Calcolatori Elettronici B
a.a. 2006/2007

RETI LOGICHE: RICHIAMI

Massimiliano Giacomin

Due tipi di unità funzionali

•Elementi di tipo *combinatorio*:
- valori di uscita dipendono solo da valori in ingresso
- Es. Porte logiche, PLA

• Elementi *di memoria:*
- capaci di memorizzare un valore
- Es. flip-flop, registri, memoria RAM

Due tipi di reti

• **RETI COMBINATORIE**

- Contengono solamente elementi di tipo combinatorio

⇒ Valori di uscita dipendono solo da valori di ingresso

• **RETI SEQUENZIALI**

- Contengono elementi di memoria

⇒ Valori di uscita dipendono dalla storia del sistema (sequenza di tutti gli ingressi) – sintetizzata nel valore di stato contenuto negli elementi di memoria.

RETI COMBINATORIE

Assumiamo **n** ingressi **m** uscite

Specifica

- Si possono specificare mediante due approcci alternativi:
  - **tabelle di verità**

    (n ingressi $\Rightarrow 2^n$ righe, per ciascuna si specificano tutte le m uscite)

Es.

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

(OR esclusivo)

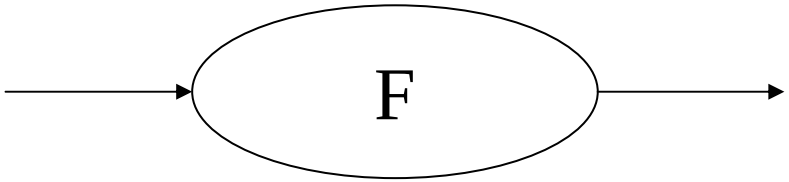
  - **equazioni logiche** (algebra booleana)

    Es. $A\overline{B}+\overline{A}B$

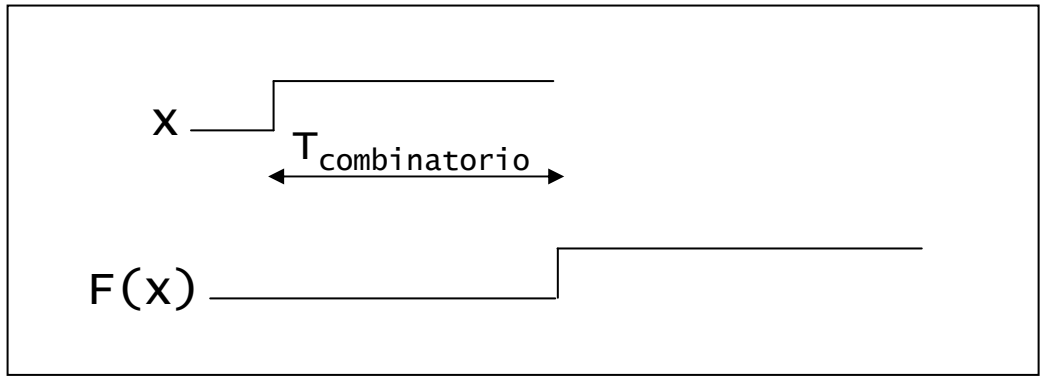
Realizzazione: Porte logiche, PLA, ROM, PLD (Programmable logic device)

- Elettronica digitale: realizzazione degli elementi sopraindicati (es. Famiglie logiche TTL, CMOS) – noi non ce ne occupiamo direttamente
- Determina parametri tecnologici che influenzano le prestazioni

Tempo di propagazione



Dal momento in cui l'ingresso è valido al momento in cui l'uscita è valida trascorre un certo intervallo temporale:



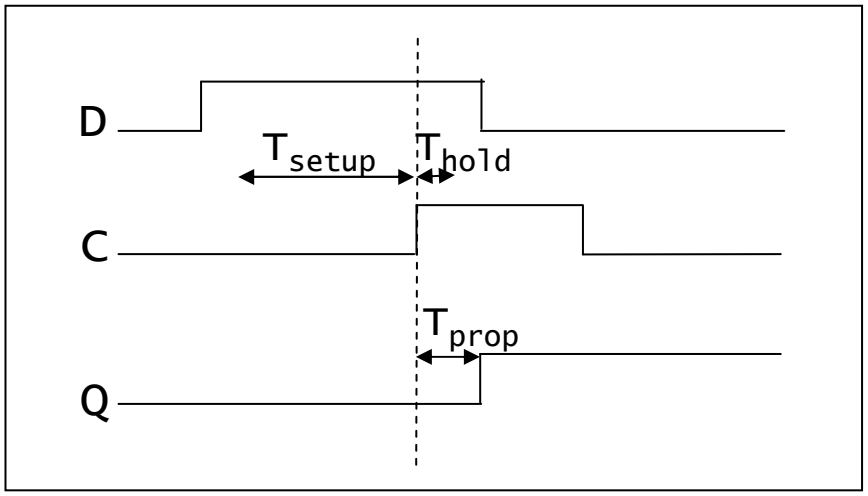
RETI SEQUENZIALI

Elementi di memoria

- **Flip-flop di tipo D**

Sensibile ai fronti: l'ingresso è memorizzato sul fronte (di salita) del clock

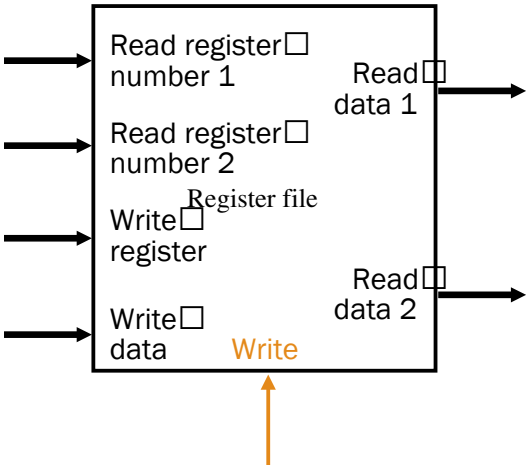
- Vincoli sull'ingresso: tempo di setup e tempo di hold
  Ritardo sull'uscita: tempo di propagazione



