Calcolatori Elettronici B
a.a. 2005/2006

RETI LOGICHE: RICHIAMI

Massimiliano Giacomin

Unità funzionali

- Unità funzionali:
  - •Elementi di tipo *combinatorio*:
    - valori di uscita dipendono <u>solo</u> da valori in ingresso
    - Es. Porte logiche, PLA
  - • Elementi *di memoria:*
    - capaci di memorizzare un valore
    - Es. flip-flop, registri, memoria RAM

<u>Elettronica digitale</u>

- Realizzazione circuitale degli elementi sopraindicati
  (es. Famiglie Logiche TTL, CMOS)
- Non ce ne occupiamo direttamente: assumiamo gli elementi come primari
- Determina parametri tecnologici quali tempi di propagazione, ecc.
  che influenzano le prestazioni

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

Specifica ed implementazione

Nella pratica progettazione e realizzazione sono attività separate
ma non indipendenti

Progettazione : risulta in un documento di specifica: il documento descrive il sistema da realizzare in una opportuna notazione.

Implementazione (realizzazione): traduce il documento di specifica in circuiti realizzati secondo una tecnologia *opportuna.*

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

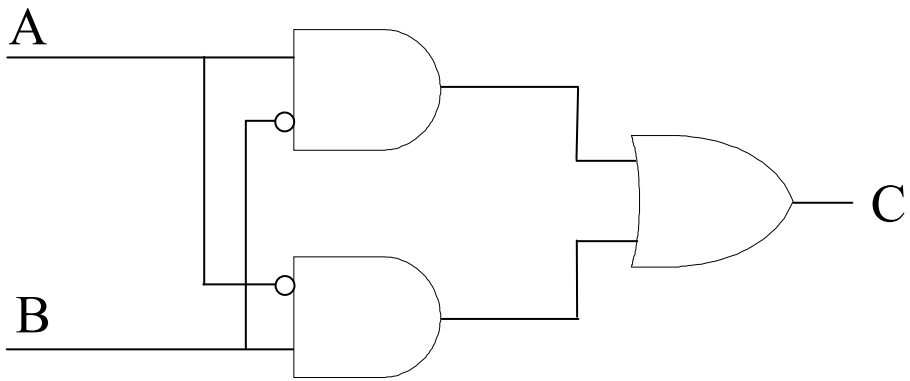
(OR esclusivo)

  - **equazioni logiche** (algebra booleana)

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

Realizzazione

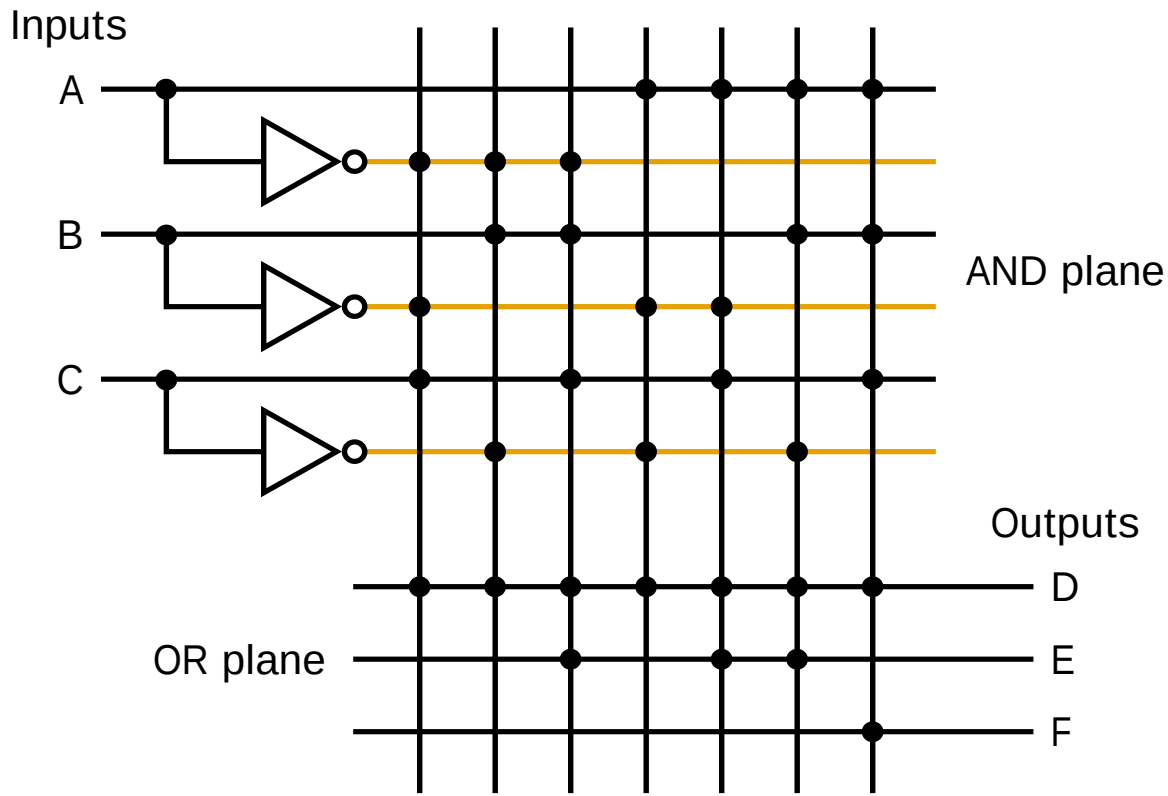
- Mediante combinazione di porte logiche



