Reti Logiche: Combinatorie e Sequenziali

Fabrizio Baiardi
f.baiardi@unipi.it

Livello Elettronico -Hardware

- In questa parte del corso vedremo come si costruiscono dei componenti elettronici che possono essere usati per definire il livello dei componenti eletttronici
- Reti logiche sono i componenti più semplici che possono essere composti in unità di elaborazione
- Reti logiche sono di due tipi
  - Combinatorie
  - Sequenziali

  entrambi implementano funzioni
- Componendo le funzioni delle reti si costruiscono unità

Matematica Finita

- Una qualunque funzione implementata da un componente elettronico è, per definizione, finita
- Quindi può essere sempre rappresentata come una tabella finita che specifica, per ogni input, qual è l'output corrispondente
- A partire da questa rappresentazione ne possono essere dedotte altre equivalenti ad esempio una espressione
- L'esistenza di una rappresentazione finita permette di risolvere problemi che, in generale, sono indecidibili
  - iniettiva
  - numero di passi
- L'astrazione mediante una funzione infinita viene talvolta preferita perché più semplice da manipolare

Matematica Finita Binaria

- Una funzione finita può mappare un qualunque dominio finito in un altro dominio finito
- Per costruire un componente elettronico ogni elemento di ognuno dei due domini deve essere rappresentato come una stringa binaria

Input	Output
A1	B2
A2	B7
A3	B2
A4	B7
A5	B1
A6	B8
A7	B4
A8	B2

Reti Logiche

- Daremo quello che viene detta una descrizione ai morsetti di una rete
  - Definizione degli input = n cifre binarie $I_1 \ldots I_n$
  - Definizione degli output = k cifre binarie $O_1 \ldots O_k$
  - Corrispondenza tra ingressi ed uscite F
- Rete Logica Combinatoria
  - F dipende unicamente da $I_1 \ldots I_n$
  - $F : I_1 x \ldots x\ I_n \rightarrow O_1 x \ldots x\ O_k$
  - F è indipendente dal tempo
- Rete Logica Sequenziale
  - F dipende da tutti gli input esterni ricevuti fino a questo momento
  - Gli input comprendono una ulteriore variabile detta lo stato interno della rete rappresentata su f bit $S_1 \ldots S_t$
  - La rete è descritta da due funzioni, F e Z, applicate a tutti gli input e che producono
    - Un output
    - Un nuovo stato che verrà usato con il prossimo input
  - Ognuna delle due funzioni è indipendente dal tempo

Rete Combinatoria

Ognuno degli input o degli output è una variabile binaria

Rete Logica Combinatoria

- Implementa una funzione finita che ha
  - input = n cifre binarie $I_1 \ldots I_n$
  - output = k cifre binarie $O_1 \ldots O_k$
- Vi sono 2^n input per ognuno si ha uno degli output

  $\Rightarrow$ Esistono $2^n * 2^k$ funzioni finite diverse su n input e k output
- Descrizione della funzione di interesse mediante
  - Tabella = un vettore di record in cui ogni record contiene una coppia <input, output> in cui
    - ogni input è diverso
    - ognuno dei 2^n input appare
  - Espressione =k espressioni,
    - una per ogni cifra dell'output
    - ogni espressione lavora sugli n bit in input
    - operatori AND, OR, NOT

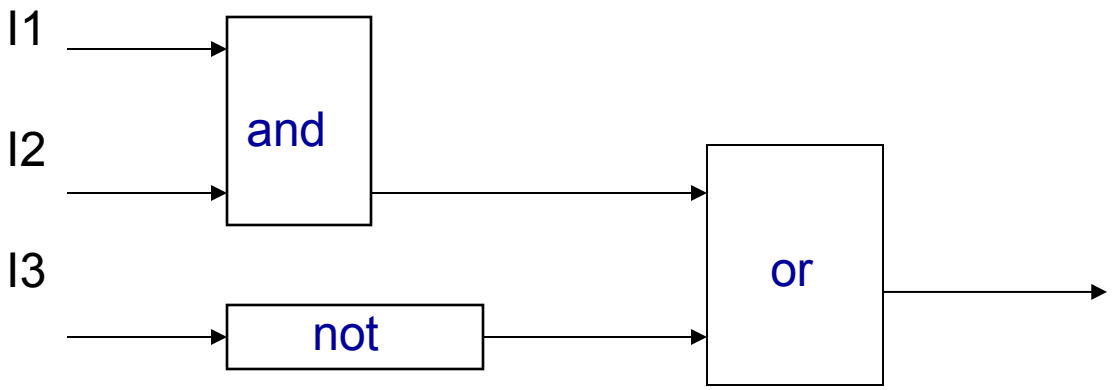
Operatori

- AND e OR sono operatori con
  - n input, n>1, $\in\{0,1\}$
  - 1 ouput $\in\{0,1\}$
  - AND=1 sse tutti gli input = 1, 0 altrimenti
  - OR=1 sse almeno un input =1 0 altrimenti
- NOT è un operatore con
  - 1 input $\in\{0,1\}$
  - 1 ouput $\in\{0,1\}$
  - NOT = 1 se input = 0, 0 altrimenti
- I tre operatori formano un insieme funzionalmente completo, i suoi operatori permettono di esprimere una qualunque funzione
- Altri insiemi {NOT, OR} , {NOT, AND} ,{NOR} {NAND}
- Ogni operatore può essere implementato da un circuito elementare

Realizzazione fisica degli operatori e reti

- Ogni operatore può essere realizzato con un circuito elettronico (porta)
- Quindi ogni funzione può essere costruita fisicamente mediante un circuito ottenuto componendo i vari circuiti = collegando il valore in uscita da un circuito all'ingresso di un altro
- I valori in input e quelli in output possono essere rappresentati come intensità di corrente o potenziale
- Massimo livello di dettaglio che useremo

OR(AND(I1, I2) NOT(I3))



Trasformazione da tabelle ad espressione

- Presa una funzione con n input ed 1 output espressa mediante una tabella (tabella di verità) esiste una regola che permette di trasformare la tabella in una espessione
  - Somma di prodotti = OR di AND (eventualmente NOT)
  - Prodotto di somme = AND di OR (eventualmente NOT)
- Somma di prodotti
  1. Il numero di prodotti (AND) è pari al numero di righe in cui la funzione vale 1
  2. Ogni prodotto comprende tutte le variabili
  3. Una variabile è negata se vale 0
- Prodotto di somme
  1. Il numero di somme(OR) è pari al numero di righe in cui la funzione vale 0
  2. Ogni somma comprende tutte le variabili
  3. Una variabile è negata se vale 1

Esempio di conversione

X1	X2	X3	O	
0	0	0	0	
0	0	1	1	AND(NOT(X1), NOT(X2),X3) =T1
0	1	1	0	
0	1	0	1	AND(NOT(X1), X2,NOT(X3))=T2
1	0	0	0	
1	0	1	0	
1	1	1	1	AND(X1, X2,X3) =T3
1	1	0	1	AND(X1, X2,NOT(X3)) =T4

- T1, T2, T3, T4 sono detti maxtermini poiché comprendono tutti gli input
- Espressione finale OR(T1, T2, T3, T4)

