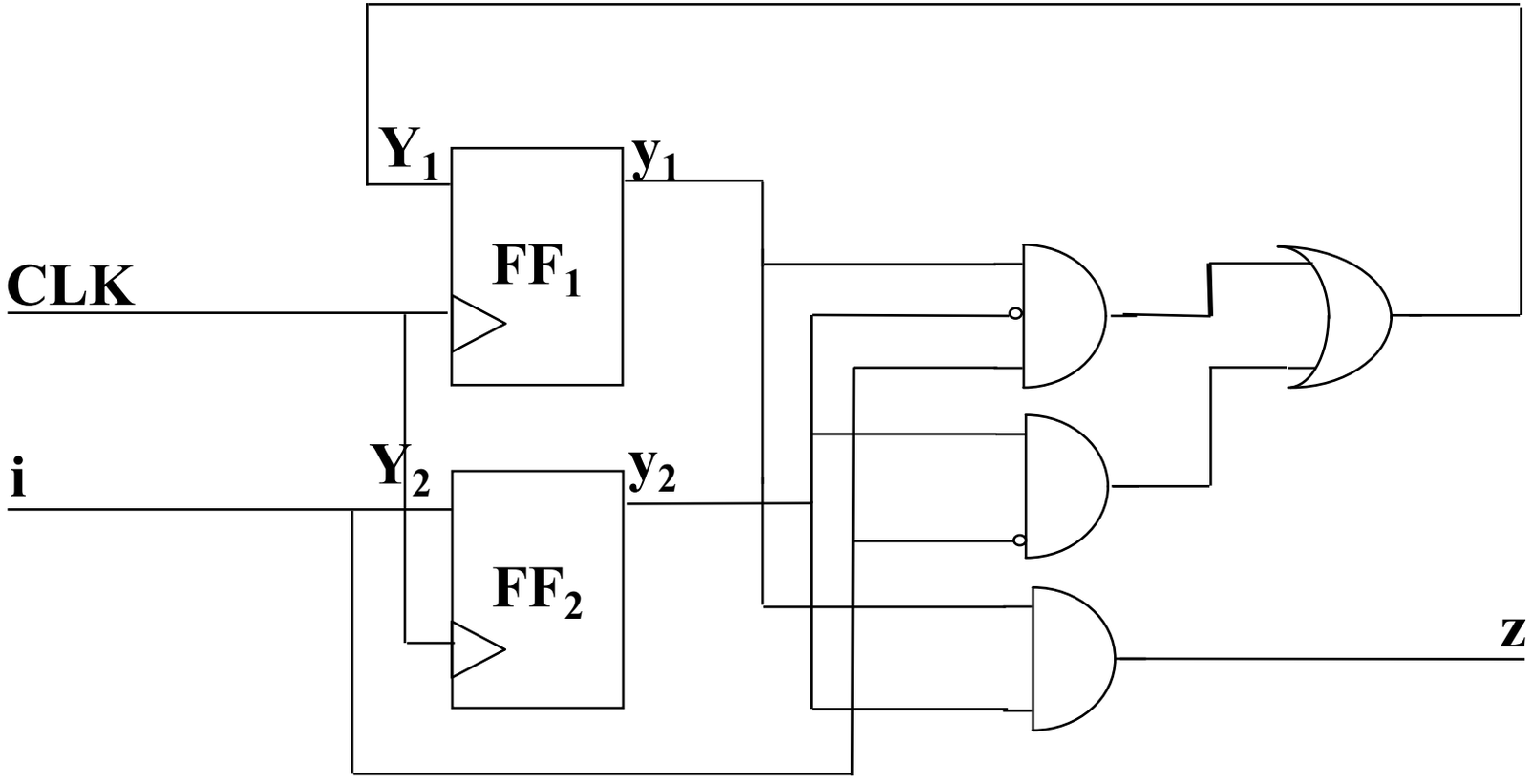
$$Y_1 = y_1\,\overline{y_2}\,i + y_2\,\bar{\iota}$$

$$Y_2 = i$$

$$z = y_1 y_2$$

Esempio (VI)

Il circuito che implementa il riconoscitore di sequenza è:



