Esercitazione RSS

FONDAMENTI DI INFORMATICA B

DIDATTICA A DISTANZA

Università degli studi di Parma
Dipartimento di Ingegneria dell'informazione

tutore: Ing. A. Tibaldi

6 maggio 2002

Indice

1 Macchine con memoria

Nella precedente esercitazione sono state prese in considerazione reti in grado di produrre uscite sotto forma di *segnali binari* applicando agli ingressi di queste per un periodo di tempo non specificato delle configurazioni **stabili** degli ingressi.

È stata assunta implicitamente l'ipotesi che l'ingresso a tali reti non avesse la possibilità di cambiare di valore ed è stata analizzata la funzionalità dello schema logico "ai morsetti", senza entrare nel dettaglio di alcuna considerazione temporale.

In realtà l'andamento dei segnali elettrici a livelli nel tempo ha delle conseguenze non trascurabili sulla progettazione delle reti, specialmente se queste devono tenere traccia del loro stato al fine di manifestare un comportamento "dinamico" in risposta agli stimoli osservati.

1.1 Fenomeni transitori

Nella realtà un insieme di segnali elettrici in ingresso ad una rete può manifestare variazioni del livello di tensione di ciascun segnale di diversa entità ed in istanti temporali differenti gli uni dagli altri.

Il livello di tensione associato ad un nodo di un circuito è in relazione alla capacità che quel nodo manifesta verso massa ed alla carica che tale nodo ospita in condizioni di isolamento.

Quando un segnale varia dunque, manifesta un andamento continuo e la variazione di più segnali avviene per quanto si possa cercare di sincronizzarli in un ordine che generalmente non è possibile prevedere.

1.2 Alee

Ipotizziamo per un attimo che l'attraversamento di una porta NAND a due ingressi sia istantaneo da parte dei segnali che si presta ad elaborare.

In tal caso se uno dei due segnali è al livello logico basso ed un altro è al livello logico alto ed è previsto per il corretto funzionamento del circuito che entrambi debbano cambiare di valore allo stesso istante (dunque teoricamente l'uscita dovrebbe permanere ad un livello logico basso), è drammatico pensare quali problemi possa portare il fatto che non è possibile avere una variazione contemporanea del livello.

Supponiamo infatti che per primo vari il segnale a livello basso, in tal caso essendo l'attraversamento della porta NAND istantaneo è chiaro che in uscita si presenterà il valore logico "uno" per il tempo necessario all'altro segnale che era alto a portarsi al livello zero.

Questo picco non voluto di segnale affermato all'uscita prende il nome di *alea statica di zero* dell'and e può compromettere il comportamento dell'intera rete.

Per correggere questo comportamento si adotta per esempio nel progetto di una RLC a SP una copertura ridondante con *raggruppamenti rettangolari* che facciano contenere ogni coppia di "uni" adiacenti in uno stesso raggruppamento.

1.3 Tempo di elaborazione e livelli

In realtà il tempo di elaborazione di un segnale da parte di una porta logica non è nullo e si può assumere ad un valore τ fissato specifico per ogni singola realizzazione della porta.

Nel caso semplice di una rete combinatoria a più livelli, se un'alea al livello della logica di ingresso permane per un tempo superiore a quello del tempo di attraversamento del blocco di livello due che è posto immediatamente in cascata a quello manifestante l'alea, l'effetto è quello di un'errata configurazione degli ingressi alla logica di secondo livello.

Questo potrebbe chiaramente compromettere tutto quello che segue e la correttezza dell'elaborazione svolta dalla rete.

1.4 Ritardo e memorizzazione

In realtà una porta può funzionare come elemento di ritardo e questo è alla base del concetto di memorizzazione nelle reti sequenziali *asincrone.*

La memorizzazione avviene attraverso l'inserimento in un elemento di memoria di un valore binario codificante una certa quantità di informazione. Le RSS contengono un numero anche notevole di questi elementi di memorizzazione; assegnata loro un'etichetta identificativa che potrebbe anche essere numerica, lo "stato" di una rete con memoria ad un certo istante temporale è da considerarsi come l'insieme ordinato dei valori H o L assunti da tutti gli elementi di memorizzazione contenuti nella rete.

Noi ci occuperemo delle reti sequenziali (dunque con memoria) sincrone, per le quali sono fissati degli *istanti cadenzati di sincronismo* che sono teoricamente di durata nulla in quanto istanti e determinano il momento in cui deve avvenire la memorizzazione del nuovo stato rappresentato dall'insieme di segnali binari elaborati dalla ***state function*** all'interno degli elementi di memoria.

Per semplificare il procedere del discorso prenderemo ad esempio il caso di **campionamento** di un segnale binario D a livello, ovverosia quando un altro segnale binario C assume per esempio il valore 1, per poi mantenere il valore del livello campionato quando il segnale C ritorna a 0.

E' questo il caso di un **latch D** campionato da un livello che ha natura **asincrona**. Questa componente ha due ingressi C e D e campiona il valore di D quando C assume il valore 1 e lo mantiene nell'uscita Q quando C assume il valore zero.

Grazie a questo componente è possibile realizzare un **Flip-Flop D Master Slave** illustrato nella figura 1. Nel breve intervallo temporale in cui il clock c_k assume il valore logico alto,

- il *DLatch Master* a sinistra campiona il segnale di stato futuro d;
- il *DLatch Slave* continua a produrre sull'uscita z il valore dello stato campionato l'intervallo di clock precedente $(d)^{n-1}$ in quanto l'ingresso C per lui è "Low" e presenta dunque in uscita l'informazione memorizzata, cioè non è *trasparente* all'ingresso;

Nell'intervallo in cui il clock c_k assume il valore logico basso invece,

- il *DLatch Master* mantiene il valore della variabile di stato campionata all'inizio dell'intervallo di clock;

- il *DLatch Slave* campiona questo valore presente in uscita dal master e lo presenta (dopo un opportuno ritardo) come valore di stato presente per il corrente intervallo di clock alla propria uscita.

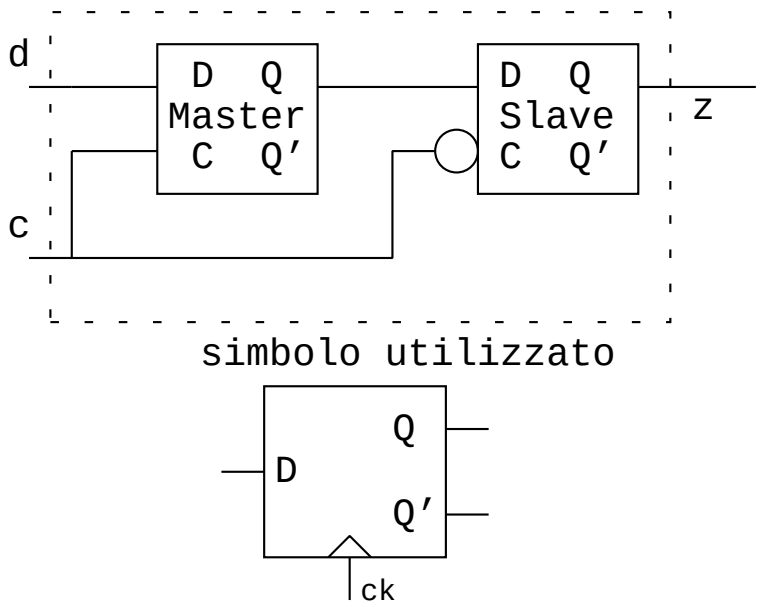


Figura 1: D Latch componenti il FF D Master Slave

Si osserva che nell'ipotesi in cui il segnale di clock rimanga al valore logico alto per un intervallo temporale più lungo del ritardo di attraversamento del *master latch*, possiamo considerare il componente FFD Master Slave **sincrono**.