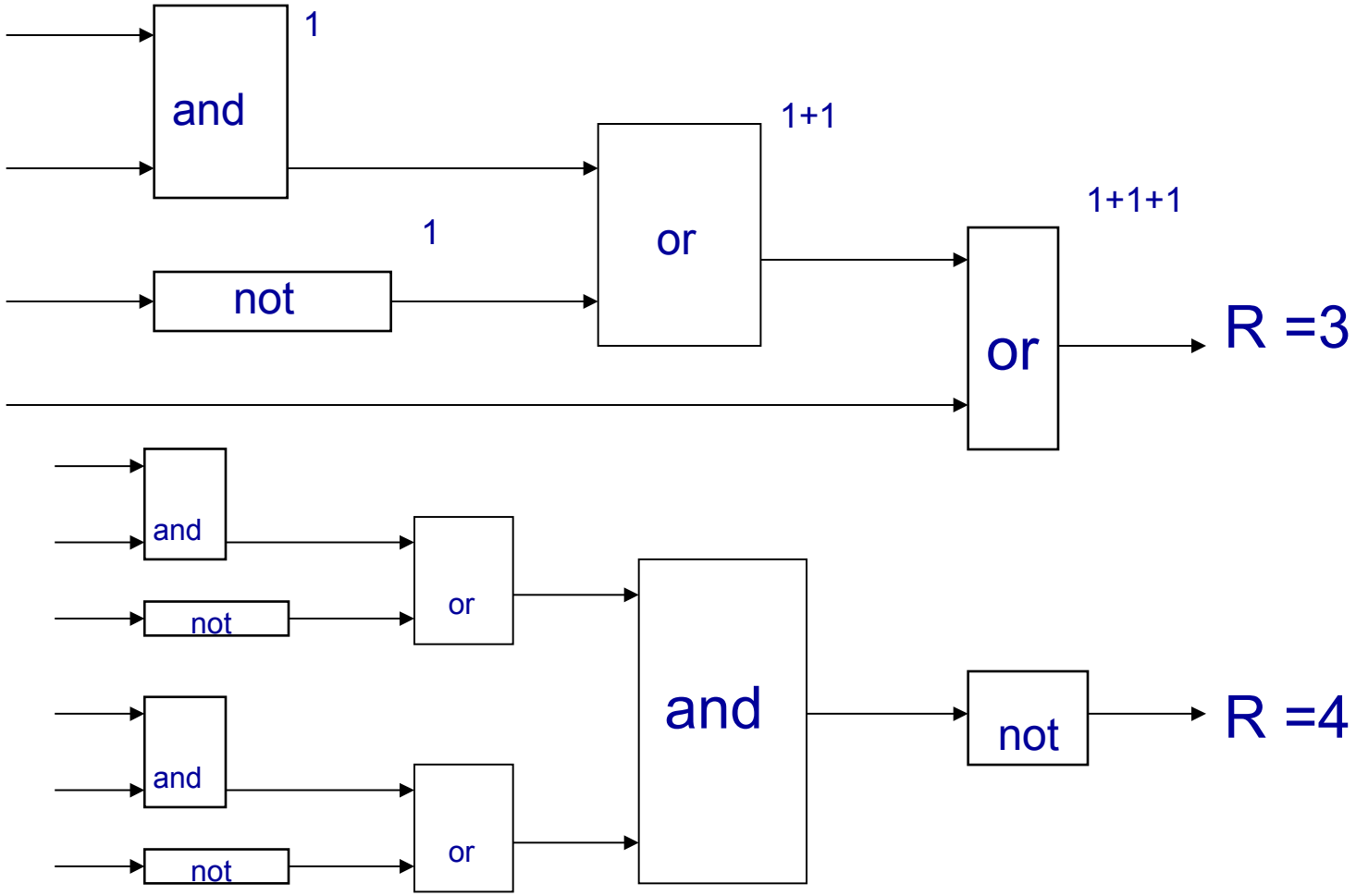
Insiemi funzionalmente completi

- Gli insiemi funzionalmente completi sono importanti perchè avendo dei circuiti per i vari operatori dell'insieme siamo sicuri di poter realizzare una qualsiasi funzione
- La cardinalità dell'insieme è importante poiché il costo di una rete dipende da due parametri
  - Numero di operatori
  - Tipo degli operatori
- A parità di numero di operatori il costo è minimo quando essi sono tutti dello stesso tipo
- Poiché esistono insiemi funzionalmente completi con un solo operatore è effettivamente possibile costruire una rete con un solo tipo di circuito elementare (nand o nor)

Espressioni, circuiti e ritardo

- Una proprietà estremamente importante di un circuito è il suo ritardo
- Il ritardo è definito come il tempo massimo tra
  - Istante in cui si applicano i valori di tensione/intensità corrispondenti ai valori di interesse (valori significativi)
  - Istante in cui il valore dell'uscita diventa costante
- Notare che agli input è sempre presente un valore di intesità o di tensione per cui è sbagliato dire che il ritardo è il tempo che passa tra quando c'è un valore in ingresso a quando c'è ne è uno in uscita perché
  - Ci sono sempre valori in ingresso
  - Ci sono sempre valori in uscita
- Ritardo di un cammino dipende dal numero di circuiti attraversati
  = profondità dell'espressione corrispondente alla funzione
  = numero massimo di circuiti sul cammino tra input e output
- Non si sommano i ritardi di cammini disgiunti
- Il ritardo di un circuito è dato dal massimo dei ritardi dei cammini

Esempi di ritardo



Nel caso generale

- Ritardo del primo circuito

  $\Delta OR + \max\{0, \Delta OR + \max\{\Delta AND , \Delta NOT \} \} =$

  $\Delta OR + \Delta OR + \max\{\Delta AND , \Delta NOT \} =$

  $2\ \Delta OR + \max\{\Delta AND , \Delta NOT \}$

- Ritardo del secondo circuito

  $\Delta NOT + \Delta AND + \max\{$

  $\Delta OR + \max\{\Delta AND , \Delta NOT \},$

  $\Delta OR + \max\{\Delta AND , \Delta NOT\}$

  $\}$ =

  $\Delta NOT + \Delta AND + \Delta OR + \max\{\Delta AND , \Delta NOT \}$

Differenze tra tabella e circuito

- La differenza sostanziale tra tabella e circuito è data dalla possibilità di calcolare il tempo necessario per il calcolo della funzione stessa
- La tabella è una specifica molto astratta perché definisce unicamente l'associazione tra ingresso e uscita stabilito dalla funzione, trasformando questa specifica in una espressione definiamo un calcolo che implementa l'associazione tra input ed output corrispondente
- Se introduciamo una porta per ogni operatore dell'espressione, possiamo trasformare l'espressione in un circuito ed associare ad ogni circuito un ritardo massimo
- Questo ritardo massimo definisce il caso pessimo del tempo necessario per il calcolo dell'espressione mediante il circuito costruito
- Poiché l'associazione tra funzione (tabella) ed espressione non è univoca dato che esistono più espressioni per una stessa funzione, ad ogni funzione possiamo associare circuiti diversi con tempi di calcolo diverso

Sintesi di reti combinatorie

- Il processo che parte dalla definizione delle specifiche della rete combinatoria e porta ad una espressione o ad un circuito viene detto **sintesi della rete combinatoria**
- Il passaggio inverso viene detto **analisi della rete combinatoria**
- Sono state definite delle procedure per ridurre il numero di operatori e quindi il numero di circuiti utilizzati per una rete
- Adottando le tecnologie attuali per la produzione dei circuiti,
  - la riduzione del numero dei circuiti non è importante
  - talvolta è meno costosa una espressione con più operatori ma più regolare
- Occorre evitare la sintesi che è un processo manuale e costoso, questo sarà uno dei nostri obiettivi nella costruzione delle unità di elaborazione
- La sintesi si evita usando componenti standard e operando sulla loro composizione e non sulla sintesi