- Tutti i tempi riferiti al fronte del clock
- In generale $T_{hold} < T_{prop}$

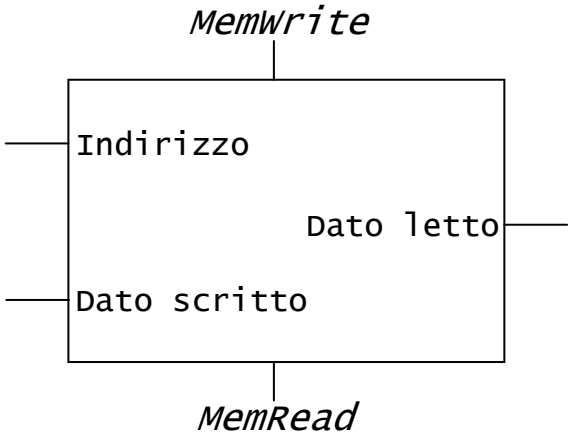
• **Registri**: capaci di memorizzare un insieme di bit
(si possono ottenere mediante *array* di Flip-flop di tipo D)

• **Register File:**



- Lettura: asincrona rispetto al clock, senza segnale di controllo *read*
- Scrittura: attiva sul fronte del clock e solo quando *write* è affermato
- Implementato con registri, multiplexer (per read port) e decodificatore (per write port)

• **Memorie** (per memorizzare quantità maggiori di dati)

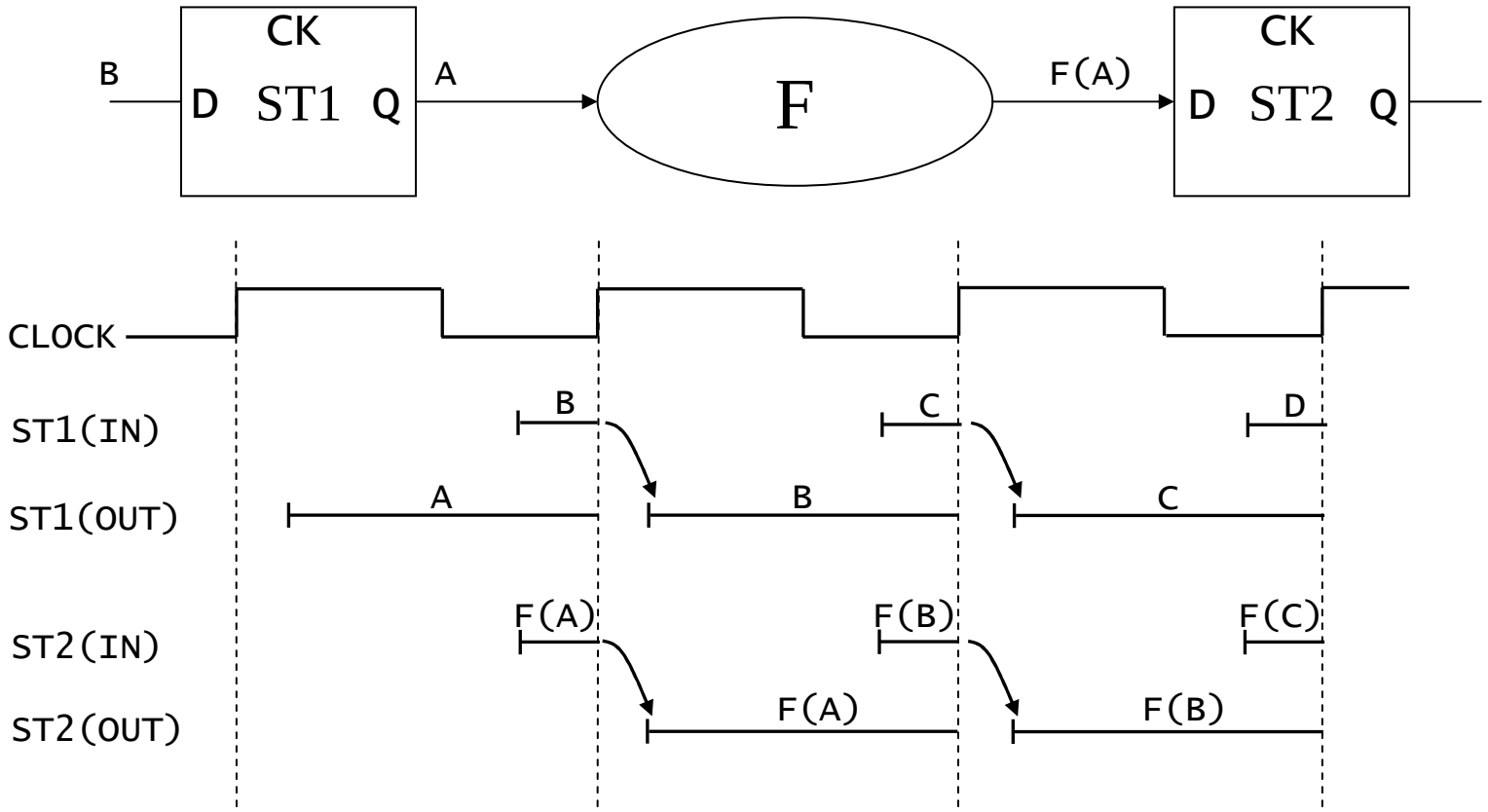


- Lettura asincrona risp. clock, scrittura attiva sul fronte del clock
- NB: forma semplificata (cfr. SRAM, DRAM, ecc.)

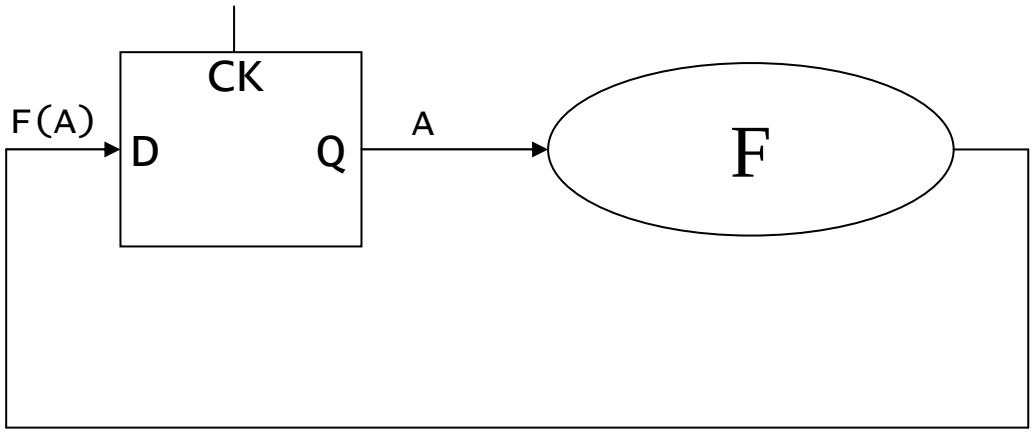
NB: il clock è presente ma non viene indicato per rendere le figure più chiare.