Frequenza di lavoro

- **La frequenza del clock *f* deve essere determinata in modo che il tempo**

***T = 1 / f* tra due fronti di clock successivi rispetti la seguente disuguaglianza**

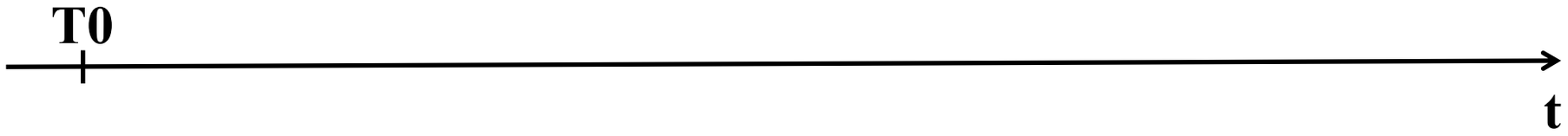
$$T > \Delta + \delta$$

dove

- **Δ è il massimo ritardo della logica combinatoria**
- **δ è il ritardo dei flip flop.**

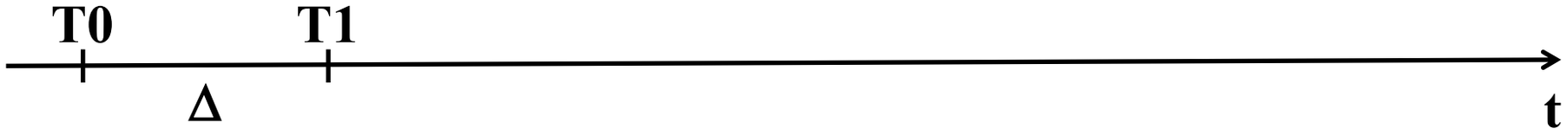
Frequenza di lavoro

- **T0: un nuovo insieme di valori è applicato agli ingressi della parte combinatoria**



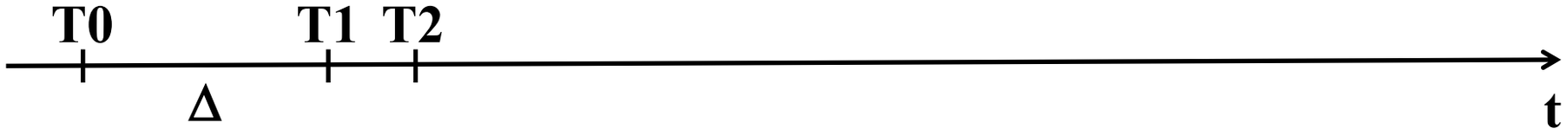
Frequenza di lavoro

- **T0: un nuovo insieme di valori è applicato agli ingressi della parte combinatoria**
- **T1 = T0 + Δ: le uscite della parte combinatoria assumono un valore stabile**



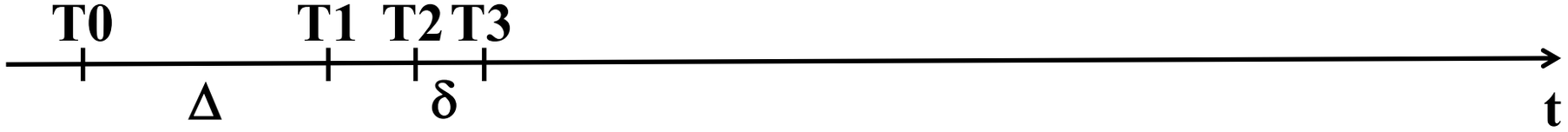
Frequenza di lavoro

- **T0: un nuovo insieme di valori è applicato agli ingressi della parte combinatoria**
- **T1 = T0 + Δ: le uscite della parte combinatoria assumono un valore stabile**
- **T2: si applica un nuovo fronte al segnale di clock; il valore sulle uscite della parte combinatoria è memorizzato nei flip flop**



Frequenza di lavoro

- **T0: un nuovo insieme di valori è applicato agli ingressi della parte combinatoria**
- **T1 = T0 + Δ: le uscite della parte combinatoria assumono un valore stabile**
- **T2: si applica un nuovo fronte al segnale di clock; il valore sulle uscite della parte combinatoria è memorizzato nei flip flop**
- **T3 = T2 + δ: il nuovo valore è disponibile sulle uscite dei flip flop**



Frequenza di lavoro

- **T0: un nuovo insieme di valori è applicato agli ingressi della parte combinatoria**
- **T**
- **T**
- **T3**

I tempi tra T1 e T2 e tra T3 e T0 possono essere minimizzati a piacere dall'utente.

In ogni caso deve essere

$$T > \Delta + \delta$$