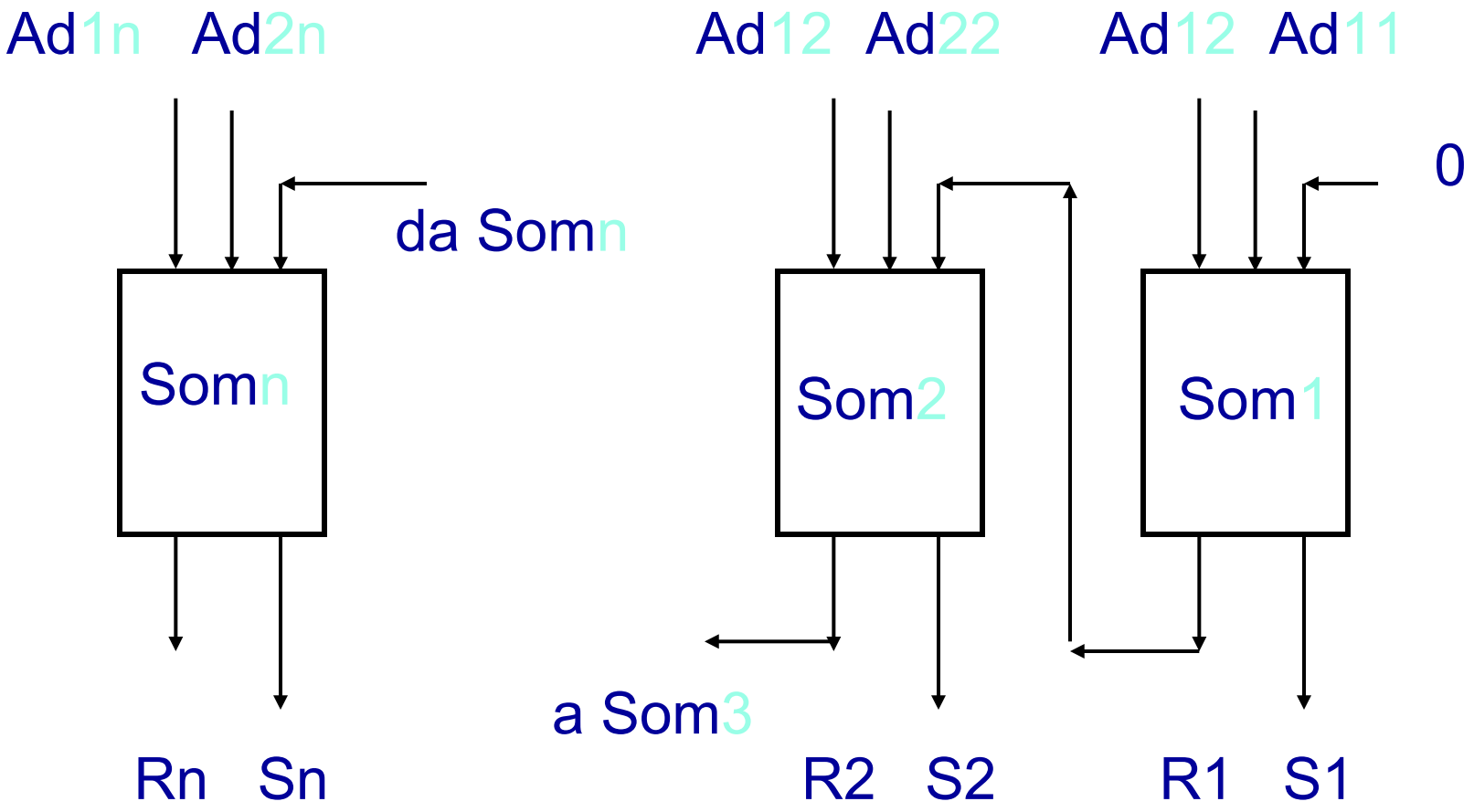
Esempio di sintesi

I1 I2 I3	S	R
0 0 0	0	0
0 0 1	1	0
0 1 1	0	1
0 1 0	1	0
1 0 0	1	0
1 0 1	0	1
1 1 1	1	1
1 1 0	0	1

S= OR(
AND(NOT(I1),NOT(I2), I3),
AND(NOT(I1), I2, NOT(I3)
AND(I1, NOT(I2),NOT(I3)),
AND(I1, I2, I3)
)

R= OR(
AND(NOT(I1), I2, I3),
AND(I1, NOT(I2),I3)
AND(I1, I2, I3)
AND(I1, I2, NOT(I3))
)

Addizionatore Seriale



Addizionatore Seriale

- Utilizzando n funzioni su 3 bit e componendole opportunamente posso sommare numeri grandi a piacere
- Si semplifica la sintesi che altrimenti sarebbe estremamente complessa perché per sommare due numeri di 32 bit la tabella avrebbe 2^{64}
- La semplificazione avviene a spese del ritardo perché il ritardo per sommare numeri di h bit è pari a $h*\Delta(Som)$ e quindi proporzionale alla dimensione dell'input
- Il ritardo è necessario per la propagazione del riporto, esiste un caso in cui il valore prodotto in uscita da Som_1 influenza l'uscita di Som_n
- Si può vedere che si può scoprire l'overflow analizzando il riporto generato dall'ultima e dalla penultima cifra
- Primo esempio di come sia possibile evitare la sintesi di una rete, invece di definire reti diverse per sommare numeri di 16, 32, 64 bit costruisco le reti componendo un numero diverso di addizionatori
- Evito la sintesi al prezzo di un ritardo maggiore

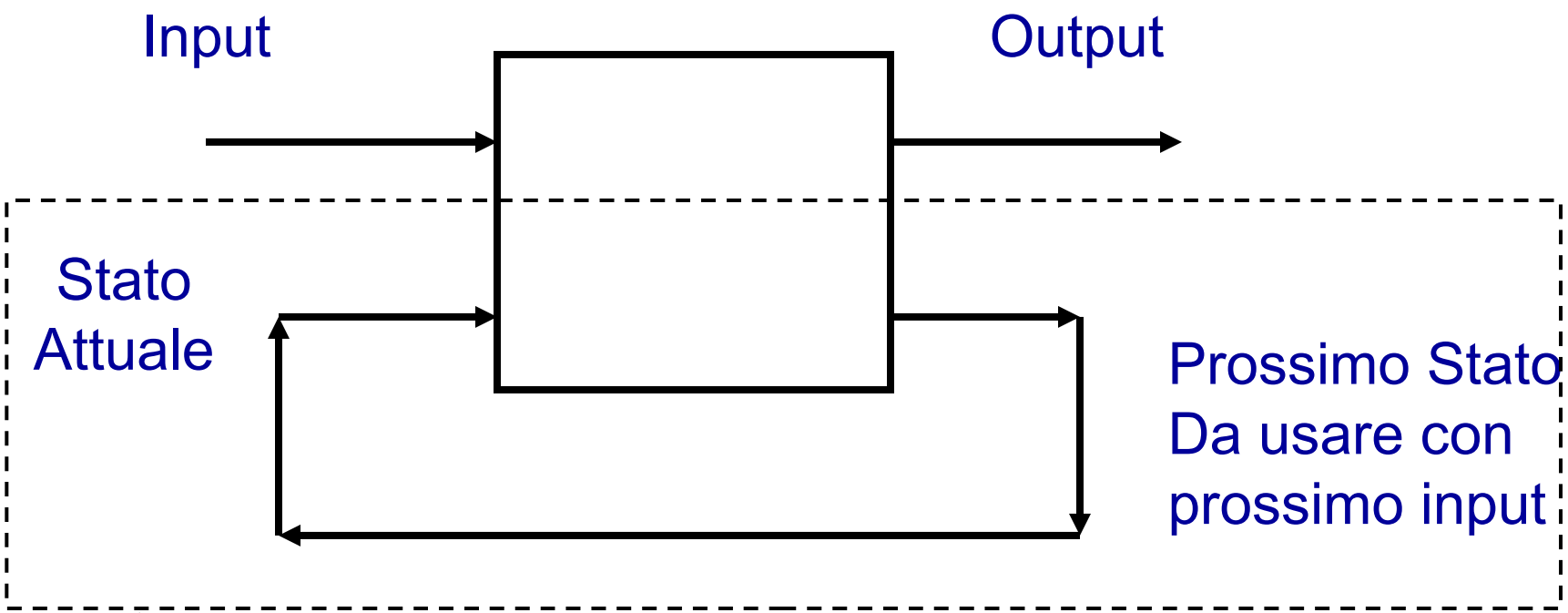
Rete Sequenziale

- L'output della rete dipende
  - da tutti gli input esterni ricevuti fino a questo momento
  - dalla storia degli input ricevuti fino ad un certo istante
- La rete ha in input una ulteriore variabile rappresentata su f bit $S_1 \ldots S_t$
  = lo stato interno della rete
- La rete è descritta da due funzioni, F e Z, applicate a tutti gli input e che producono
  - Un output
  - Un nuovo stato che verrà usato con il prossimo input
- Ognuna delle due funzioni è indipendente dal tempo