Sarà questa la componente di memoria di base che utilizzeremo come esempio per sviluppare elementi di memoria dal comportamento più elaborato.

Ricordiamo che i circuiti possono essere:

- SINCRONI quando gli elementi di memoria sono cadenzati e tutte le variazioni dei segnali di stato interno avvengono in concomitanza;
- ASINCRONI quando gli elementi di memoria variano il loro stato interno appena risulta loro possibile, in funzione dei ritardi del circuito e degli istanti di variazione dei segnali di ingresso.

1.5 Il Flip-Flop JK

Procediamo con il progetto di una RSS che memorizzi. Tale rete sarà cadenzata dal segnale di clock c_k, avrà due ingressi sincroni J e K ed un'uscita Q che rappresenta anche lo stato interno presente.

Questo significa che vi è un solo segnale binario di stato e che la MFN non è funzione degli ingressi, ma del solo stato, ed inoltre risulta essere la funzione identità.

Il comportamento di questo componente dovrà essere il seguente:

- per JK=10 (configurazione di **set**), nello *stesso* ciclo di clock il valore di stato (e di uscita) dovrà essere posto ad 1;
- per JK=01 (configurazione di **reset**), nello stesso ciclo di clock il valore di stato dovrà essere posto a 0;
- per JK=00 (**memoria**) lo stato nel corrente intervallo di clock dovrà assumere lo stesso valore dell'intervallo precedente;
- per JK=11 (**commutazione**) lo stato nell'intervallo corrente dovrà essere il complemento di quello dell'intervallo precedente.

Progetto con mappa e schema logico ottenuto dalla descrizione a parole $\rangle$ L'utilizzo di una componente sincrona (FFD) permette di realizzare la rete in forma sincrona tralasciando i problemi connessi alle RSA

L'uscita corrisponde al valore del segnale di stato corrente e l'eccitazione di ingresso al Flip Flop D si ottiene direttamente dalla mappa di Karnaugh con una copertura ottima SP, osservabile in figura 2:

$$D = J \cdot \overline{Q} + \overline{K} \cdot Q$$

convertendo a NAND si ottiene l'espressione utilizzata per ricavare lo schema logico:

$$D = (J \uparrow \overline{Q}) \uparrow (\overline{K} \uparrow Q)$$

Dal momento che l'ingresso D attuale corrisponde allo stato futuro Q dalla prima delle due espressioni è possibile ottenere l'**equazione caratteristica** del FFJK:

$$Q^{n+1} = (J \cdot \overline{Q} + \overline{K} \cdot Q)^n$$

In realtà anche il segnale di clock può essere considerato un ingresso al FF-JK ma questo non influenza la mappa di transizione, codificante la natura del minimo numero di ingressi JK **specificati** da applicare per ottenere una certa transizione dello stato.

Noto che sia infatti lo stato corrente e lo stato futuro in cui ci si deve portare, si possono sfruttare delle condizioni di indifferenza per le eccitazioni: prendiamo ad esempio nella figura 3 la transizione dallo stato $Q^n = 1$ allo stato $Q^{n+1} = 0$ (i valori 1 e 0 si riferiscono in tutta la trattazione seguente rispettivamente al livello logico alto e basso del segnale).

In questa transizione osserviamo che considerando la riga della tabella di verità che contiene $\overline{Q^n}$ come stato futuro (da considerare in quanto la transizione in esame presenta una variazione di stato) e quella contenente come stato futuro il valore 0 presentano come eccitazioni JK rispettivamente 11 e 01, dunque J andrà posto a don't care e K ad 1.

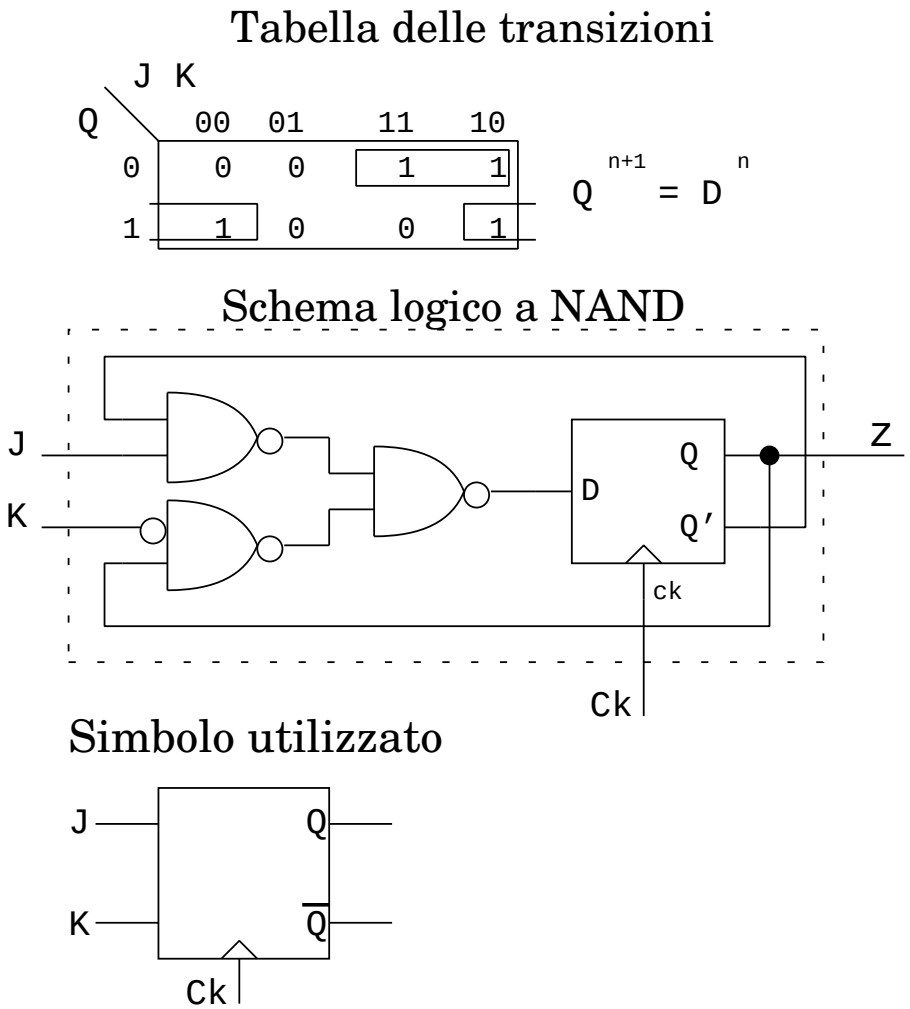


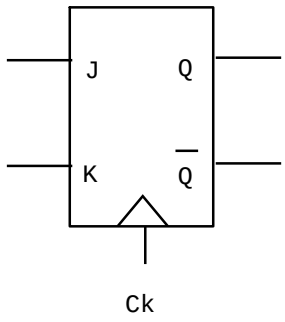
Figura 2: Progetto del Flip Flop JK

2 Automi a stati finiti e RSS

Le reti sequenziali sono dette *sincrone* in quanto cadenzate dal segnale di clock c_k che all'inizio di ogni intervallo (di clock) presenta generalmente un'onda quadra di durata breve e trascurabile rispetto a quella dell'intervallo stesso.