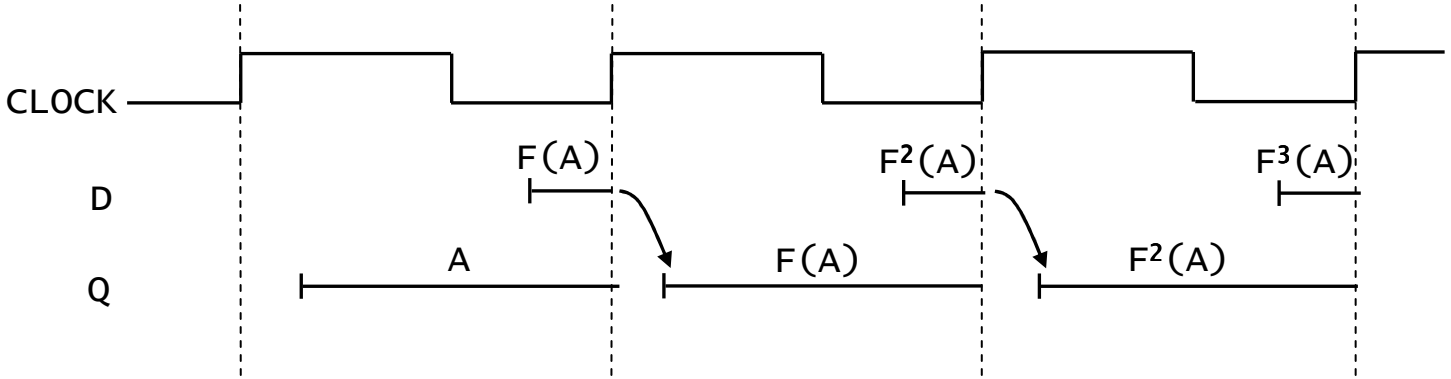
Temporizzazione

- Sistemi sincroni: segnale di clock comune determina aggiornamento elementi di stato

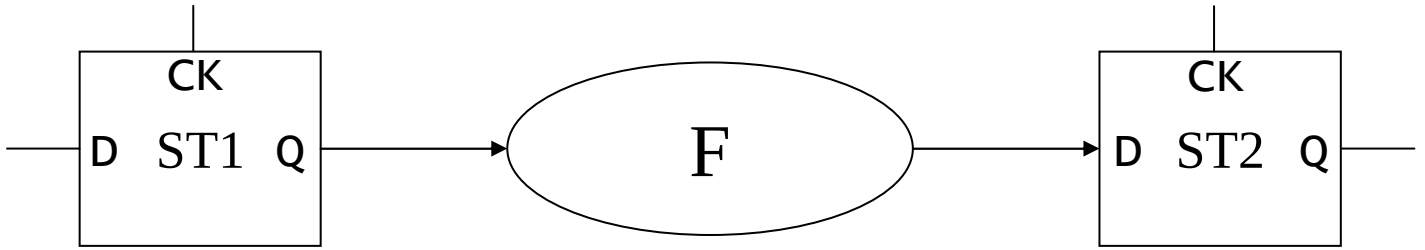


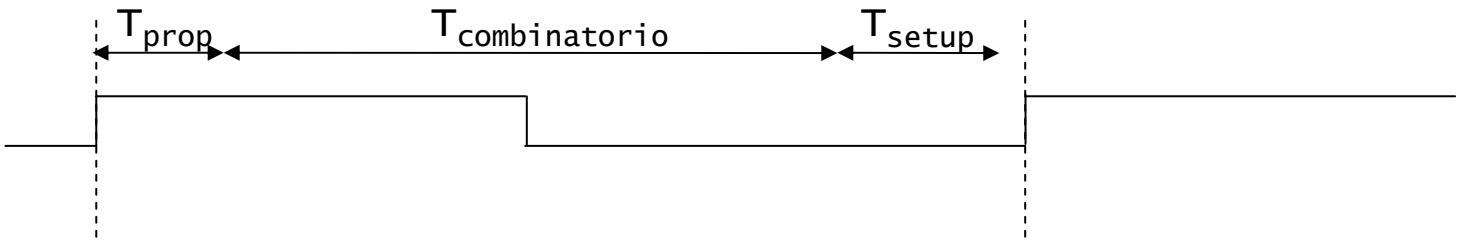
- Al fronte di clock, un elemento di stato memorizza il valore di ingresso
- Nel periodo di clock, un nuovo valore di ingresso viene propagato dalla parte combinatoria e sarà disponibile al successivo fronte


CK
F(A)
D
Q
A
F


CLOCK
D
Q
F(A)
F²(A)
F³(A)
A
F(A)
F²(A)

Temporizzazione e vincoli temporali





- Dopo $T_{prop} + T_{combinatorio}$, ingresso a ST2 è stabile: anticipo di almeno T_{clock}

  ⇒ $T_{clock} \geq T_{prop} + T_{combinatorio} + T_{setup}$

- Vincolo per rispetto di T_{hold} : ingresso ST2 permane per almeno T_{hold} dopo il fronte

  ⇒ $T_{prop} + T_{combinatorio} \geq T_{hold}$

  [verificato automaticamente perché $T_{hold} < T_{prop}$]