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

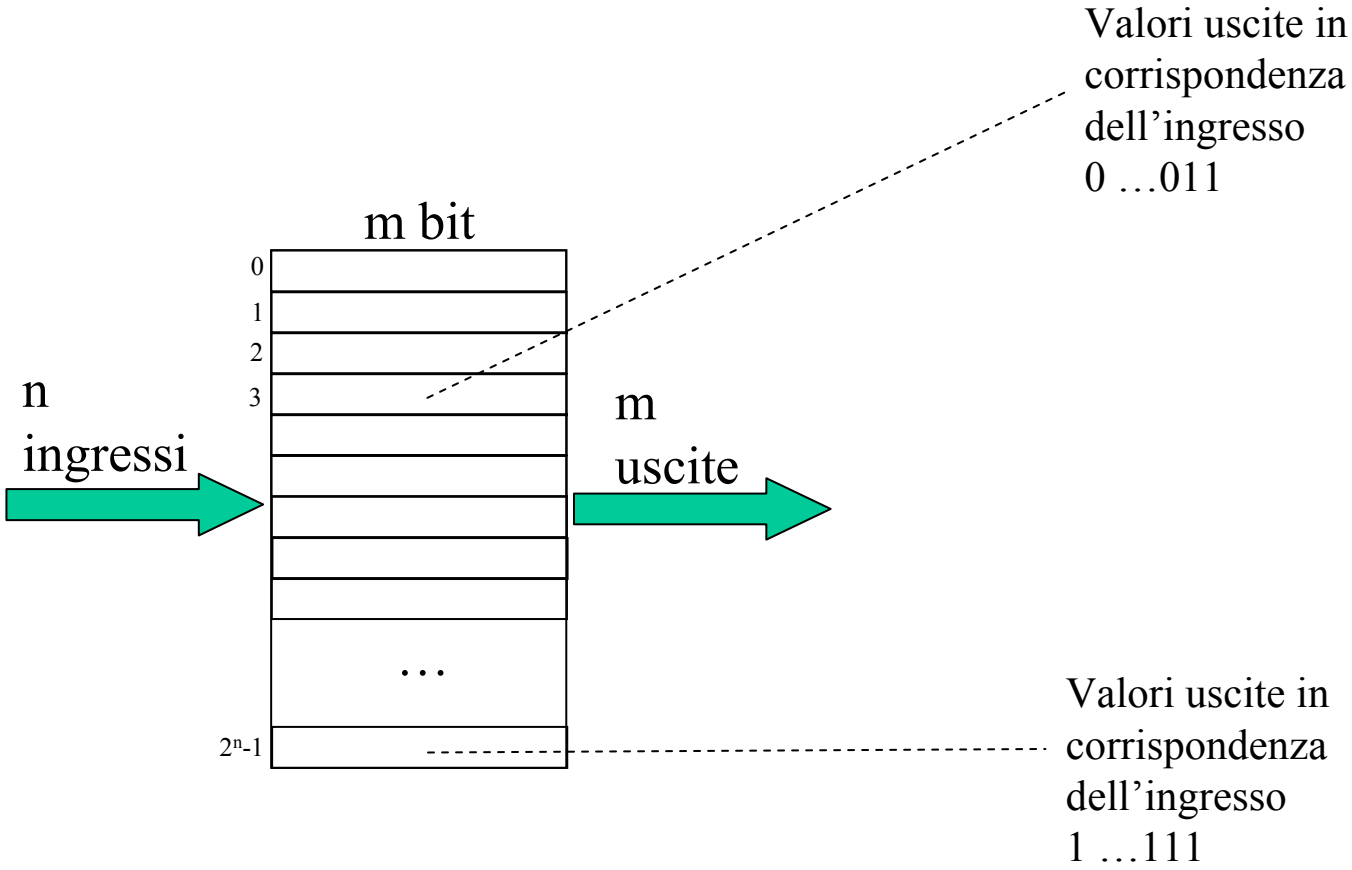
- Mediante **ROM** (Read Only Memory)
  - Dispositivi “completamente decodificati”
  - Forma: Altezza = 2^n e ampiezza = m
  - Numero di bit = **2^n * m**

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 codificata (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

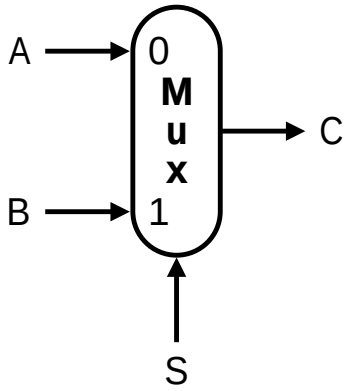
Vantaggio ROM: maggior facilità di cambiamento

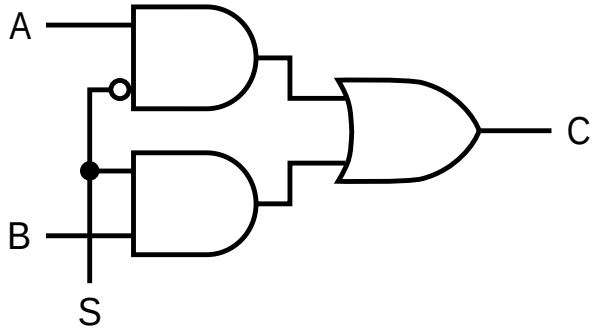
- dati n e m, le dimensioni della ROM non cambiano al variare della funzione logica: se funzione logica cambia, basta modificare il contenuto della ROM