L'ascesa istantanea del livello di c_k è contemporanea all'inizio del periodo T_{c_k} ed in seguito a questa si osservano sequenzialmente gli eventi seguenti:

- alla salita di c_k i segnali di stato futuro vengono campionati dal latch master;
- alla discesa di c_k il latch slave campiona i segnali di stato futuro che sono presentati in ingresso a mfn e a sfn per tutto T_{c_k};



J	K	Ck	Q(n+1)
X	X	0	Q(n)
1	0	1	1
0	1	1	0
0	0	1	Q(n)
1	1	1	$\overline{Q(n)}$

transizione	J	K
0 -> 0	0	d
0 -> 1	1	d
1 -> 0	d	1
1 -> 1	d	0

Figura 3: Particolare delle **transizioni JK**

- i segnali di stato sono dunque ora quelli validi per l'intervallo corrente e presumibilmente lo saranno anche dopo un certo periodo di tempo i segnali di ingresso;
- dopo due differenti transitori le due RLC mfn e sfn calcoleranno il valore di uscita e di stato futuro validi per quel particolare T_{c_k}.

Il valore di stato futuro calcolato rimarrà stabilmente allo stesso valore dimodoché possa essere campionato alla prossima salita di c_k da ogni latch master delle componenti di memorizzazione.

L'estensione temporale dell'intervallo di clock è dimensionata in modo da essere più lunga del massimo della somma degli intervalli di ritardo di caso peggiore che si possono presentare sequenzialmente nelle funzioni RLC **mfn** e **sfn**, nell'ipotesi che i singoli latch dei FFJK abbiano ritardi trascurabili rispetto a questi.

Le RSS sono **sequenziali** in quanto hanno memoria della storia passata e la esprimono attraverso lo *stato interno.* Le **macchine sequenziali** o **automi a stati finiti** possono essere modellizzate da una quintupla:

M={I, U, S, F, G}

- I=$\{i_1,\ldots,i_n\}$ alfabeto di ingresso
- U=$\{u_1,\ldots,u_m\}$ alfabeto di uscita
- S=$\{s_1,\ldots,s_k\}$ insieme degli stati
- funzione di uscita
  F:SxI $\longrightarrow$ U
  $u^n = F(s^n, i^n)$

- funzione di stato
  G:SxI $\longrightarrow$ S
  $s^{n+1} = G(s^n, i^n)$

2.1 Diagramma degli stati

Un automa a stati finiti è rappresentabile da un grafo chiamato **diagramma degli stati**; in questo diagramma ogni stato e' rappresentato da un nodo ed ogni transizione da uno stato all'altro e' rappresentata da un ramo orientato.

Il diagramma degli stati è di **Mealy** quando l'uscita è associata ad una transizione da uno stato all'altro, nel qual caso avrà riportato il valore di questa sull'arco assieme alla configurazione di ingresso che identifica l'evento di transizione.

Il diagramma degli stati è di **Moore** quando l'uscita è associata al permanere in uno stato da parte dell'automa; l'uscita viene in tal caso indicata all'interno del nodo associato allo stato.

Gli automi a stati finiti possono avere

- ampiezza di memoria limitata: la risposta ad ogni nuovo ingresso rimarrà sempre influenzata dallo stato in cui si trovava la macchina all'inizio della sequenza;
- ampiezza di memoria non limitata: la scelta sbagliata dello stato iniziale produce un risultato errato solo per un numero limitato di simboli di ingresso.

2.2 Tabella di flusso

La tabella di flusso rappresenta le medesime informazioni contenute nel grafo degli stati. In questa sono presenti tante righe quanti sono gli stati e tante colonne quante le possibili configurazioni degli ingressi.

Le righe sono identificate da lettere o numeri codificanti lo stato e le colonne sono identificate da numeri codificanti tutte le possibili configurazioni di ingresso.

Nelle tabelle di Mealy agli incroci sono presenti le indicazioni dello stato futuro e dell'uscita prodotta da quella transizione, mentre nelle tabelle di Moore agli incroci è indicato solamente lo stato futuro. La funzione di uscita è indicata in un'ultima colonna sulla destra, in quanto solo lo stato la caratterizza.

2.3 Sommatore seriale

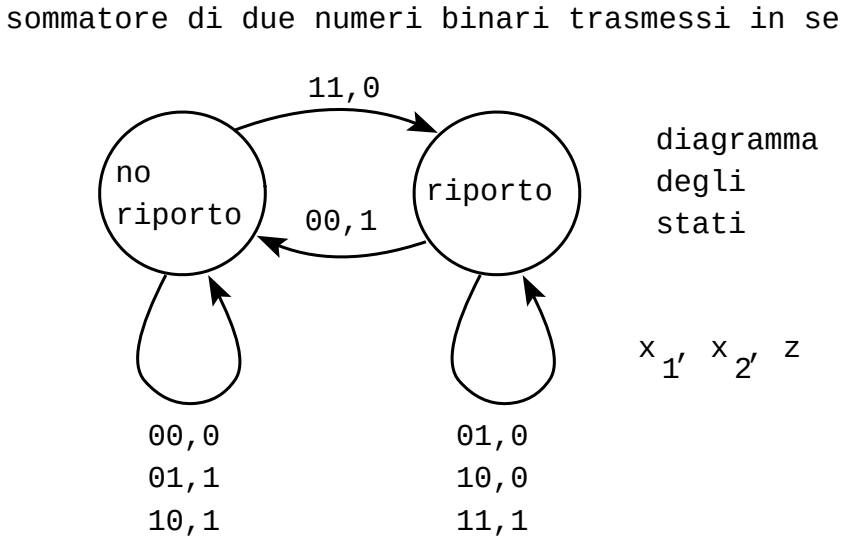
Un esempio di automa a stati finiti è quello di un sommatore seriale di figura 4, che partendo da uno stato iniziale di presenza o di assenza di riporto somma serialmente i bit di due numeri che si presentano agli ingressi producendo in uscita il valore del bit di peso corrente del risultato.

Un sommatore seriale che si trova inizialmente nello stato di assenza di riporto passa all'altro quando riceve entrambi i bit di ingresso affermati sulle linee seriali di somma restituendo zero in uscita.

Dallo stato di assenza di riporto le configurazioni di ingresso con un numero di uni pari a zero o a uno riportano nello stato stesso producendo in uscita rispettivamente zero ed uno.