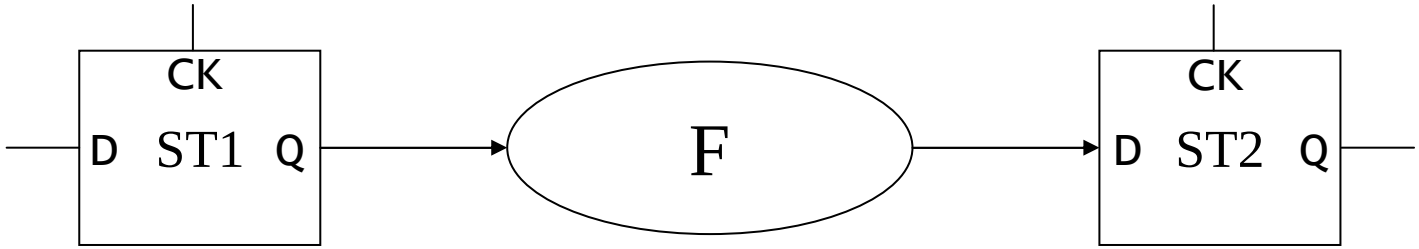
Un problema: lo sfasamento del clock

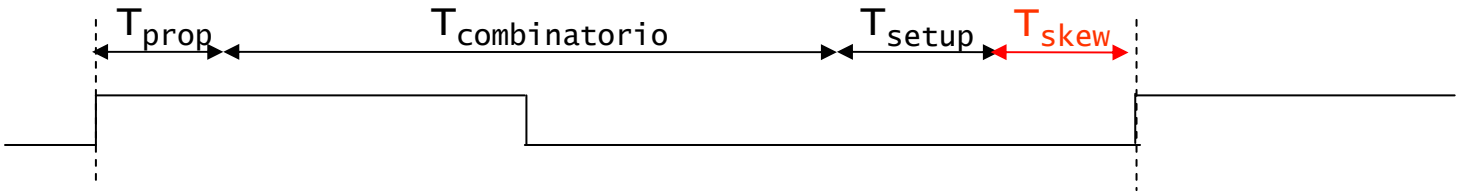
- Segnale di clock può percorrere cammini diversi per arrivare a elementi di stato

  ➡ T_{skew} = differenza tra istanti temporali in cui due elementi di stato ricevono il fronte di clock

  ➡ Vincoli su tempi di propagazione / tempo di hold

Caso 1: il clock è in anticipo di T_{skew} su ST2, che si aggiorna in anticipo:

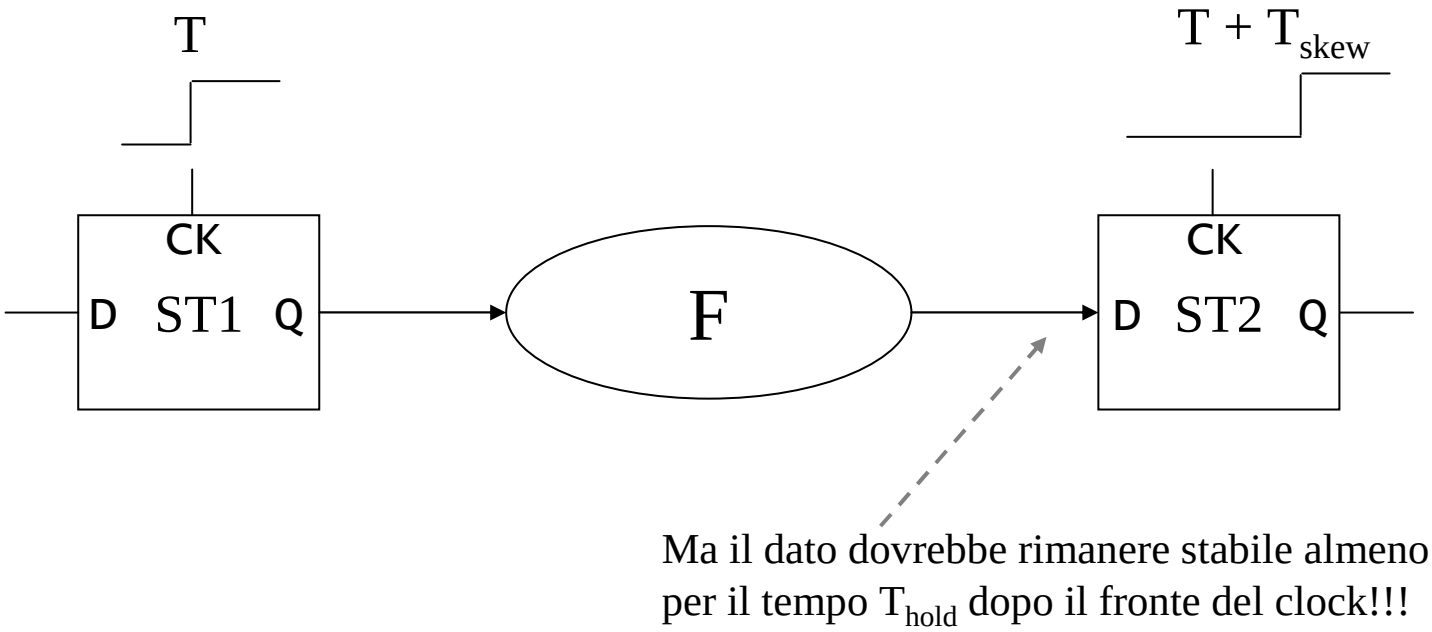




➡ $T_{clock} \geq T_{prop} + T_{combinatorio} + T_{setup} + T_{skew}$

Caso 2: il clock è in anticipo di T_{skew} su ST1

Può accadere che il nuovo dato si propaghi fino all'ingresso di ST2 prima che ST2 si sia aggiornato: ST2 perde un dato!