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

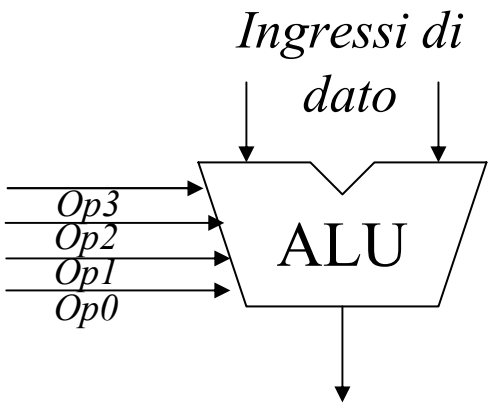
Esempi di reti combinatorie

Multiplexer



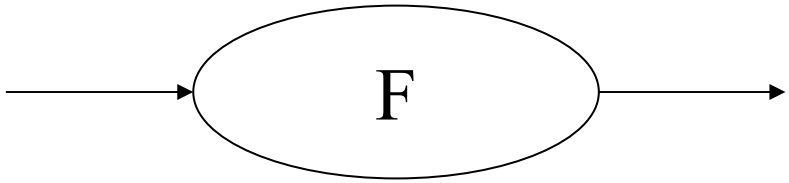


ALU

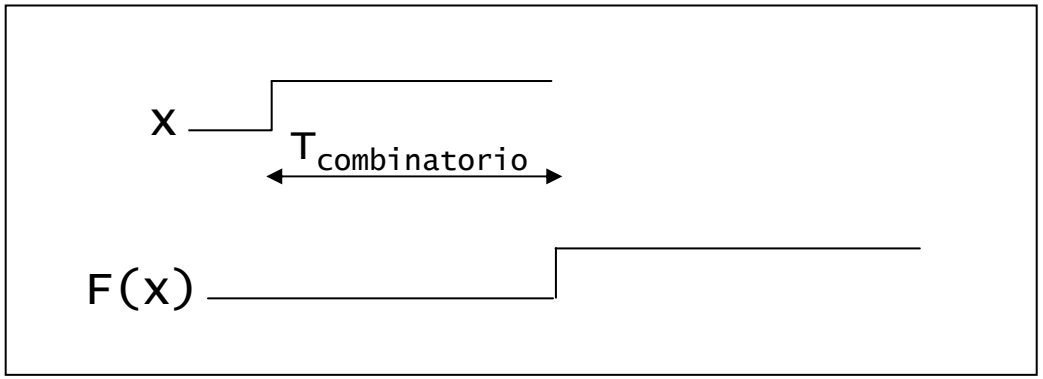


	Op3	*Op2*	*Op1*	*Op0*
AND	0	0	0	0
OR	0	0	0	1
Somma	0	0	1	0
Sottrazione	0	1	1	0
Set on less than	0	1	1	1
NOR	1	1	0	0

Tempo di propagazione



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



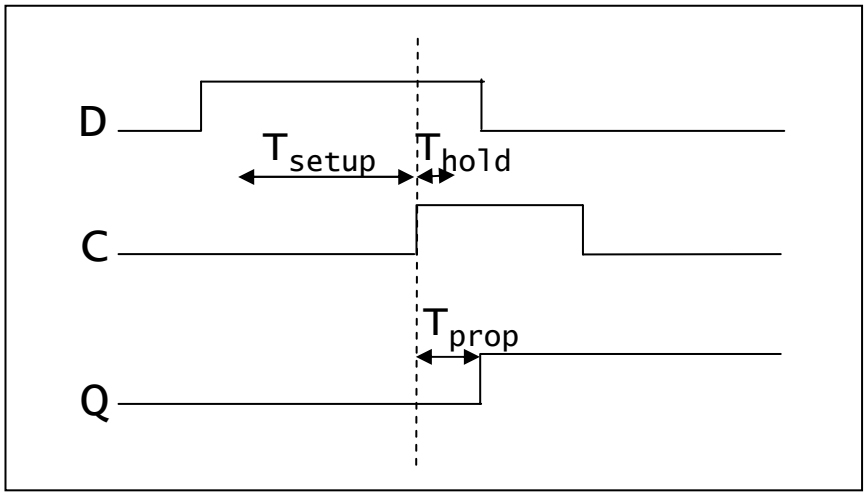
RETI SEQUENZIALI

Elementi di memoria

- **Flip-flop di tipo D**

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

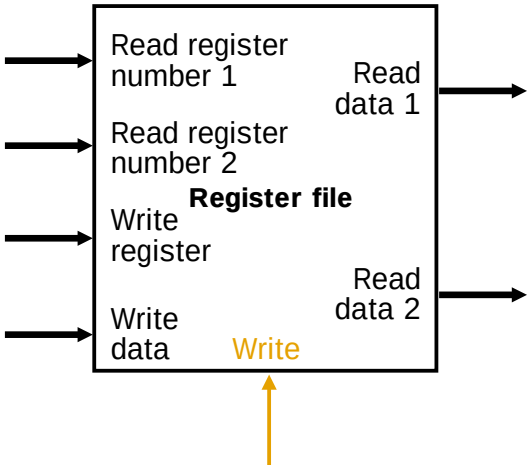
- Vincoli sull'ingresso: tempo di setup e tempo di hold
  Ritardo sull'uscita: tempo di propagazione



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

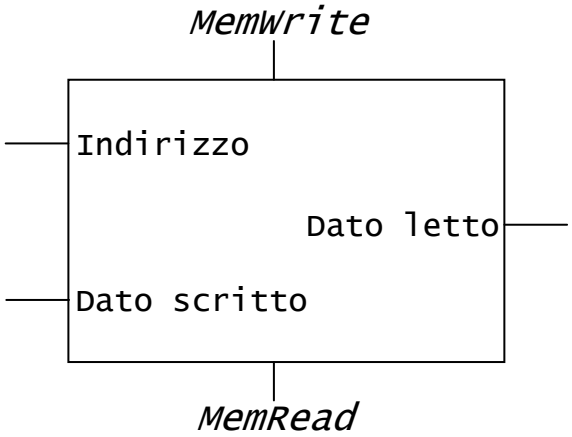
• **Registri**: capaci di memorizzare un insieme di bit
(si possono ottenere mediante *array* di Flip-flop di tipo D)