Nello stato codificante la presenza di riporto solo la configurazione di ingresso priva di valori logici alti riporta nello stato codificante l'assenza di riporto producendo uno in uscita, mentre le altre configurazioni fanno permanere l'automa nello stato corrente restituendo zero tranne che nel caso in cui due i due bit di somma di uguale peso valgano entrambi uno, in quanto la somma di questi con il riporto dei bit di peso precedente restituisce il valore tre.

Questo valore si suddivide in un bit affermato da produrre in uscita al livello di peso corrente (un terzo del risultato) ed in un "due" da riportare come un bit affermato al livello di peso successivo.



stato presente \ ingresso	00	01	11	10
NO	NO, 0	NO, 1	SI, 0	NO, 1
SI	NO, 1	SI, 0	SI, 1	SI, 0

stato futuro, uscita

Figura 4: Particolare: **grafo e tabella dell'automa sommatore**

2.4 Esercizio: analizzatore di comportamento

Una rete sequenziale sincrona verifica l'andamento del segnale di ingresso X in funzione del segnale di controllo C attivando opportunamente l'uscita Z.

Se C=0 la rete ha il compito di verificare se negli ultimi due intervalli il segnale X ha mantenuto il valore 1; se C=1 la rete deve verificare invece se X commuta ad ogni intervallo.

In entrambi i casi la rete manterrà Z=1 finché la condizione è soddisfatta, porterà l'uscita a 0 allorché la situazione non rimanga piu' verificata, e comunque Z dovrà rimanere a livello logico basso nel primo intervallo in cui C e' cambiato.

Si riporti il diagramma degli stati della rete e si effettui il progetto del circuito con FF-JK e porte NAND.

Svolgimento:

Consapevoli che il testo dell'esercizio richiede esplicitamente che il segnale di uscita notifichi le condizioni da segnalare fin dal primo ciclo di clock in cui queste risultano verificate, scegliamo necessariamente il paradigma dell'automa che non associ l'uscita ad uno stato (automa di Moore con SFN dipendente solamente da $(S)^n$) ma esplicitamente alle transizioni.

Con gli automi di Moore infatti un "evento" rilevato è associato alla transizione per un arco ma la transizione stessa non è in relazione alcuna con l'uscita. Le uscite in tal caso vengono indicate all'interno dello stato (nodo del grafo) ed in seguito ad un evento la sola conseguenza della transizione è il raggiungimento di uno stato futuro (che potrebbe anche coincidere con quello presente).

Se realizzassimo la rete con un automa di Moore e per due cicli di clock il segnale di ingresso X assume il valore H al permanere di C al valore L, questo avrebbe indotto l'automa ad effettuare una transizione di stato e solo al terzo ciclo avremmo potuto osservare il valore di uscita Z alto H in quanto giunti nel nuovo stato, quando oramai però la condizione sarebbe potuta essere non più verificata.

Nell'individuazione del grafo degli stati è necessario porre particolare attenzione a:

- non inserire *stati assorbenti* dai quali l'automa non presenti archi uscenti per almeno una configurazione degli ingressi. Uno stato terminale potrebbe tuttavia essere accettato se richiesto ad esempio dalle specifiche.
- Fare in modo che l'automa risulti connesso e che dunque non esistano sottoinsiemi di stati per i quali non esista almeno una transizione che vada da un elemento di un sottoinsieme ad uno dell'altro.
- Cercare di inserire transizioni per ogni possibile configurazione degli ingressi (non si svolgono ottimizzazioni per stati equivalenti o compatibili e la rete è sincrona dunque anche più di un segnale alla volta può variare): in genere questo porta a scoprire nuovi stati che completano l'automa, o al massimo genera stati ridondanti, ma la corretta funzionalità è preservata.
- Non mescolare la notazione per l'uscita associata agli automi di Moore (Z sul nodo) a quella associata a Mealy (Z sull'arco).
- Identificare **tutte** le condizioni di "reset" che riportano allo stato iniziale.

Per l'individuazione del grafo relativo all'automa è consigliabile iniziare da uno **stato stabile** per il quale gli ingressi (segnale di controllo e segnale sotto analisi) abbiano mantenuto valori costanti e non significativi, dunque ignorati dalla rete che si limita a permanere in quello stato in attesa di condizioni opportune (da cui il nome stato stabile).

Lo stato iniziale identifica una condizione di attesa alla quale si perviene quando la rete viene resettata o quando il compito svolto è stato portato correttamente a compimento ma si rimane in attesa di compierlo nuovamente in futuro se le condizioni previste dalla *descrizione a parole* della rete ritorneranno ad essere soddisfatte.

Bisogna ricordarsi dunque di far ritornare la rete allo stato di partenza quando ogni preciso incarico richiesto è stato portato a compimento o anche quando ciò non è stato possibile.

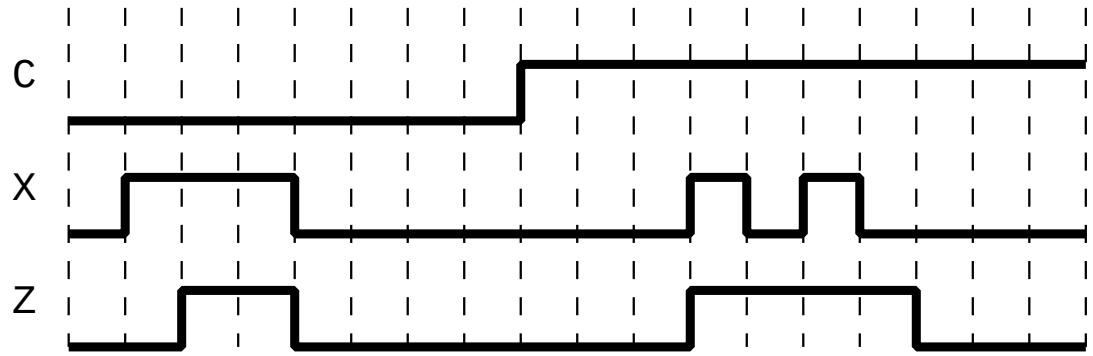
Essendo la rete sequenziale sincrona (RSS) cadenzata da c_k è **ammissibile** che anche più di un segnale di ingresso differisca dalla configurazione utilizzata nel periodo precedente. Nelle RSA (asincrone) per motivi fisici solamente un segnale alla volta poteva "cambiare" di livello (in quanto gli ingressi vengono reputati validi sempre e non solo dopo la marcatura dello stato ai fini del prossimo campionamento).

Per le RSA questo induce problemi con la codifica dei segnali di stato per evitare che **alee essenziali e funzionali** ne compromettano il funzionamento. La **codifica** degli stati e dunque dei segnali di stato nel caso delle RSS non presenta problemi di sorta è può essere completamente **arbitraria**.