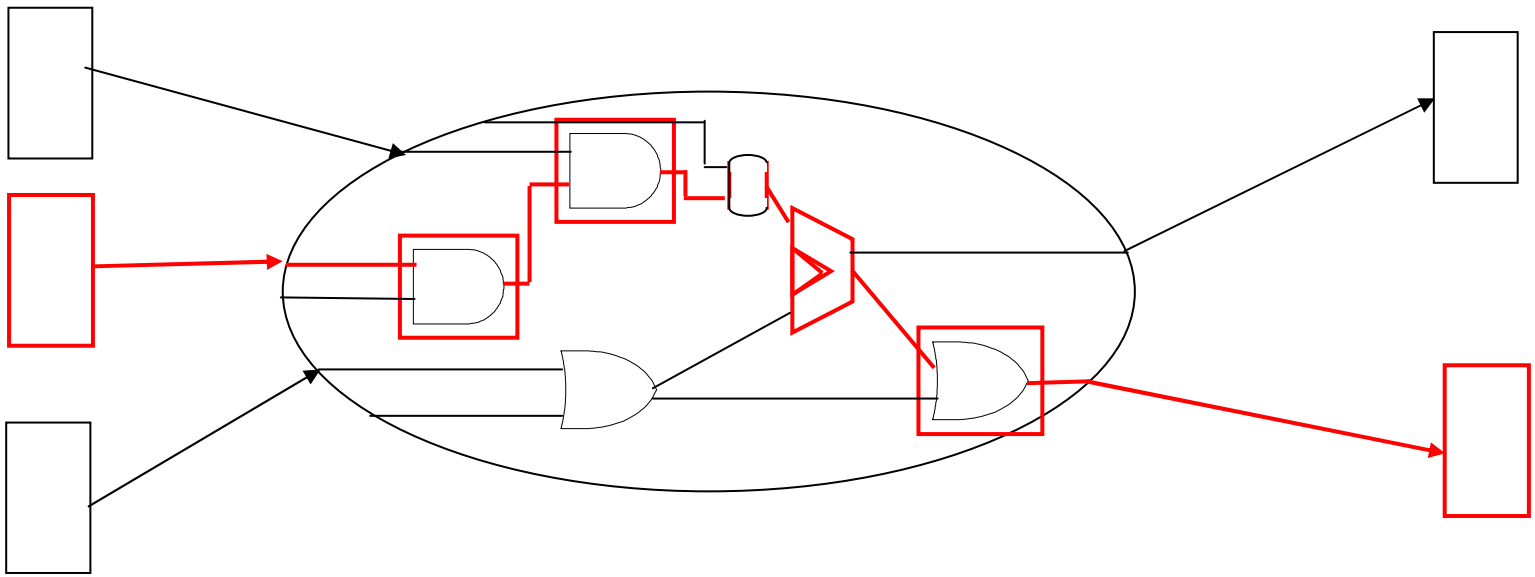


$$T_{prop} + T_{combinatorio} - T_{skew} \geq T_{hold}$$

NB: si vede che l'effetto in questo caso è sul tempo di hold (non sul periodo di clock!) che può diventare determinante; se $T_{skew} > T_{prop}+T_{combinatorio}$, non c'è niente da fare!

Attenzione dei progettisti mediante attento instradamento segnale di clock

ESTENDENDO QUESTE CONSIDERAZIONI AD UNA RETE COMPLESSA…



Occorre considerare il caso peggiore; in particolare il **“cammino critico”** vincola la lunghezza del periodo di clock e quindi limita la frequenza ottenibile!

$$T_{clock} \geq T_{prop} + T_{\text{cammino critico}} + T_{setup} + T_{skew}$$

Nel caso peggiore!

Specifica di una rete sequenziale: macchine a stati finiti (FSM)

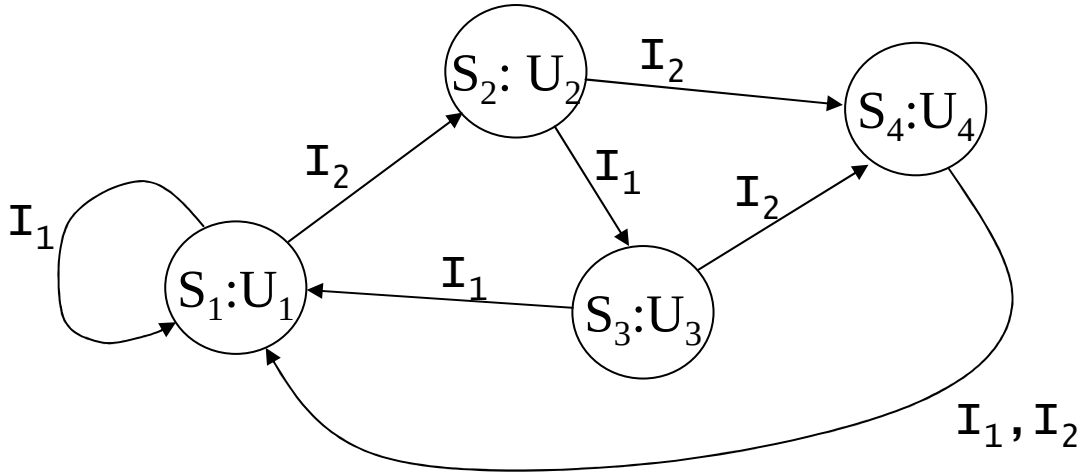
Occorre definire:

- L'insieme degli ingressi (dominio **I**) e delle uscite (dominio **U**)
- L'insieme degli stati **S**
- Dinamica (come si passa da uno stato all'altro):