• **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 e decodificatore

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

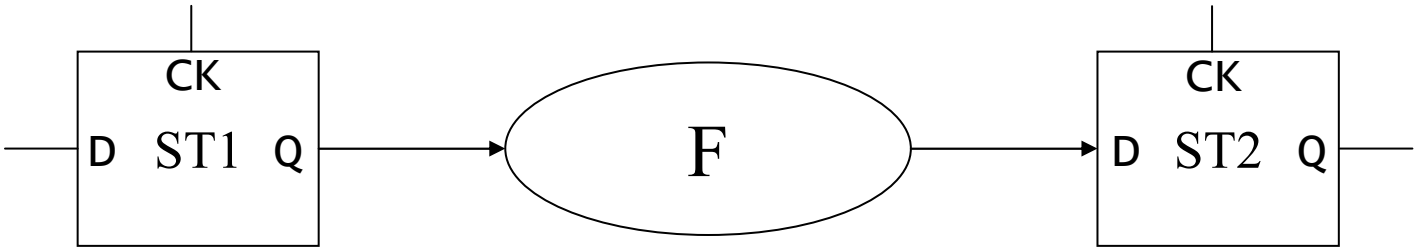


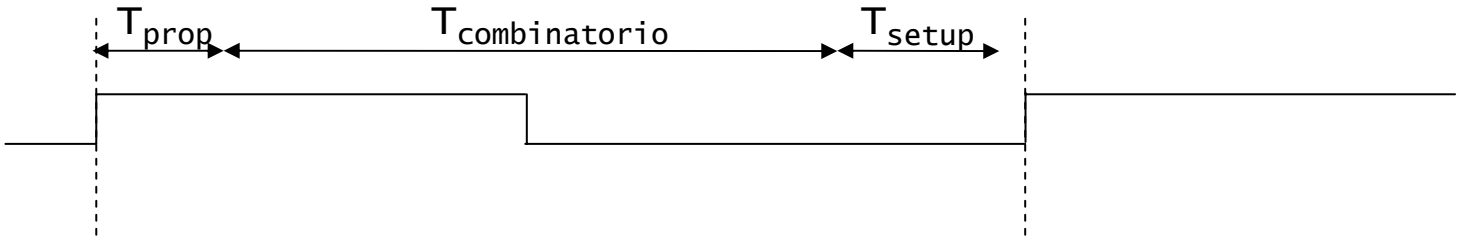
- Lettura asincrona risp. clock, scrittura attiva sul fronte del clock
- NB: forma semplificata (cfr. SRAM, DRAM, ecc.)

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





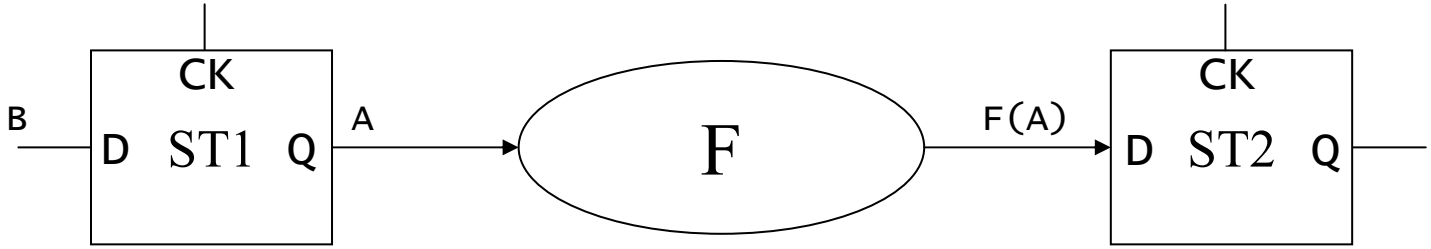
• Dopo $T_{prop} + T_{combinatorio}$, ingresso a ST2 è stabile: anticipo di almeno T_{clock}

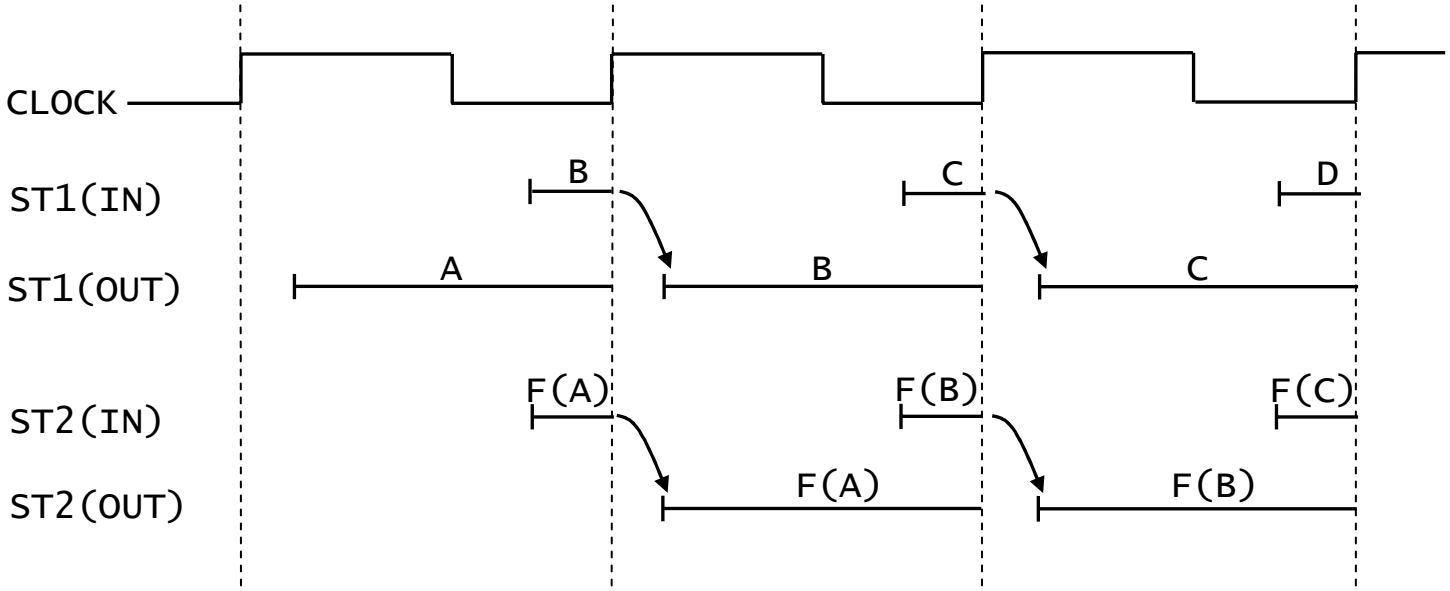
$$T_{clock} \geq T_{prop} + T_{combinatorio} + T_{setup}$$