2.4.1 Analisi temporale

Può essere sempre utile tracciare alcuni **segnali binari di esempio** come quelli della figura seguente: da questi è possibile osservare un sottoinsieme dei comportamenti desiderati dalla rete. Si parte da una condizione in cui C=X=Z=0 per ogni intervallo precedente non rappresentato. Si osservano in sequenza:

- per C stabilmente "Low", X affermato ad "High" per tre intervalli, una Z affermata nel secondo intervallo e nel terzo;
- all'intervallo successivo X ricade a "Low" e dunque anche Z
- per un certo numero di intervalli non accade nulla di significativo fino a che C si afferma "High" ed il segnale X permane "Low": cambia qui la modalità di analisi;
- C rimane "High" e appena X sale Z risulta essere affermata (in quanto negli ultimi due intervalli è risultato vero che l'ingresso è commutato ad ogni intervallo);
- quando X nuovamente scende, quindi sale e poi scende negli intervalli successivi continua ad essere verificata la stessa condizione e Z permane nello stato affermato;
- solamente quando X rimane a "Low" per il secondo intervallo consecutivo l'uscita si riporta a L e lì permane (assenza di commutazione ad ogni intervallo).

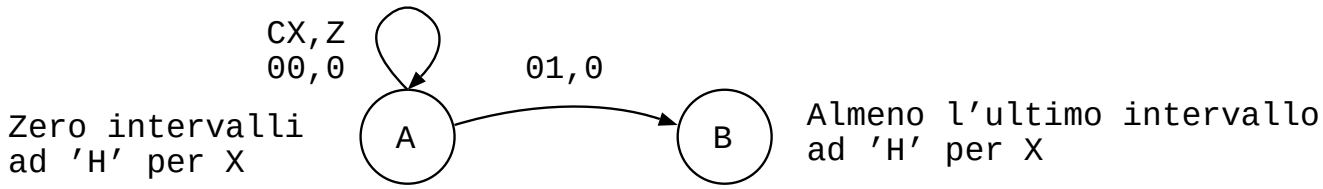


2.4.2 Costruzione dell'automa

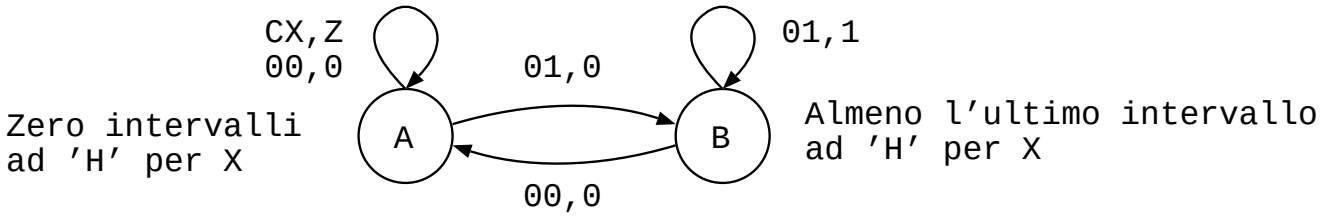
Il passo successivo sarà quello di identificazione dell'automa. Lo stato di partenza/riposo etichettato con "A" possiede un *autoanello* che fa rimanere sempre in quella condizione di attesa per controllo L e ingresso L (con relativa uscita Z "Low").

È bene riportare a fianco di ogni stato significativo una breve descrizione a parole che esprima storia passata da questo codificata, ovverosia il significato associato a quel nodo del grafo.

Nel caso dello stato "A" la descrizione potrebbe essere: "Zero intervalli rilevati al valore logico "High" per il segnale di ingresso X con controllo C Low". Da questo si ha la possibilità di identificare intuitivamente una nuova transizione verso un nuovo stato da aggiungere all'automa. Nel caso infatti in cui permanendo la situazione C="Low", il segnale X passi da L ad H per un intervallo, bisogna identificare un nuovo stato "B" codificante l'informazione "Almeno l'ultimo intervallo presenta X=H con C=L" e inserire nel grafo un arco che faccia transitare dallo stato A a tale stato per la configurazione degli ingressi (CX=01) e recante uscita Z a valore basso.

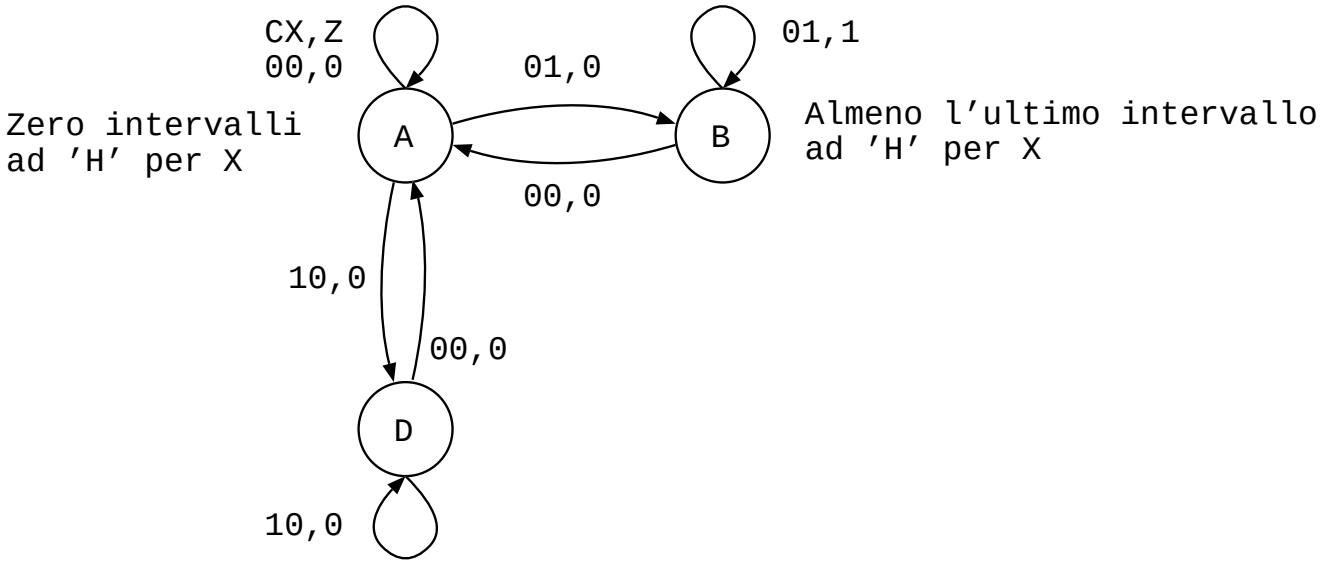


Nello stato B ogni configurazione di ingresso con C=L ed X=H dovrà produrre in uscita Z=1 in quanto per la codifica dello stato B un altro intervallo con ingresso X=H soddisfa alle condizioni identificate dalla descrizione a parole della rete. Per la configurazione di ingresso CX=00 da B dovrà dipartirsi una transizione che porti nello stato A (zero intervalli ad "uno" cumulati per X con uscita Z=L in quanto la rete secondo le specifiche deve portare immediatamente l'uscita a "zero" quando le condizioni non risultano essere più soddisfatte.



Analizzando quali ulteriori archi uscenti debbano essere presenti nello stato A si individua la configurazione CX=10; si identifica dunque uno stato "D" che codifichi la situazione "Segnale di controllo C=H con ultimo valore osservato per X=L". La transizione dovrà riportare uscita Z=0 ("Low") in quanto come da specifica quando il segnale di controllo C cambia di valore l'uscita va mantenuta al valore logico basso.