  funzione **f**: **S*I $\rightarrow$ S**
- Come si generano le uscite

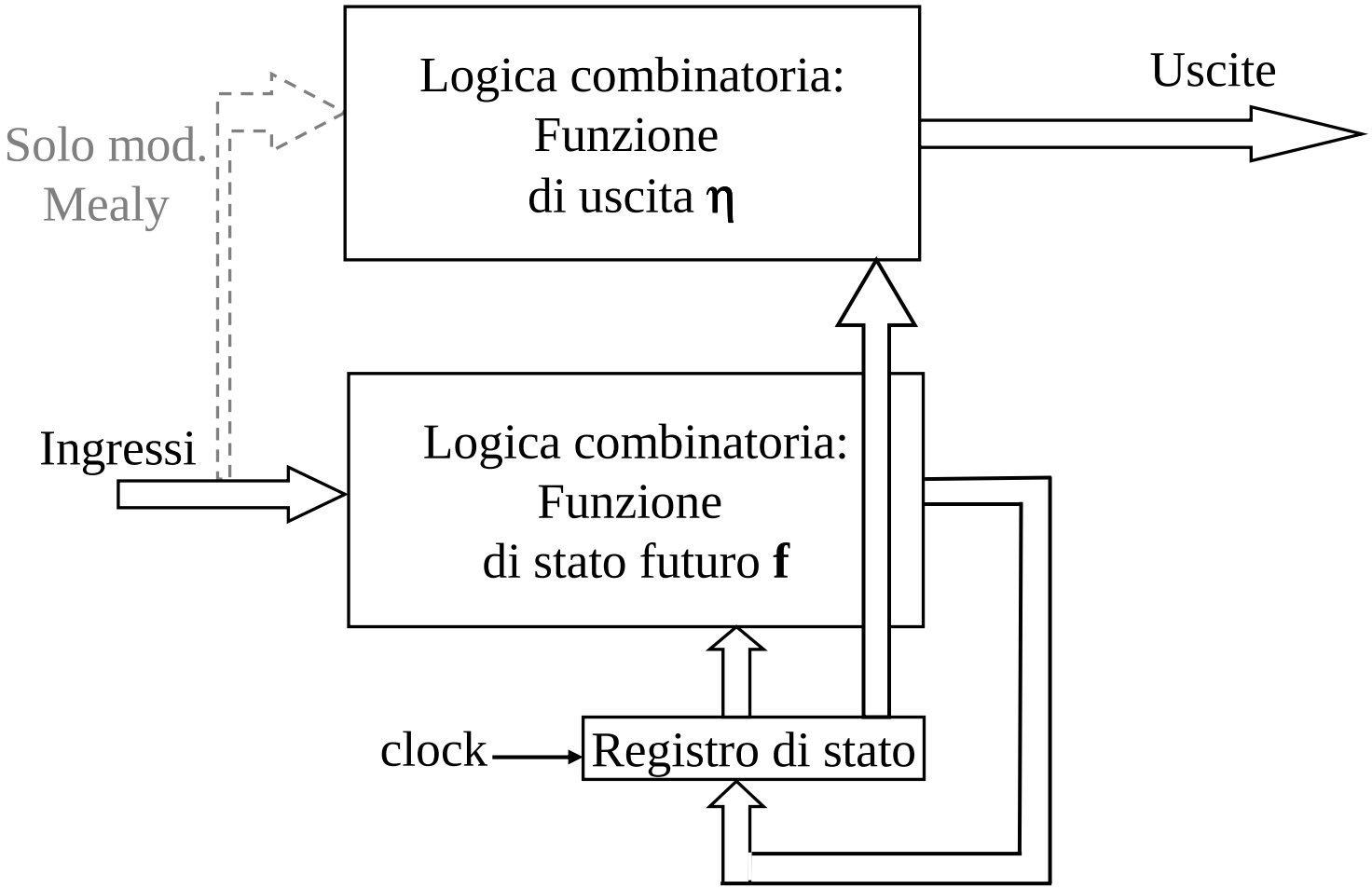
  funzione η **η: S $\rightarrow$ U** **Modello di Moore**

  η: S*I $\rightarrow$ U **Modello di Mealy**



[Modello di Moore]

Implementazione di una rete sequenziale



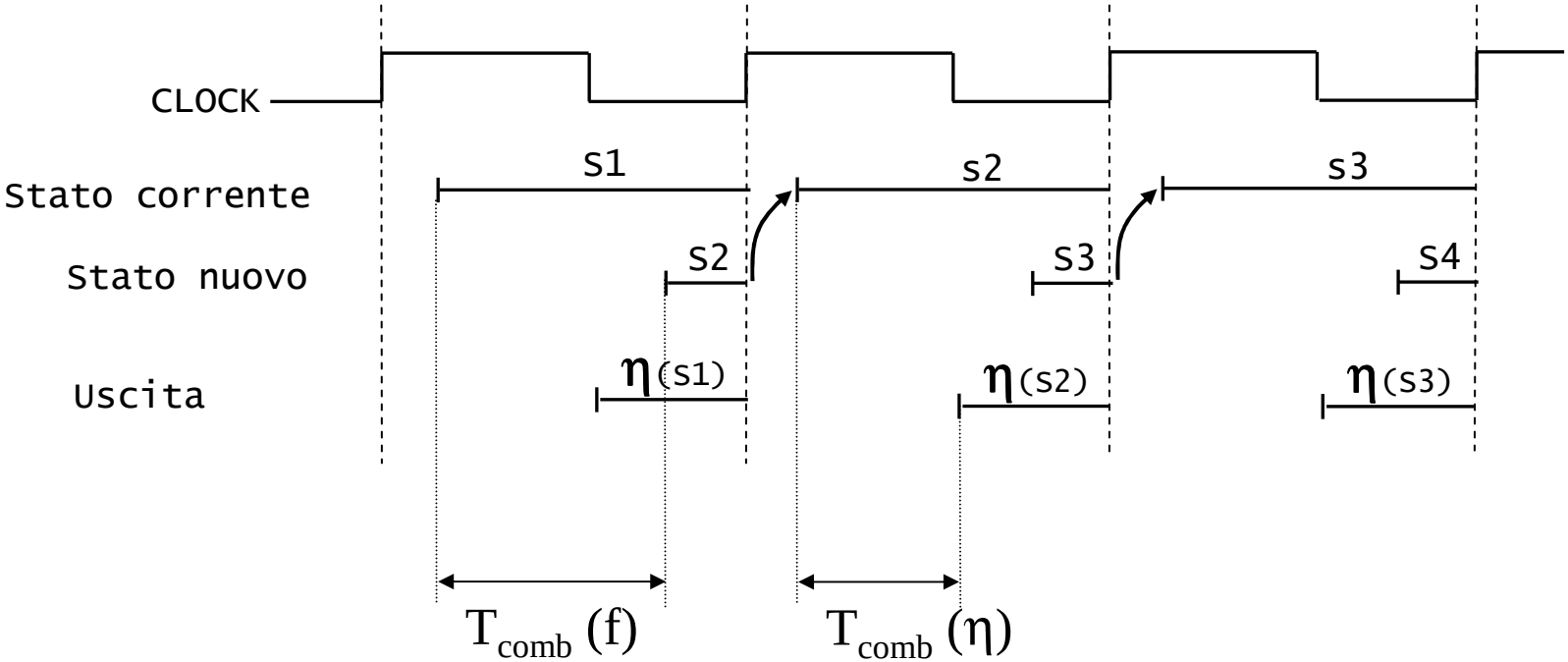
NB: logica combinatoria realizzata con PLA o ROM