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

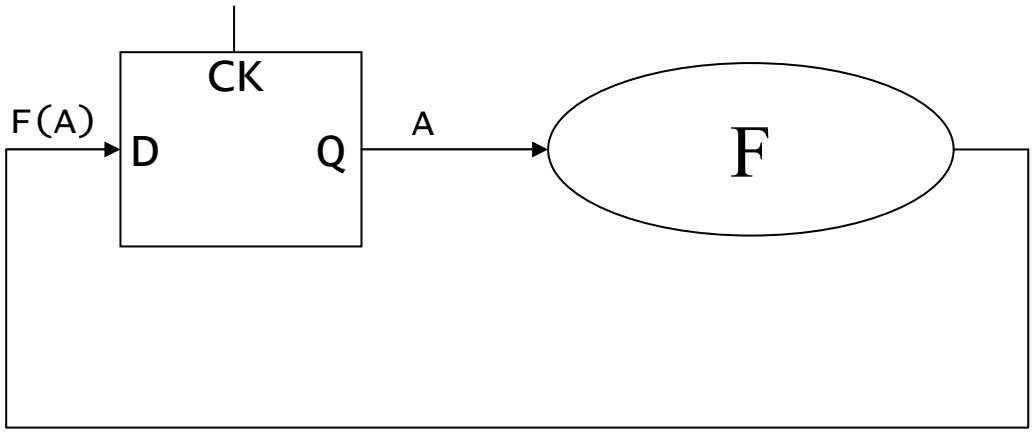
$$T_{prop} + T_{combinatorio} \geq T_{hold}$$

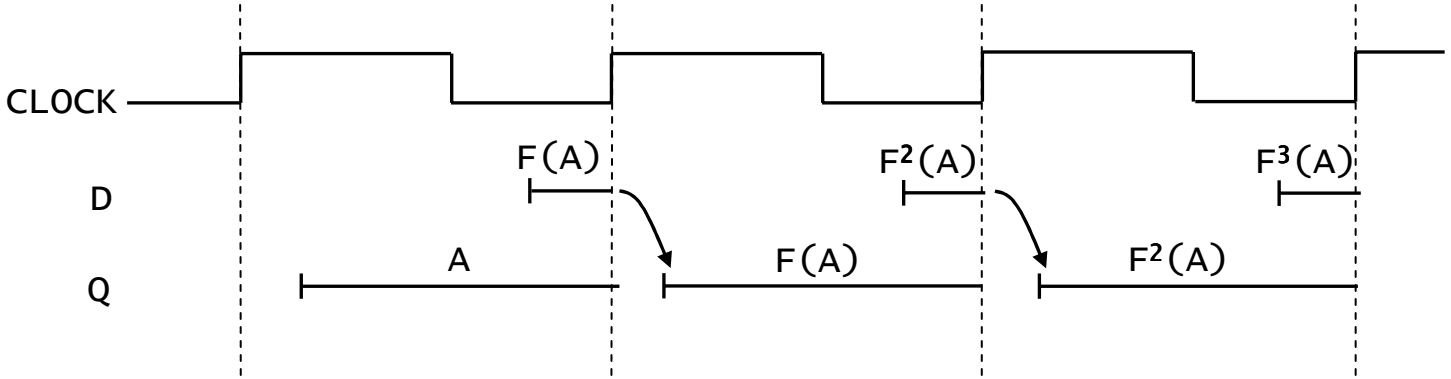
[verificato automaticamente perché $T_{hold} < T_{prop}$]





- 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)

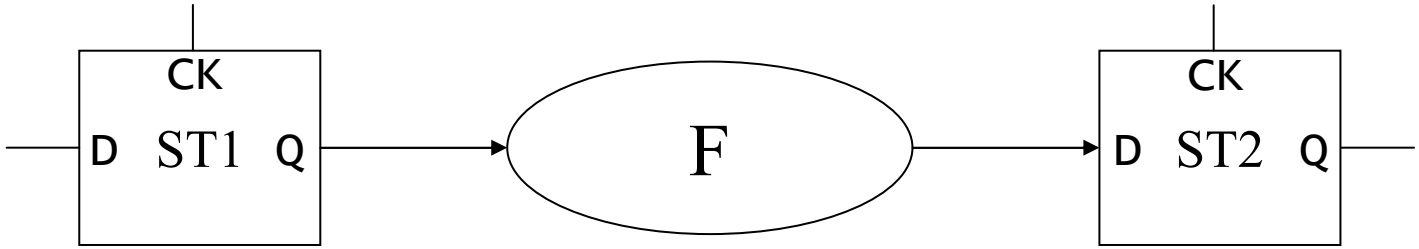
Un problema: lo sfasamento del clock

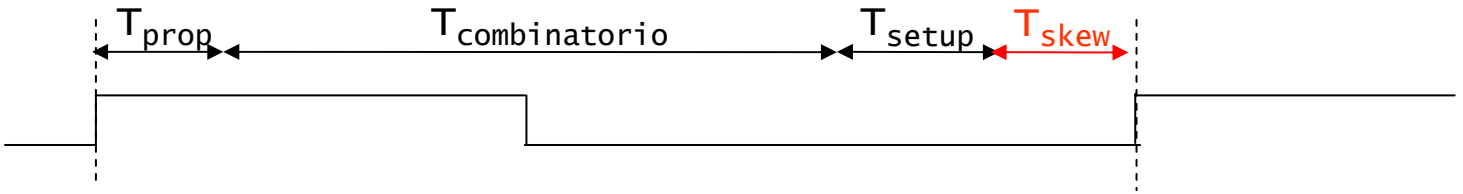
- Segnale di clock può percorrere cammini diversi per arrivare a elementi di stato

  ⇒ 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:

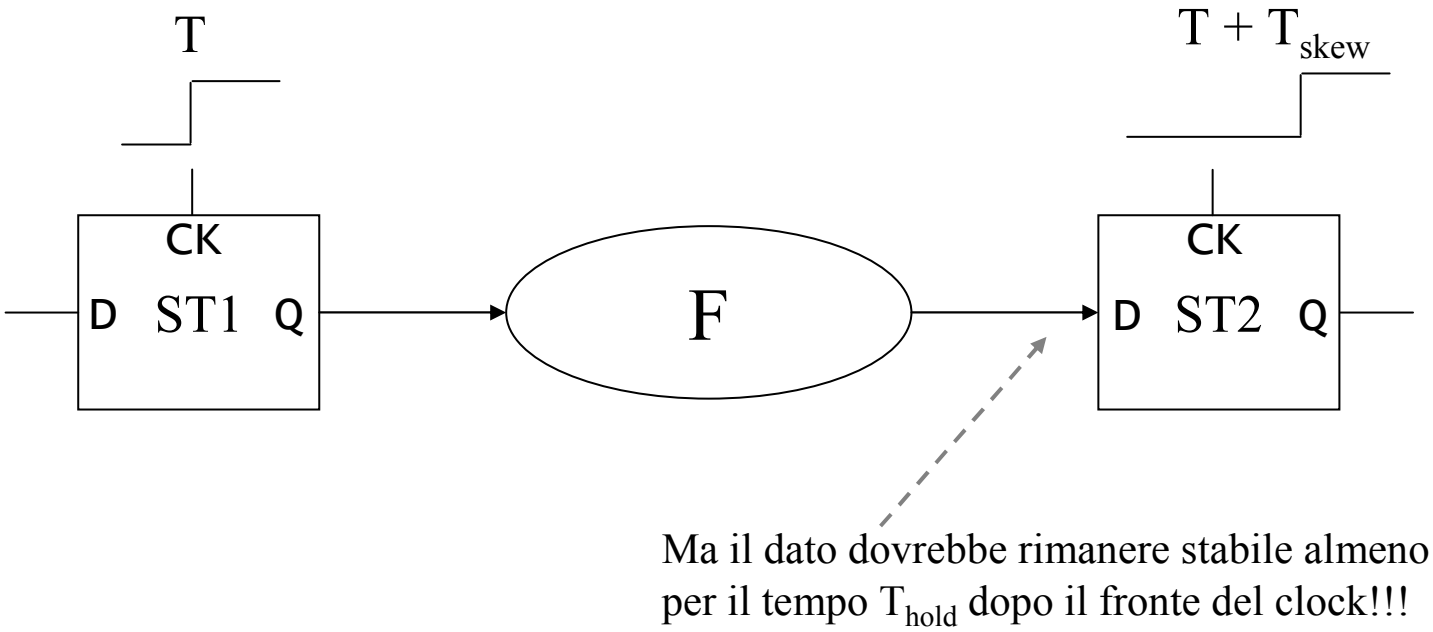




⇒ $T_{clock} \geq T_{prop} + T_{combinatorio} + T_{setup} + T_{skew}$

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!

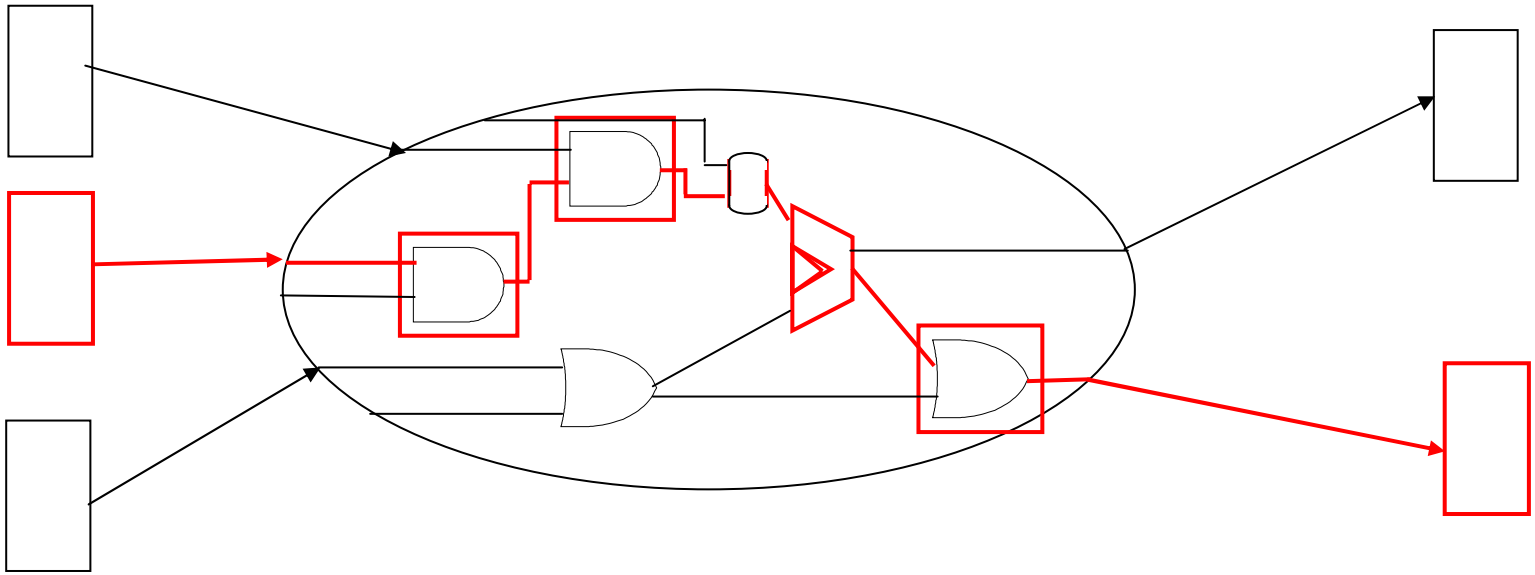


➡ $T_{prop} + T_{combinatorio} - T_{skew} \geq T_{hold}$

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!

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

Nel caso peggiore!

Le prestazioni sono quindi influenzate da scelte progettuali a livelli diversi:

- A livello di “calcolatori”: organizzazione delle componenti, parallelismo, ecc.
  [cfr. CPU singolo ciclo vs. multiciclo vs. pipeline]
- A livello di “reti logiche”: ridurre il numero di livelli di porte logiche
- A livello di elettronica digitale:
  attenzione al carico: evitare di pilotare grandi carichi con piccoli carichi
  famiglie logiche con prestazioni diverse
- A livello di tecnologie microelettroniche:
  riduzione tempi di setup, hold, propagazione, ecc.