Rete Sequenziale

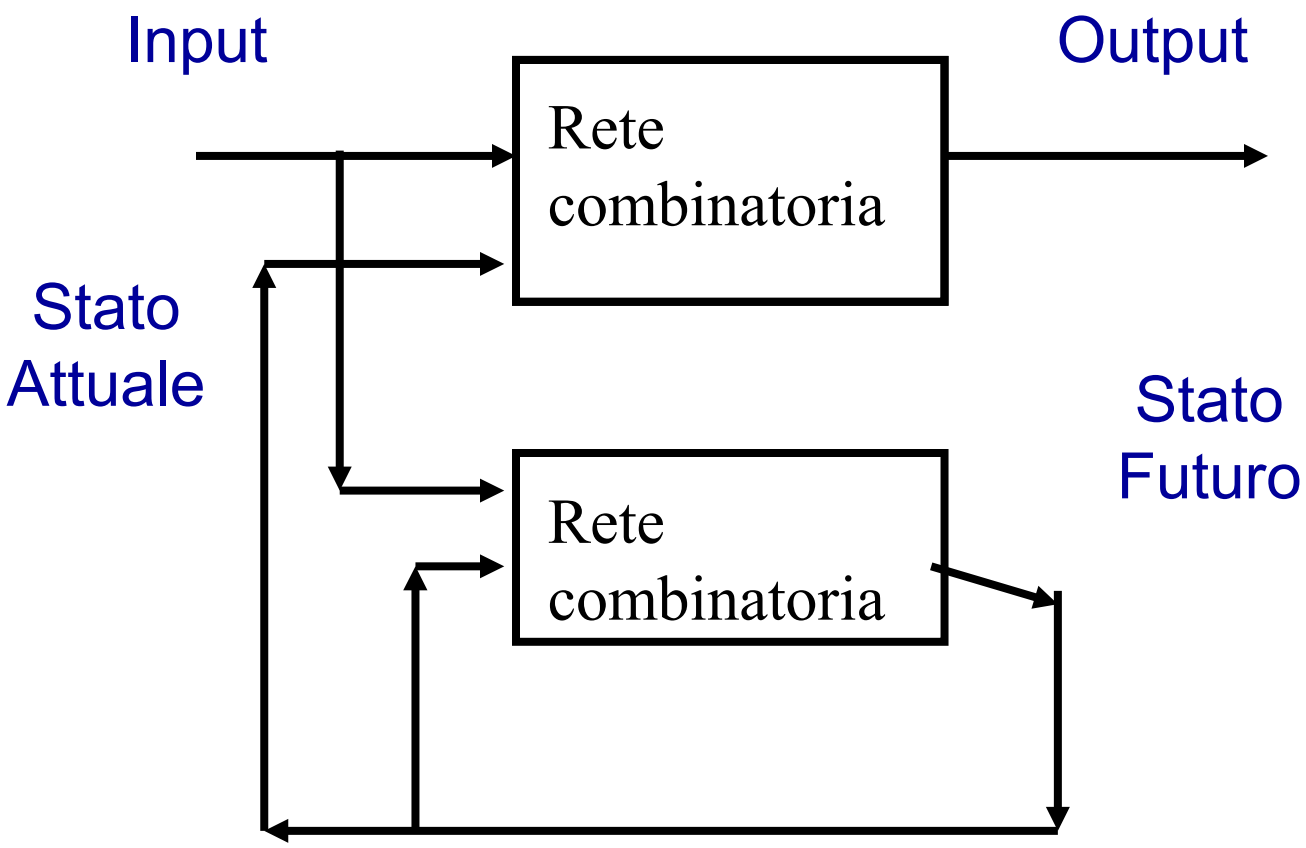
- Costruita a partire da due funzioni
  - Entrambe lavorano su input e stato
  - Una produce output
  - Una produce il nuovo stato
- Formalmente
  - Out: $I \times S \rightarrow O$ più bit, finita
  - St: $I \times S \rightarrow S$ più bit, finita
  - Lo stato prodotto al tempo Ti (dall'i-esimo input) viene utilizzato per calcolare output e stato al tempo Ti+1 (con l'i+1-esimo input)
  - Richiusura dello stato sulla rete
- Globalmente una rete sequenziale può essere visto come un automa a stati finiti in cui per ogni input si ha
  - Un output
  - Una transizione di stato

Rete sequenziale – Schema di principio



Uno dei problemi da risolvere è come implementare la richiusura dello stato

Rete sequenziale – Implementazione di principio



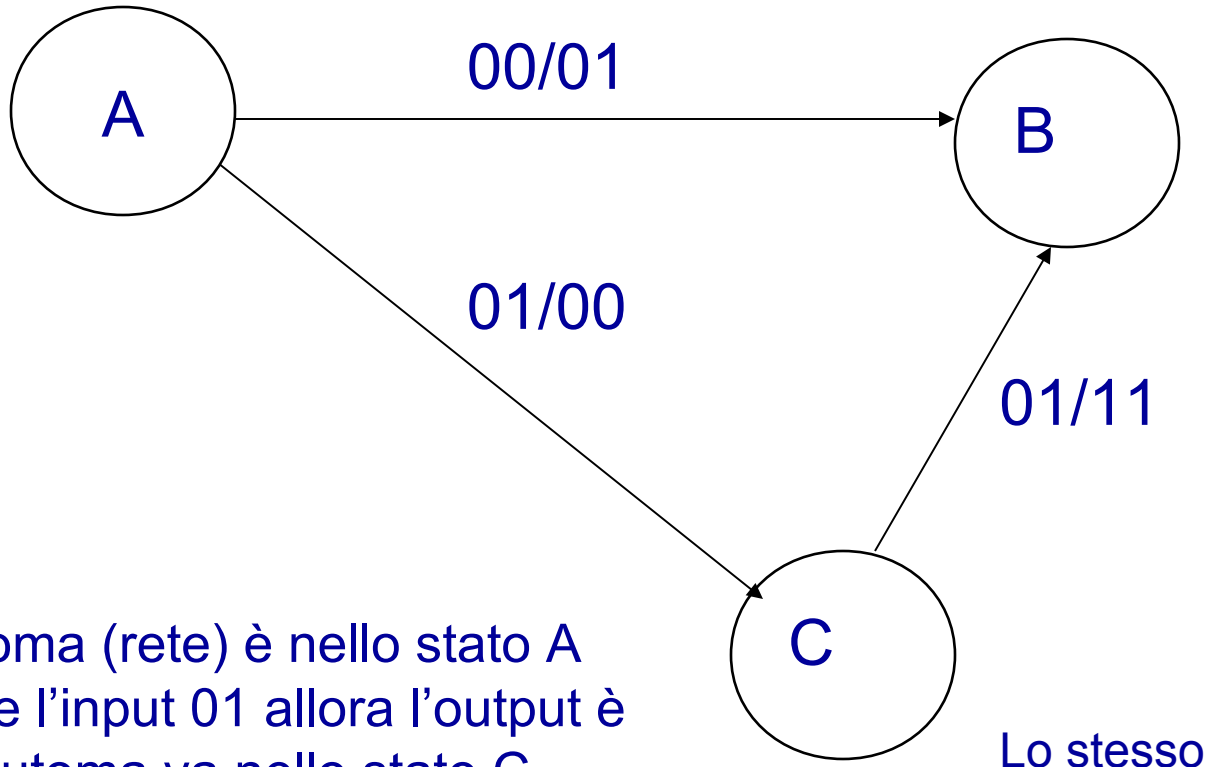
Automi a stati finiti

- La definizione di rete sequenziale richiama quella di automa a stati finiti
- Anche nel caso degli automi abbiamo una funzione che cambia in base allo stato in cui si trova l'automa
- Partendo da uno stato iniziale predefinito, lo stato raggiunto ad un certo istante dipende dalla sequenza di input presentati fino a quell'istante
- Un automa a stati finiti ricorda il passato ma solo un numero finito di informazioni, che dipendono dal numero di stati
- Degli input passati, potenzialmente in numero illimitato, si ricordano solo un numero finito di informazioni rappresentate dagli stati
- Gli input passati partizionati in un numero finito di classi di equivalenza