Mealy vs. Moore

- Modelli logicamente equivalenti, però:
  - FSM di Mealy richiede in genere meno stati
    (cfr. stesso stato con uscite diverse)
  - FSM di Moore è in genere più veloce
    (funzione di uscita più semplice)

→ Adottato: **MOD. MOORE**

Ancora sulla temporizzazione (Mod. Moore)

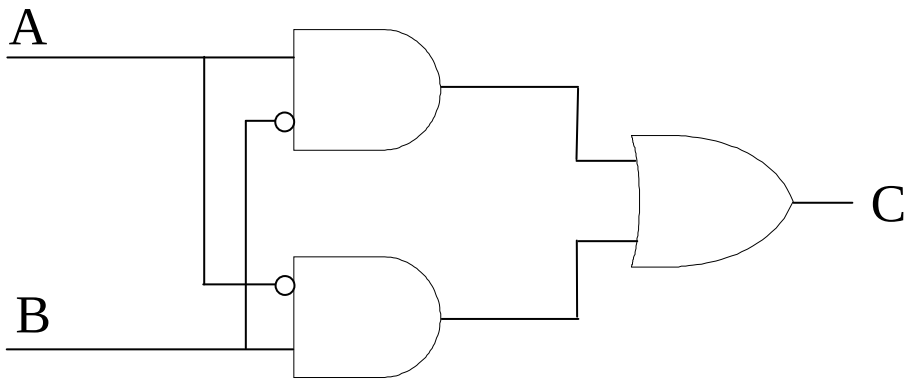


Appendice

Realizzazione di reti combinatorie mediante PLA e ROM

Realizzazione di reti combinatorie

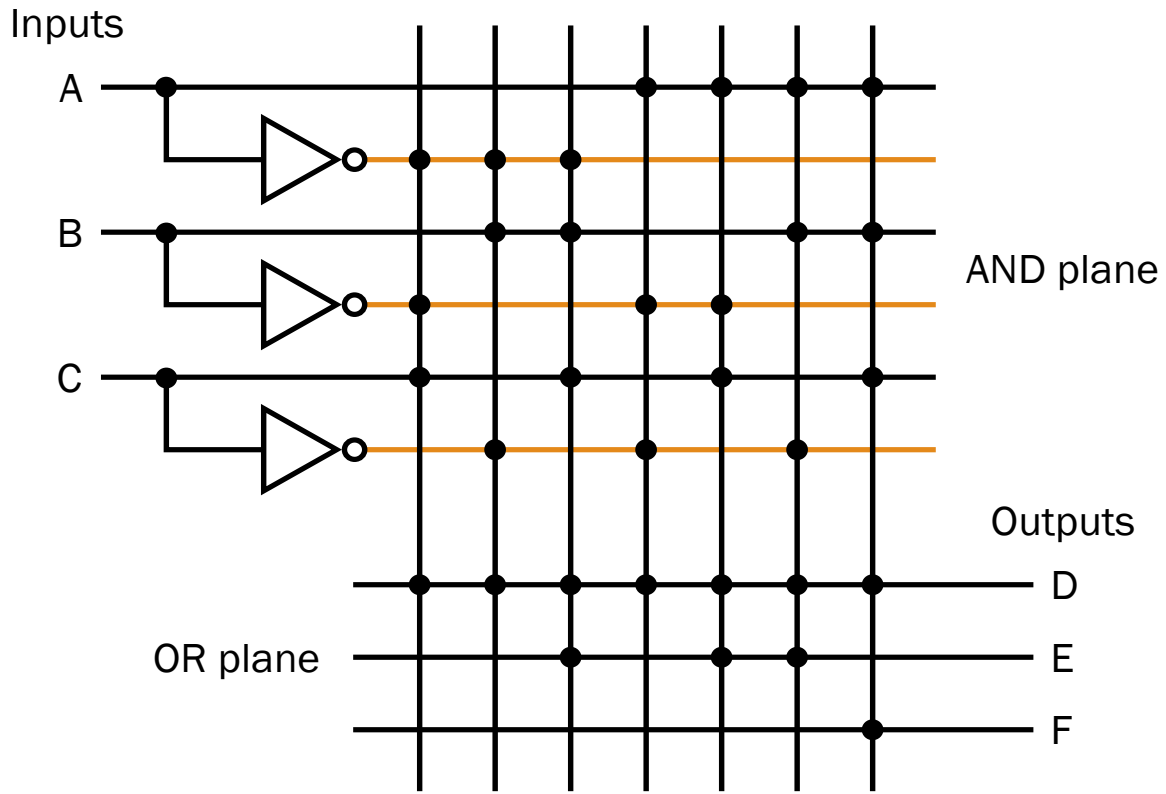
- Mediante combinazione di porte logiche



- Mediante **PLA** (Programmable Logic Array)
  - Specifica come somma di prodotti
  - Dimensione: **n*P + P*m** [P=numero dei termini prodotto]