… e i livelli non sono completamente indipendenti …

Specifica: macchine a stati finiti (FSM)

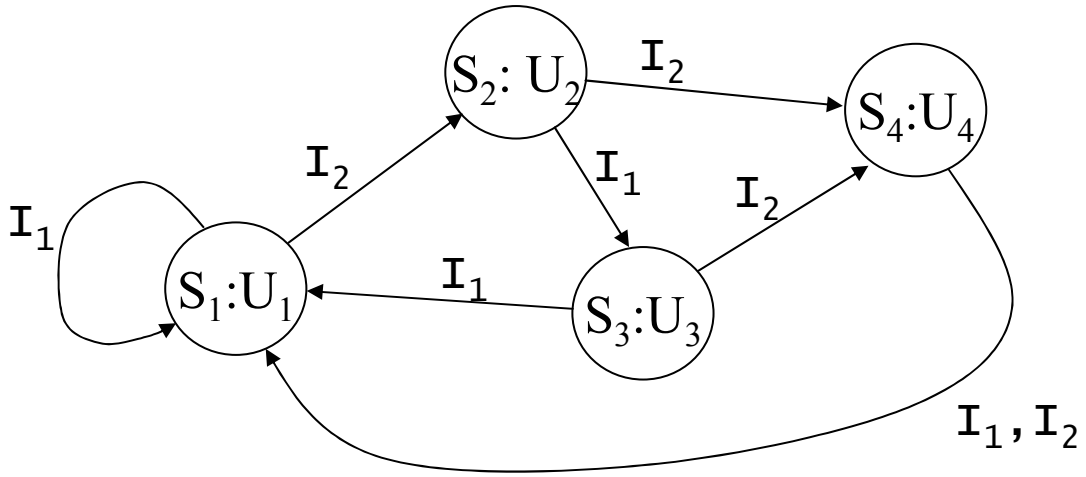
Occorre definire:

- 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 → S**
- Come si generano le uscite

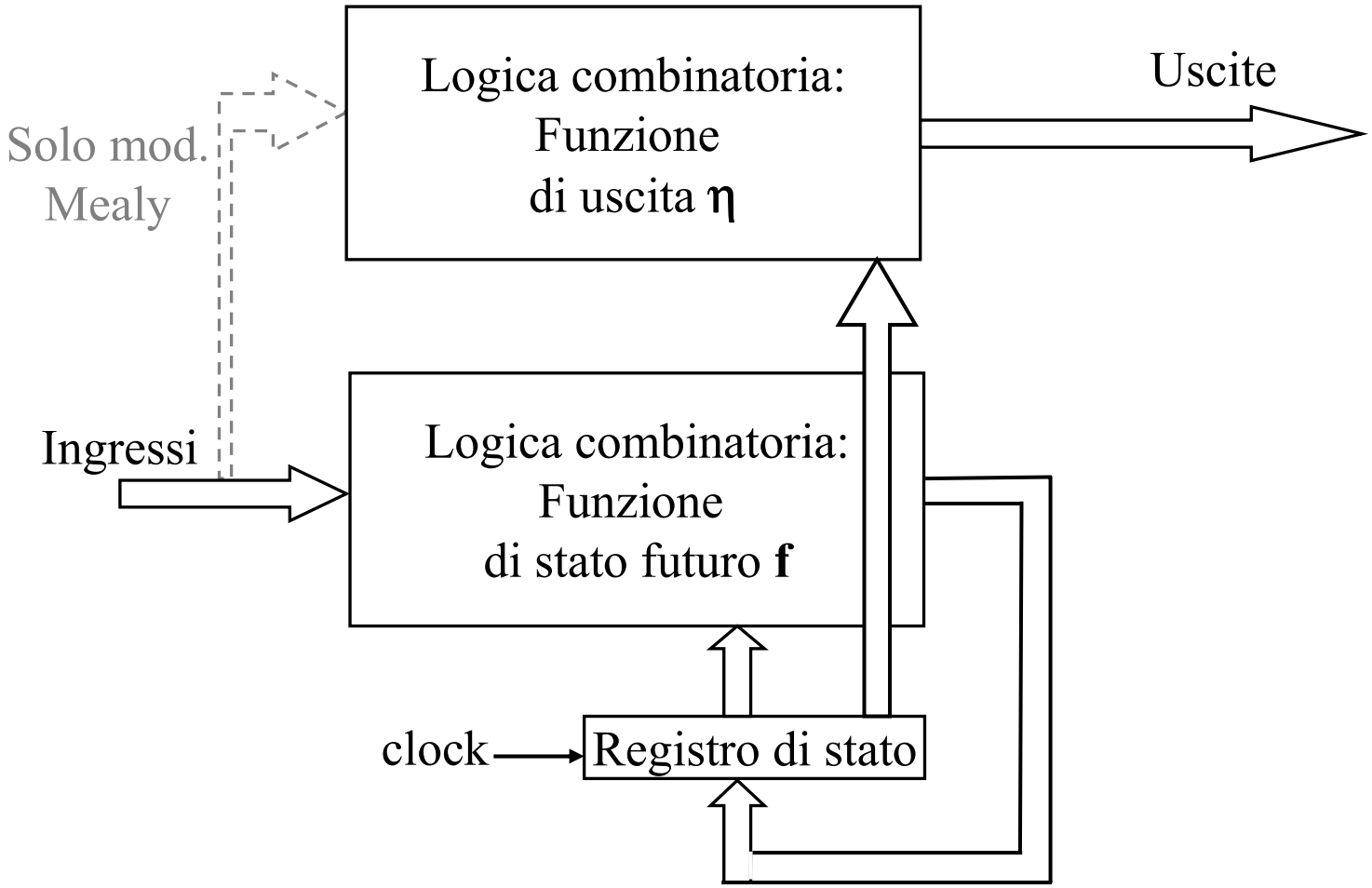
  funzione η **η: S → U** **Modello di Moore**

  η: S*I → U **Modello di Mealy**



[Modello di Moore]