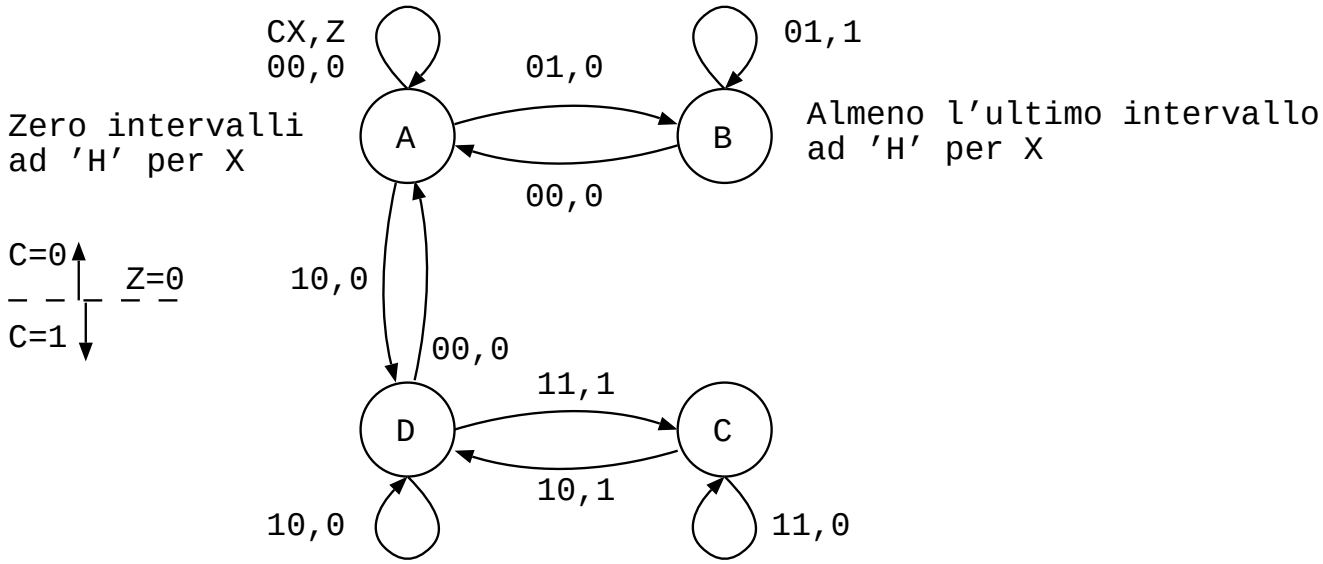
In questo stato D successive configurazioni di ingresso CX=10 vanno associate ad un autoanello in quanto se la situazione attuale indicava come ultimo ingresso uno zero, non si ha commutazione di ingresso se X presenta altri simboli "Low". Sempre da D una configurazione di ingresso CX=00 riporta nello stato A (con Z=L in quanto C ha subito una variazione) in quanto non si sono cumulati intervalli ad "H" per X.



Dallo stato D, al permanere del controllo ad "High" ed al presentarsi di X=H va posta Z=H in quanto si è identificata una variazione negli ultimi due intervalli di clock in X e si perviene in un nuovo stato "C" codificante "Valore di controllo alto ed ultimo valore di X="H". Dallo stato C una configurazione CX=10 identifica una variazione (Z=H) e riporta allo stato D. Una configurazione CX=11 nello stato C invece non identifica variazione (Z=L) e genera un autoanello in quanto l'ultimo ingresso rilevato aveva valore logico alto.

OSS: si è scelto di rappresentare tutti gli stati associati ad un valore logico basso del controllo con nodi presenti nella prima "riga" dell'automa, mentre tutti gli stati codificanti C=H sono riposti nella seconda "riga" del grafo.

OSS: ogni transizione che passa da una "riga" di stati all'altra presenta uscita Z=L.



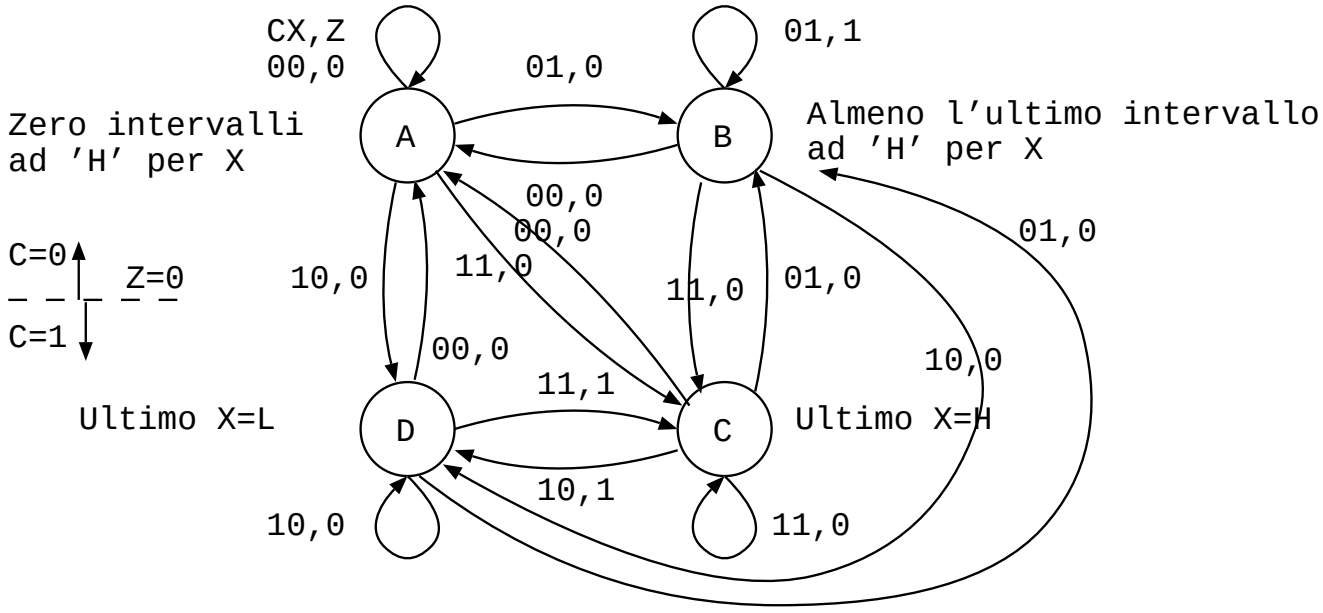
Dallo stato C l'ingresso CX=01 presenta Z=L (cambia il controllo) e porta alla prima riga nello stato B che codifica X=H almeno nell'ultimo intervallo.

Da B, CX=11 restituisce Z=L e porta nella seconda riga nello stato C in quanto l'ultimo X valeva H.

Da C, CX=00 porta in A con Z=L; da A, CX=11 porta in C con Z=L.

Da D, CX=01 porta in B con Z=L; da B, CX=10 porta in D con Z=L.

Ogni configurazione di ingresso per ogni stato è stata mappata in un arco transitante verso stati già stati individuati in precedenza; l'automa risulta connesso e privo di stati assorbenti: è dunque completo in ogni sua parte e si prosegue con il processo di *sintesi* della rete.