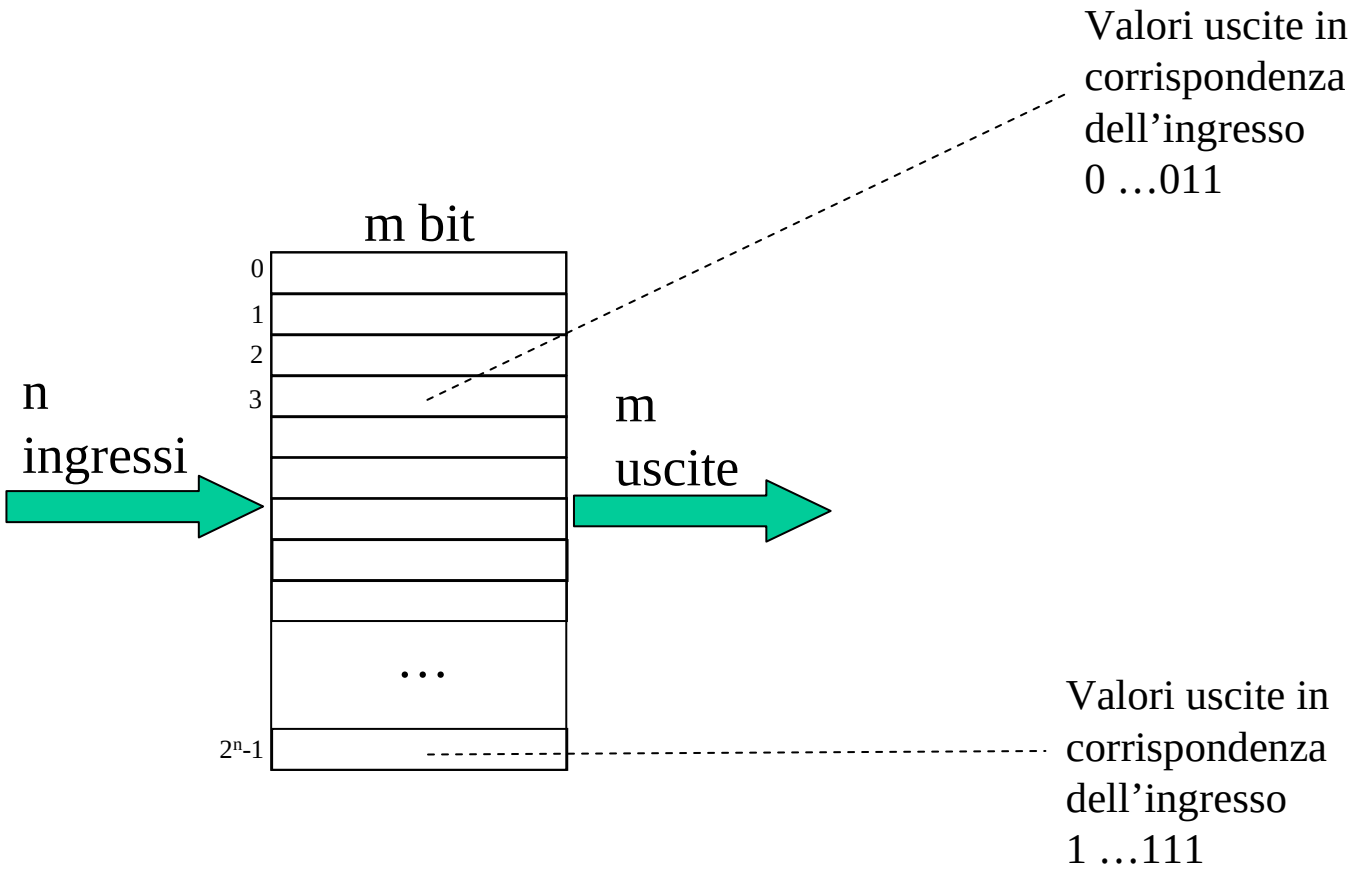
- Mediante **ROM** (Read Only Memory)
  - Dispositivi “completamente decodificati”
  - Forma: Altezza = 2^n e ampiezza = m
  - Numero di bit = **2^n * m**

PLA



Dimensione = dimensione della matrice AND + matrice OR =
= n*P + P*m

ROM



Dimensione = numero delle celle * ampiezza (in bit) della cella =

= $\mathbf{2^n * m}$

PLA vs. ROM

Vantaggio PLA: dimensioni contenute

- è sufficiente tenere conto dei soli elementi della tabella di verità che producono un valore vero per almeno un'uscita;
- i termini prodotto della PLA possono essere utilizzati in più uscite
- invece, la ROM è completamente decodificata (m bit per ognuna delle 2^n righe)

⇒ Numero di elementi ROM esponenziale rispetto a numero n di ingressi
Numero di elementi PLA in pratica cresce molto più lentamente

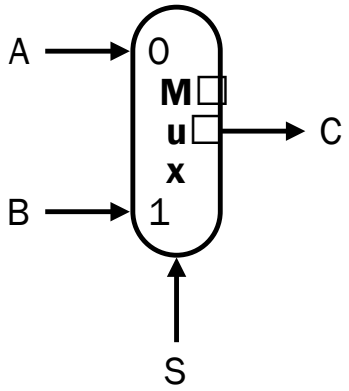
Vantaggio ROM: maggior facilità di cambiamento

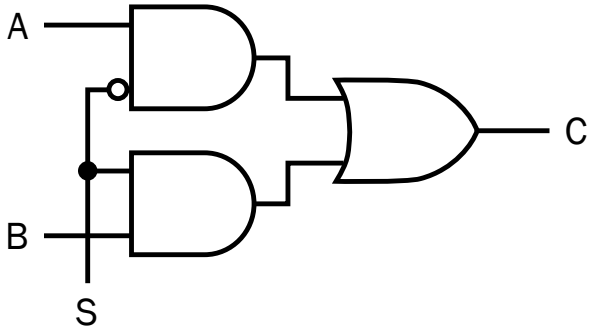
- dati n e m, le dimensioni della ROM non cambiano al variare della funzione logica: se funzione logica cambia, basta modificare il contenuto della ROM

In generale, diamo per scontato di saper realizzare qualsiasi funzione logica!

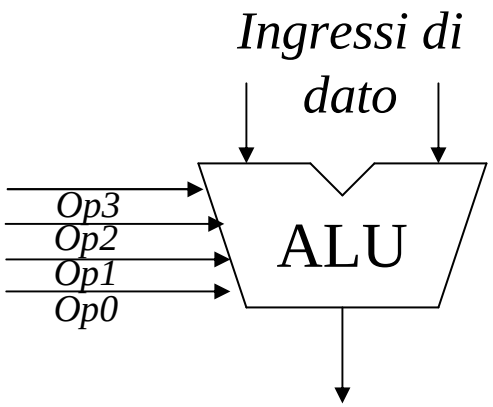
Esempi di reti combinatorie

Multiplexer





ALU



	Op3	*Op2*	*Op1*	*Op0*
AND	0	0	0	0
OR	0	0	0	1
Somma	0	0	1	0
Sottrazione	0	1	1	0
Set on less than	0	1	1	1
NOR	1	1	0	0