Reti Sequenziali e Automi a stati finiti

Per ogni stato esiste un arco per ogni input

L' arco uscente da un nodo mi dice dato un input quale è l'output e quale è il prossimo stato

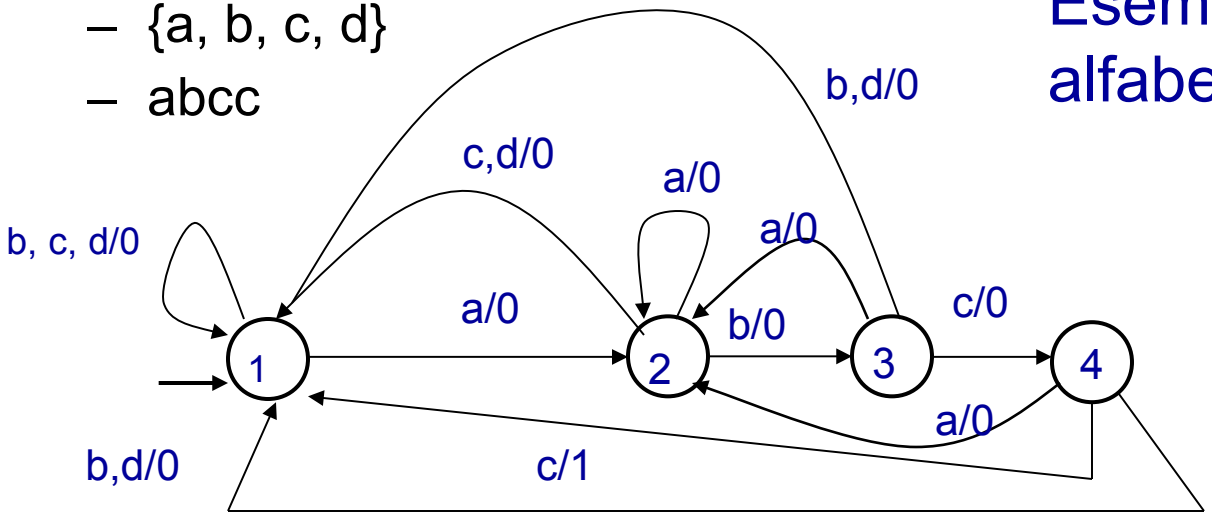


Se automa (rete) è nello stato A
E riceve l'input 01 allora l'output è 00 e l'automa va nello stato C

Lo stesso input di A produce un output diverso

Esempio di costruzione di unarete sequenziale

- Riconoscitore di sequenza
  - un alfabeto finito in ingresso
  - una stringa S finita di lunghezza k
  - Output
    - 1 se gli ultimi k caratteri sono uguali a S
    - 0 altrimenti
- Esempio
  - {a, b, c, d}
  - abcc



Esempio di rete su un alfabeto non binario

Esempio di rete - I

I	S	Output	Stato
a	1	0	2
b	1	0	1
c	1	0	1
d	1	0	1
a	2	0	2
b	2	0	3
c	2	0	1
d	2	0	1
a	3	0	2
b	3	0	1
c	3	0	4
d	3	0	1
a	4	0	2
b	4	0	1
c	4	1	1
d	4	0	1

Per definire la rete sequenziale occorre codificare

1. Input
2. Output
3. Stato

come stringhe binarie

- 4 input
- 4 stati
- 2 output

Occorrono quindi

- 2 bit per l'input
- 2 per lo stato
- 1 per output

Trasformazione su alfabeto binario

I	S	Output	Stato
00	00	0	01
01	00	0	00
11	00	0	00
10	00	0	00
00	01	0	01
01	01	0	11
11	01	0	00
10	01	0	00
00	11	0	01
01	11	0	00
11	11	0	10
10	11	0	00
00	10	0	01
01	10	0	00
11	10	1	00
10	10	0	00

a= 00 b= 01 c=11 d=10

1=00 2=01 3=11 4=10

Abbiamo quindi 4 valori in ingresso per ogni funzione

- Due codificano input
- Due codificano lo stato

Complessivamente 3 funzioni

- Due per i bit dello stato
- Una per il bit di output

Esempio di rete - III

I	S	Output	Stato1	Stato2
00	00	0	0	1
01	00	0	0	0
11	00	0	0	0
10	00	0	0	0
00	01	0	0	1
01	01	0	1	1
11	01	0	0	0
10	01	0	0	0
00	11	0	0	1
01	11	0	0	0
11	11	0	1	0
10	11	0	0	0
00	10	0	0	1
01	10	0	0	0
11	10	1	0	0
10	10	0	0	0

Funzioni

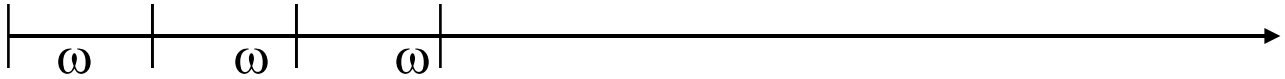
Output = AND(I1, I2, S1,NOT(S2)

Stato1= OR(
AND(NOT(I1), I2, NOT(S1), S2),
AND(I1, I2, S1, S2)
)

Stato2 = OR(
AND(NOT(I1), NOT(I2), NOT(S1), NOT(S2)),
AND(NOT(I1), NOT(I2), NOT(S1), S2),
AND(NOT(I1), I2, NOT(S1), S2),
AND(NOT(I1), NOT(I2), S1), S2),
AND(NOT(I1), NOT(I2), S1, NOT(S2))
)

Memorizzare lo stato

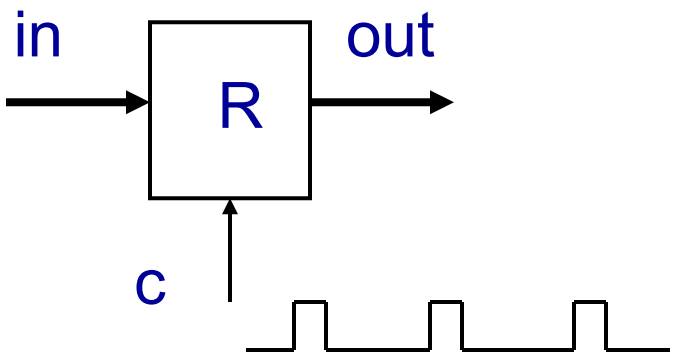
- Per definire completamente come è realizzata la rete sequenziale dobbiamo implementare la richiusura, cioè stabilire una connessione tra
  - lo stato attuale (calcolato con l'ingresso attuale) e
  - lo stato successivo (quello che verrà usato con il prossimo ingresso)
- Per stabilire questa connessione dobbiamo imporre dei vincoli sugli istanti in cui gli ingressi arrivano al sistema, in particolare dobbiamo imporre che ogni ingresso
  - arrivi dopo un intervallo di tempo fisso dal successivo ω = ad ogni intervallo un nuovo input
  - rimanga costante per tutto il tempo ω



ing1 ing2 ing3 ing4 ….