Giunti a questo punto si potrebbero adottare algoritmi di minimizzazione del numero di stati ma si tralascia questo e si traduce il *diagramma degli stati* direttamente nella **tabella di flusso**.

La tabella di flusso presentata di seguito è caratterizzata dalle seguenti caratteristiche:

- ogni riga è associata ad uno stato dell'automa
- ogni colonna rappresenta una configurazione delle variabili di ingresso
- ad ogni intersezione si specifica la coppia (stato futuro, uscita) codificata dalla transizione

Essendo arbitraria la codifica degli stati si identificano due variabili binarie y_1 ed y_2 e si riportano i valori associati a queste per ogni stato accanto al simbolo presente nella tabella di flusso.

Il numero di variabili di stato necessarie alla codifica si calcola dal numero di stati mediante la formula
SEGNALI $_{COD} = \lceil \log_2 \text{ STATI} \rceil$
che per questo esercizio indica con due i segnali necessari a codificare quattro stati.

La tabella di flusso o di *transizione* è la seguente:

y2y1	CX	00	01	11	10
00	A	A,0	B,0	C,0	D,0
01	B	A,0	B,1	C,0	D,0
11	C	A,0	B,0	C,0	D,1
10	D	A,0	B,0	C,1	D,0

Figura 5: Tabella di transizione dell'analizzatore

2.4.3 Tabelle di eccitazione ed uscita

Attraverso la *mappatura delle eccitazioni* per il FF-JK scelto si perviene alla **tabella delle funzioni di eccitazione e di uscita** che pone nelle prime colonne tutti i segnali di ingresso a MFN ed SFN (stati correnti ed ingressi alla rete), quindi in colonne centrali i valori dei segnali di stato futuro (indicati di seguito con $(y_i)^{n+1}$ e nelle ultime colonne i valori dei segnali di eccitazione ai Flip Flop $(J_i;K_i)$ ed il valore della funzione di uscita.

Per la creazione della tabella vengono inizialmente inserite tutte le possibili configurazioni per l'ingresso, quindi cercando nella tabella di flusso lo stato futuro per ogni configurazione degli ingressi si inseriscono i valori binari delle variabili di stato futuro e dell'uscita. Infine si considera lo schema di *mapping evoluzione eccitazione*, si confronta per ogni segnale di stato l'evoluzione corrispondente ad una data configurazione di ingresso e si è in grado di specificare i valori binari delle eccitazioni agli ingressi JK dei FF.

Seguono nella figura 6 la tabella di mappatura e la *tabella delle funzioni di eccitazione e di uscita.*

J	K	Qn+1
0	0	Qn
0	1	0
1	0	1
1	1	$\overline{Qn}$

Qn → Qn+1	J	K
0 → 0	0	d
0 → 1	1	d
1 → 1	d	0
1 → 0	d	1

C	X	y2	y1	$(y2)^{n+1}$	$(y1)^{n+1}$	J2	K2	J1	K1	Z
0	0	0	0	0	0	0	d	0	d	0
0	0	0	1	0	0	0	d	d	1	0
0	0	1	0	0	0	d	1	0	d	0
0	0	1	1	0	0	d	1	d	1	0
0	1	0	0	0	1	0	d	1	d	0
0	1	0	1	0	1	0	d	d	0	1
0	1	1	0	0	1	d	1	1	d	0
0	1	1	1	0	1	d	1	d	0	0
1	0	0	0	1	0	1	d	0	d	0
1	0	0	1	1	0	1	d	d	1	0
1	0	1	0	1	0	d	0	0	d	0
1	0	1	1	1	0	d	0	d	1	1
1	1	0	0	1	1	1	d	1	d	0
1	1	0	1	1	1	1	d	d	0	0
1	1	1	0	1	1	d	0	1	d	1
1	1	1	1	1	1	d	0	d	0	0

Figura 6: Tabella delle eccitazioni dell'analizzatore

2.4.4 Mappe ed espressioni ottime

Per ognuna delle funzioni di eccitazione e per la funzione di uscita si generano le mappe di Karnaugh trasponendo i valori per ognuna delle configurazioni degli ingressi come è possibile osservare in figura 7.

Dalla ricopertura SP ottima si ottengono le espressioni

- $J_2 = C$
- $K_2 = \overline{C}$

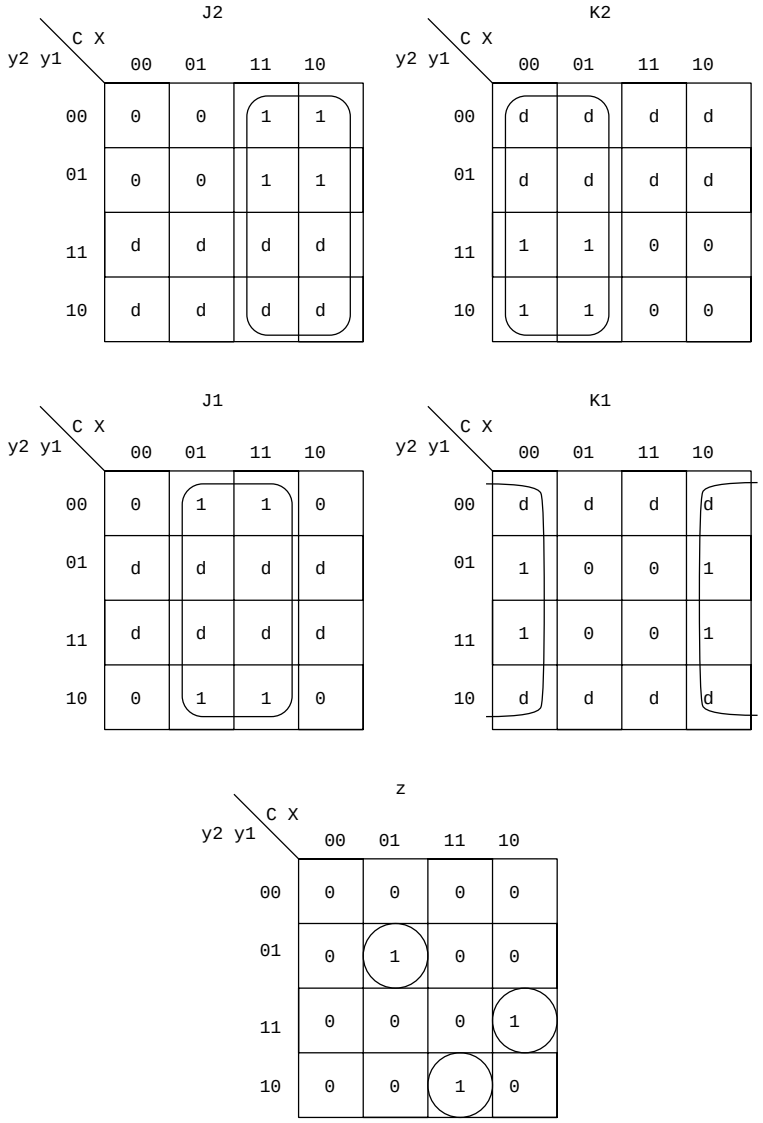


Figura 7: Mappe di Karnaugh dell'analizzatore

- $J_1 = X$
- $K_1 = \overline{X}$
- $z = \overline{C} \cdot X \cdot \overline{y_2} \cdot y_1 + C \cdot \overline{X} \cdot y_2 \cdot y_1 + C \cdot X \cdot y_2 \cdot \overline{y_1}$