Implementazione



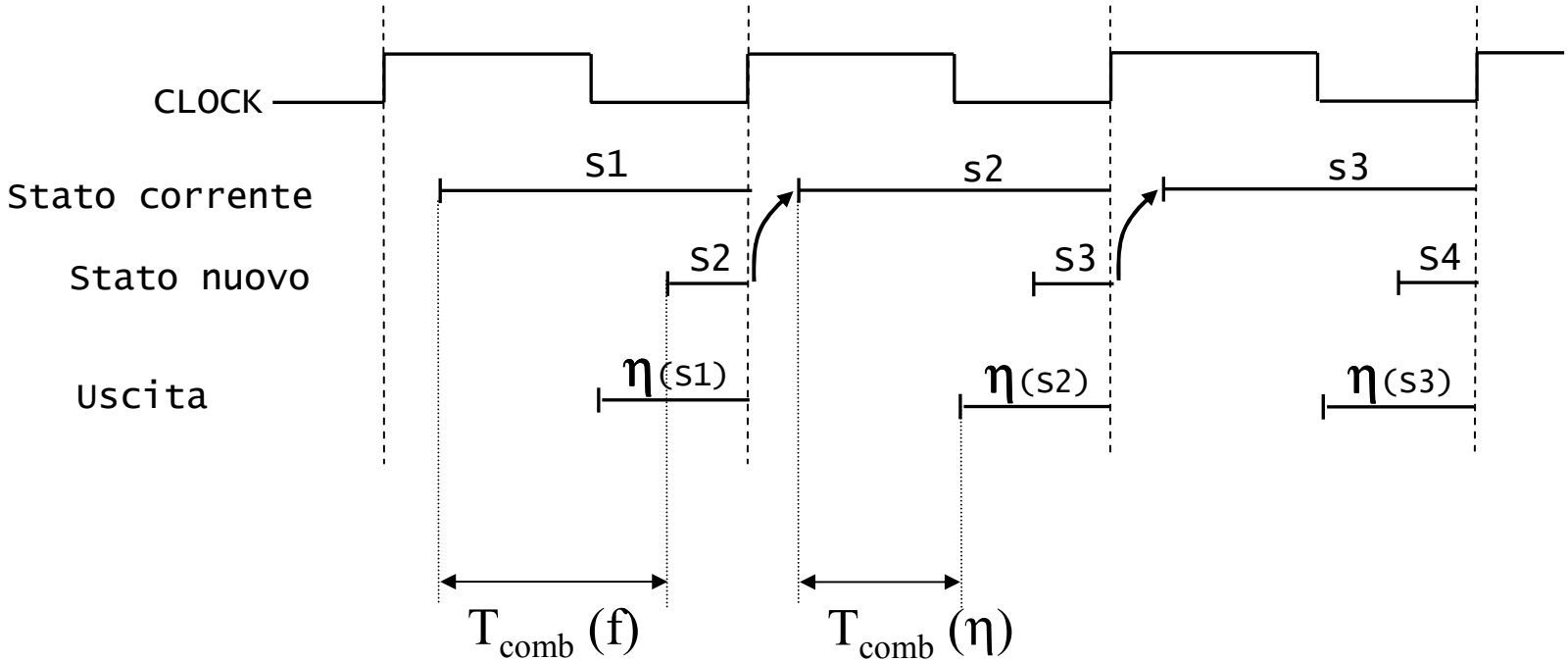
NB: logica combinatoria realizzata con PLA o ROM

• Mealy vs. Moore: sono 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)

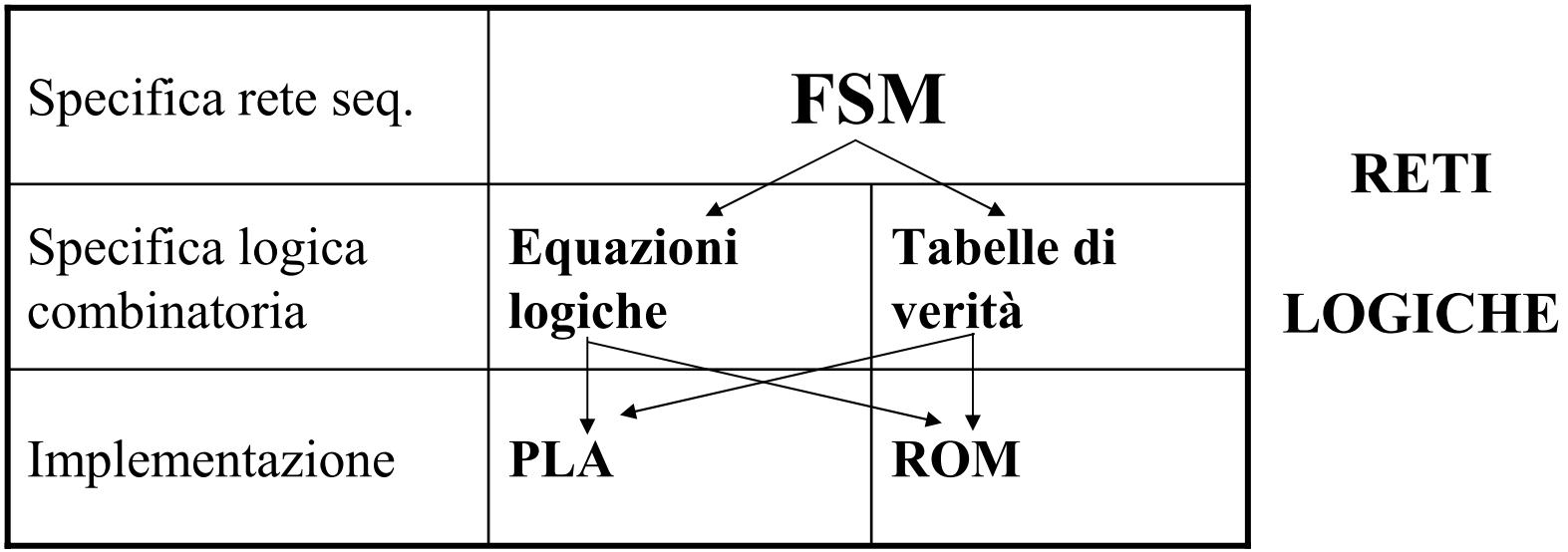
Adottato:
MOD. MOORE

• Temporizzazione



Quadro riassuntivo

CALCOLATORI



Realizzazione: Elettronica digitale - Microelettronica