Implementazione Sincrona

- Un comportamento che rispetta i vincoli precedenti viene detto sincrono
- Abbiamo una sincronizzazione tra il mondo esterno e la nostra rete
- ω è un parametro che caratterizza la rete
  - viene chiamato il ciclo di clock della rete
  - dipende sia dal mondo esterno che dalla rete stessa
- Possiamo realizzare una rete mediante un elemento di memoria o di registro

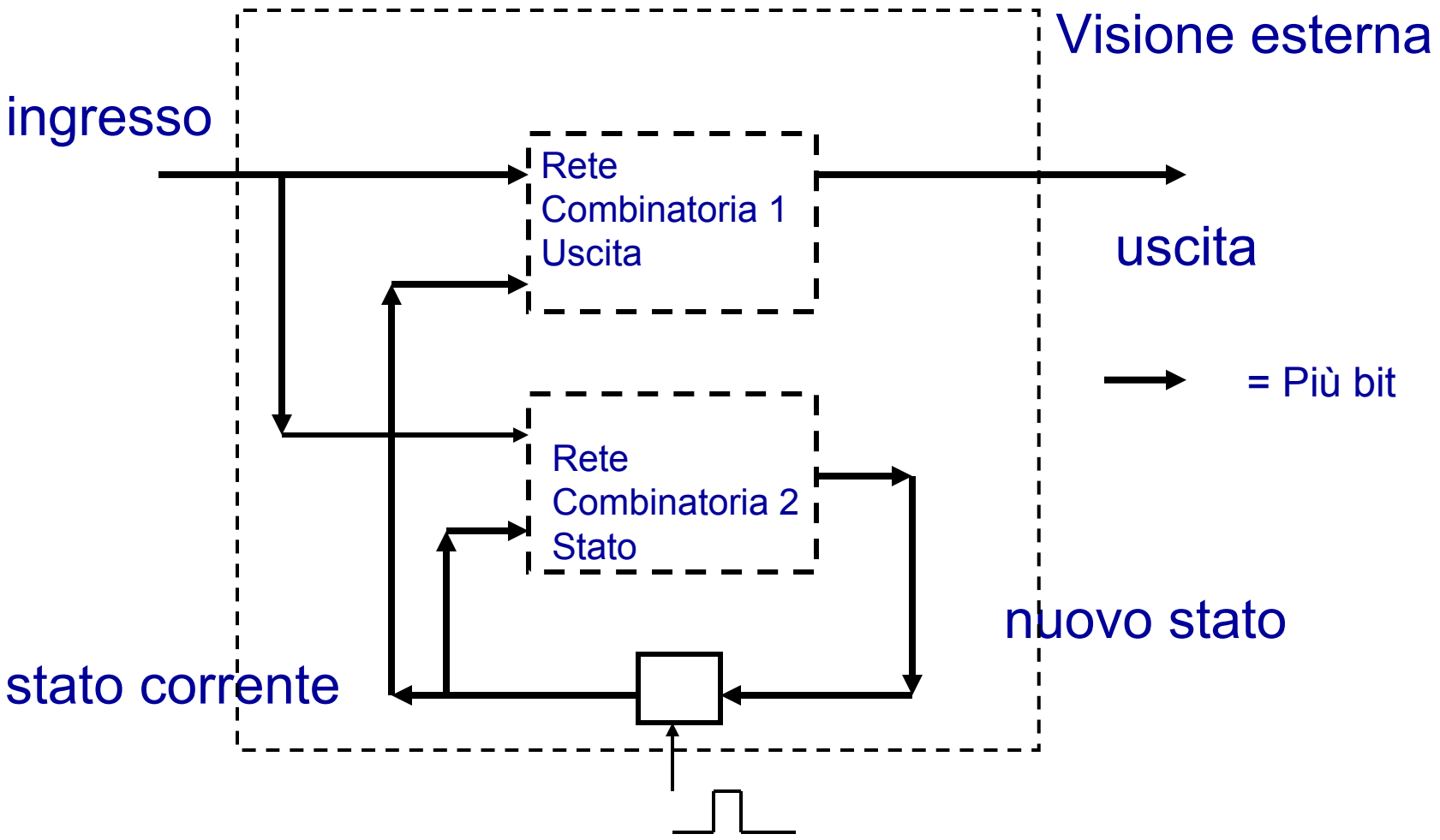


in e out sono ingressi con lo stesso numero di bit >0

c è un ingresso impulsivo di un bit

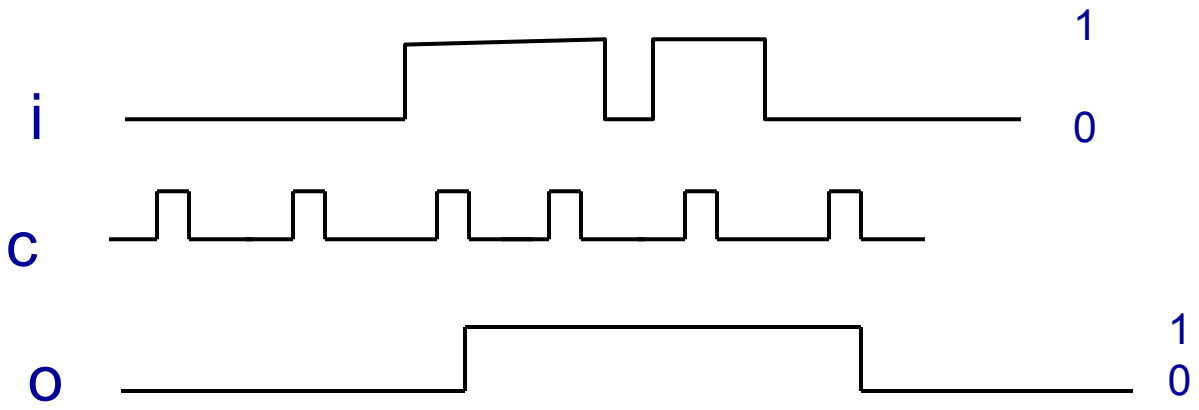
- normalmente è a zero
- resta a 1 per un tempo predefinito

Realizzazione sincrona di Rete Sequenziale



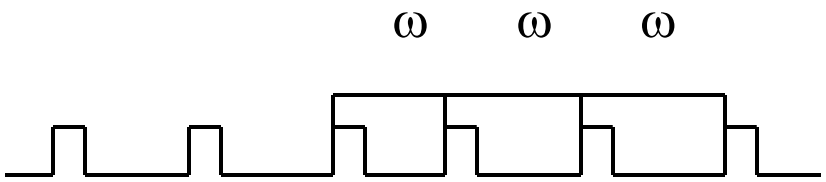
Elemento di memoria o registro

- Si assume che quando l'impulso è presente = (c=1) i valori in ingresso non cambino
- Quando l'impulso è presente il valore in ingresso viene
  1. memorizzato
  2. trasmesso in uscita, al termine dell'impulso, fino a quando non arriva un nuovo impulso
- Ogni valore in ingresso a cui non corrisponda un impulso non influenza il valore ricordato e trasmesso in uscita dal registro.



Ciclo di clok di una rete sincrona

- In realtà non abbiamo un impluso isolato ma quello che si chiama un treno di impulsi
- Una sequenza infinita di impulsi con distanza fissa ω tra due impulsi successivi



- ω = ciclo di clock
- Inverso di ω è una frequenza misurata in hertz (1/t)
- ω è anche il tempo tra due ingressi successivi della rete
- ω deve essere maggiore del ritardo delle due reti combinatorie che compongono la rete, detta δ la durata dell'impulso abbiamo
  - ← $\omega = \alpha + \delta$
  - ← $\alpha \geq \max(\Delta 1, \Delta 2)$ dove $\Delta 1$ e $\Delta 2$ sono i ritardi delle due reti

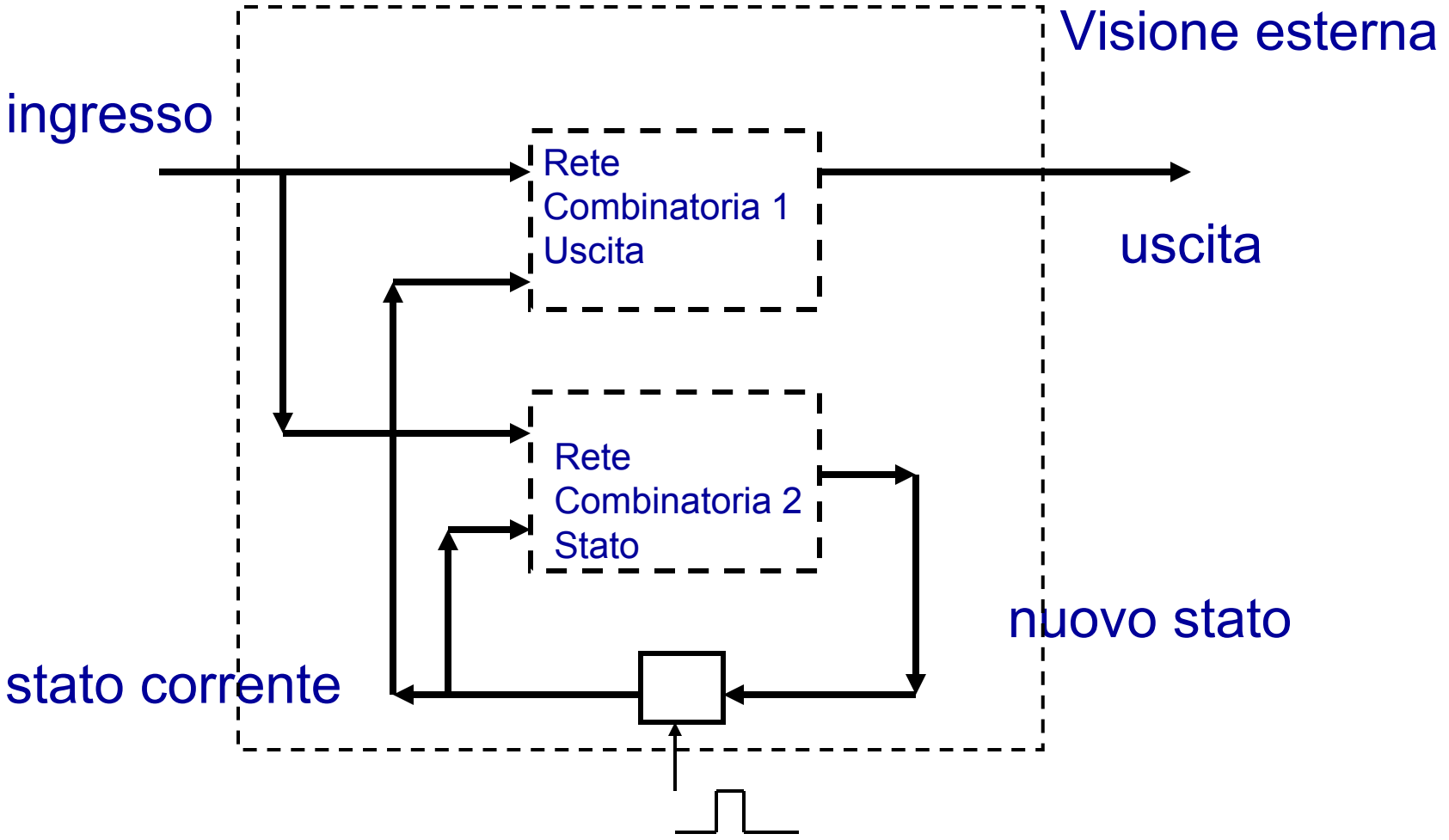
Vincoli sulla rete sequenziale

- Se non sono in grado di costruire due reti combinatorie con $\Delta 1$ e $\Delta 2$ tali che $\alpha \geq \max(\Delta 1, \Delta 2)$, allora non posso costruire una rete sequenziale in cui il ciclo di clock = ω
- A questo punto ho due sole soluzioni
  - Cerco di rallentare il mondo in modo che mandi due input successivi ad un intervallo > ω
  - Modifico la tecnologia di costruzione delle reti combinatorie in modo da diminuire il ritardo massimo delle due reti
- Notare che si considera il massimo dei due ritardi poiché le due reti combinatorie lavorano in parallelo
- Poichè ognuna delle due reti è costruita mediante una rete per ogni bit, abbiamo che $\max(\Delta 1, \Delta 2) =$

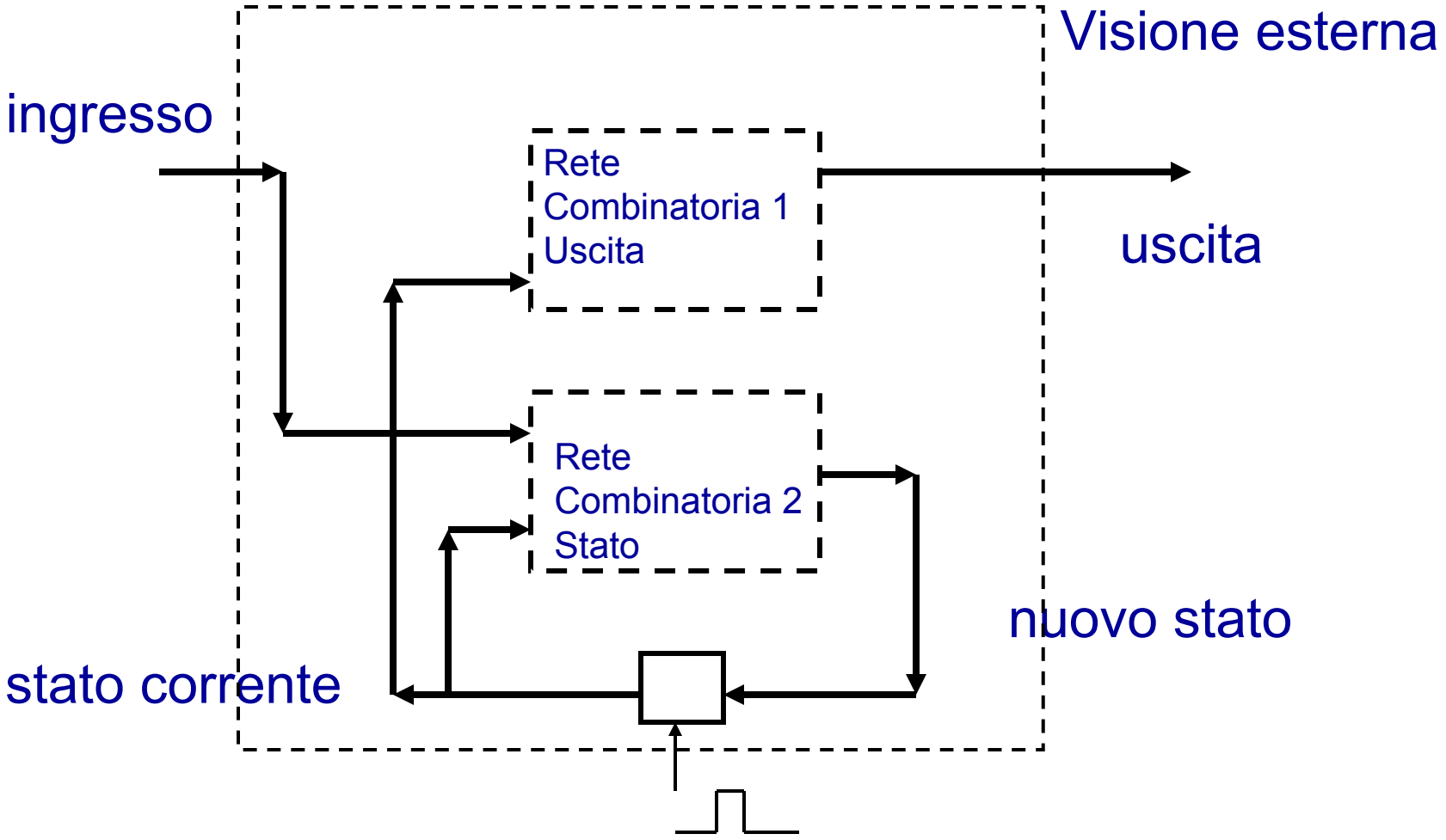
  $\max(\Delta 1i, i \in 1..n, \Delta 2j, j \in 1..k))$

  dove n e k sono il numero di bit di output delle due reti

Rete Sequenziale di Mealy

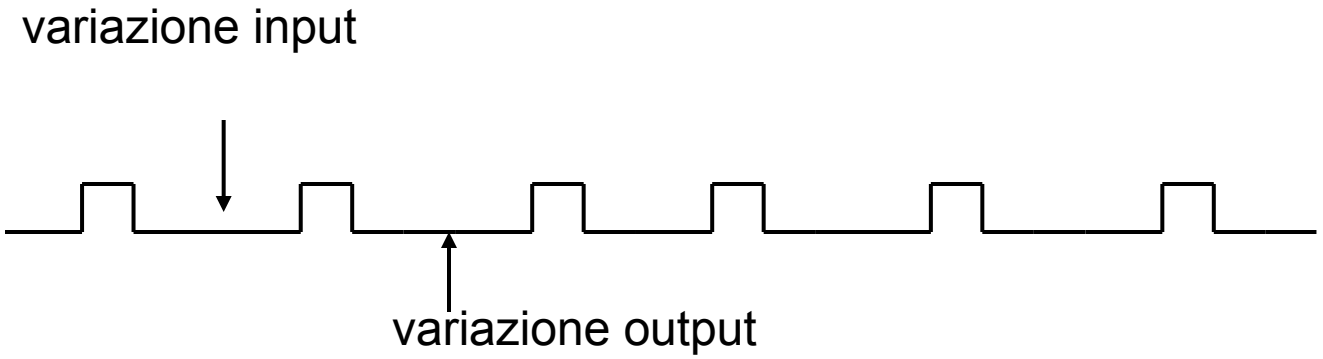


Rete Sequenziale di Moore



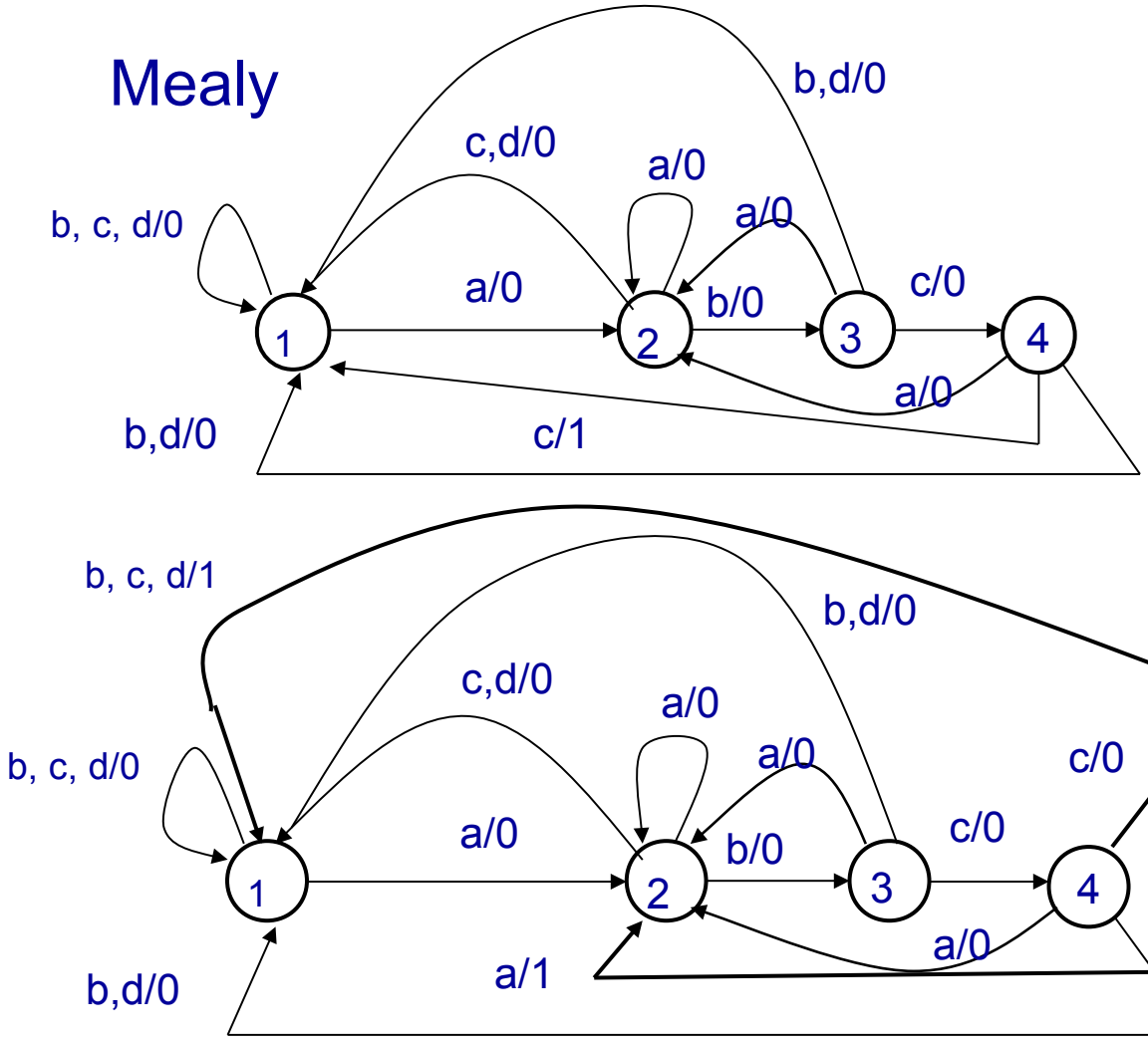
Modello di Moore

- In questo modello di rete, l'output ad un certo ciclo non dipende dall'input dello stesso ciclo
- Quindi una rete di Moore è in grado di calcolare il proprio output senza conoscere il proprio input a quel ciclo
- Ogni variazione dell'input si ripercuote sull'output con almeno un ciclo di clock di ritardo



- In una rete di Mealy, invece, ogni variazione si ripercuote immediatamente sull'uscita
- Le reti di Moore sono fondamentali per il progetto di unità di elaborazione

Moore e Mealy



L'indipendenza dell'output da Input fa si che in ogni stato si possa associare l'output che si avrà qualunque sia l'input

Date una rete di Mealy ed una di Moore equivalenti (risolvono lo stesso problema) la rete di Moore

1. produce la soluzione un ciclo di clock dopo di Mealy
2. ha più stati

Reti sequenziali asincrone

- Si può dimostrare che aumentando il numero di bit per la codifica dell'alfabeto si possono produrre reti sequenziali asincrone
- Queste reti non hanno un ciclo di clock ma, dato un input producono una sequenza di stati e di uscite fino a raggiungere uno stato interno detto stabile = stato in output uguale a quello in input
- Funzionano purchè il tempo tra due input diversi sia maggiore del tempo di stabilizzazione della rete
- Per molto tempo sono state solo di interesse teorico, ora sono tornate di interesse per cercare di capire se si possa costruire un calcolatore senza clock che, in teoria, potrebbe essere più veloce di uno con clock