Per la realizzazione a NAND è necessario convertire solamente l'espressione dell'uscita z:
$z = (\overline{C} \uparrow X \uparrow \overline{y_2} \uparrow y_1) \uparrow (C \uparrow \overline{X} \uparrow y_2 \uparrow y_1) \uparrow (C \uparrow X \uparrow y_2 \uparrow \overline{y_1})$

2.5 Esercizio: riconoscitore di sequenza

Progettare una RSS con il compito di verificare l'andamento di due segnali di ingresso x_1 ed x_2. L'uscita z dovrà assumere il valore logico H (indicato in seguto anche con1) per ognuna delle due seguenti sequenze di configurazioni osservate agli ingressi:

1. $(x_1, x_2) = (0,0) \rightarrow (0,1) \rightarrow (1,1)$
2. $(x_1, x_2) = (0,0) \rightarrow (1,0) \rightarrow (1,1)$

L'uscita z inoltre dovrà permanere inalterata (H) fintanto che permane la configurazione finale $(1,1)$ delle due sequenze. Si progetti il circuito utilizzando Flip Flop JK e porte logiche.

Suggerimento: per l'automa sono sufficienti solamente tre stati.

Osservazione: se si ha una sola sequenza da identificare di lunghezza n è necessario un automa con al massimo n stati; questi possono essere in numero maggiore se le specifiche sono particolari, ma non in numero minore. Nel caso in esame però le sequenze da riconoscere sono due e cadono le ipotesi.

Svolgimento:

L'identificazione della sequenza deve essere contemporanea all'intervallo in cui questa si è completata: sceglieremo dunque il modello di Mealy, con le uscite indicate sugli archi.

Lo stato iniziale di riposo individuato con la lettera "A" codificherà entrambe le condizioni seguenti:

- non è stata riconosciuta precedentemente alcuna sottosequenza delle due sequenze da individuare
- l'ultima delle configurazioni in ingresso non corrispondeva alla configurazione iniziale delle sequenze

Lo stato "A" presenterà dunque autoanelli con indicazione di uscita z a livello logico basso (nessuna sequenza è stata ancora identificata) e per tutte le configurazioni di ingresso all'infuori della configurazione di inizio sequenze $(x_1, x_2) = (0,0)$ che identificherà invece una transizione verso un nuovo stato etichettato con "B" e codificante le situazioni seguenti:

- è stata riconosciuta solamente una configurazione iniziale delle sequenze
- sono state riconosciute più configurazioni **consecutive** uguali corrispondenti a quella iniziale delle sequenze

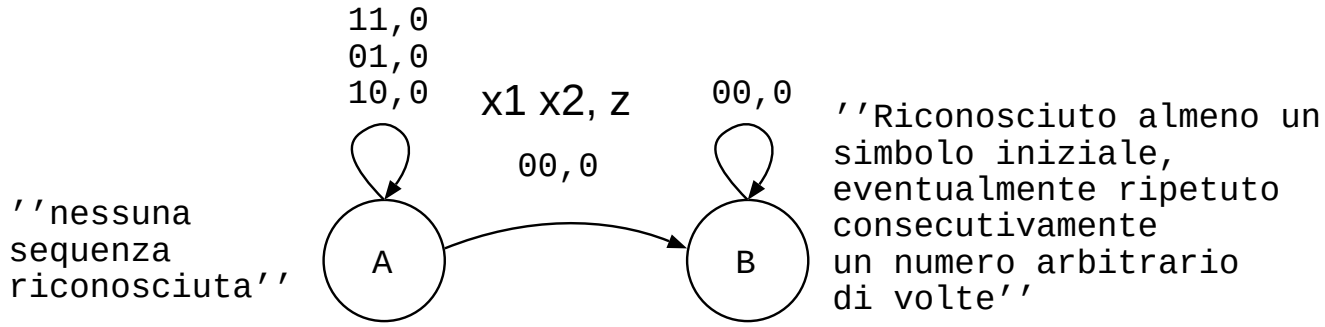
Ogni volta infatti che si presenta la configurazione iniziale bisogna portarsi necessariamente nello stato che identifica la condizione.

Quando la sequenza non prosegue come voluto dalle specifiche, si ritorna allo stato iniziale A se il primo "simbolo vettore" scorretto in ingresso non può essere un eventuale inizio di sequenza, nello stato "B" altrimenti.

Nei riconoscitori infatti bisogna considerare che le sequenze possono iniziare in qualsiasi momento ed un primo fallimento con l'identificazione di una di esse può eventualmente iniziarne una nuova.

2.5.1 Individuazione dell'automa

Riportiamo graficamente l'inizio della costruzione dell'automa con i primi stati e le prime transizioni appena individuate:



Dallo stato B se viene riconosciuto la configurazione 11 si ha che la la sequenza in fase di identificazione è stata terminata non correttamente, ed essendo tale configurazione differente da quella iniziale è necessario adottare una transizione verso lo stato iniziale di riposo A con uscita a livello logico basso in quanto non si ha avuto identificazione.

Dallo stato B invece il riconoscimento di una delle due configurazioni 10 o 01 portano ad identificare un nuovo stato "C", codificante la situazione "trovati consecutivamente i primi due simboli di una delle due sequenze da identificare".

Le due sequenze infatti hanno la lunghezza di tre configurazioni e differiscono solo per il simbolo centrale (vedi figura 8); questo permette di realizzare un automa relativamente semplice.

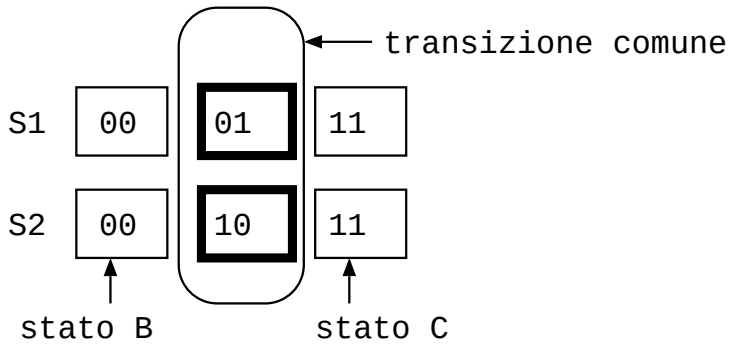


Figura 8: Ottimizzazione dell'analizzatore

Quando l'automa avrà raggiunto lo stato C avrà identificato i primi due simboli o configurazioni delle sequenze in ingresso e da questo, al presentarsi di una o più configurazioni consecutive di $(x_1, x_2) = (1,1)$ dovrà mantenere uscita $z = 1$ la prima volta per avvenuta identificazione e le successive per mantenimento della configurazione finale, come riportato dalla descrizione a parole della rete. Ricordiamo infatti quanto

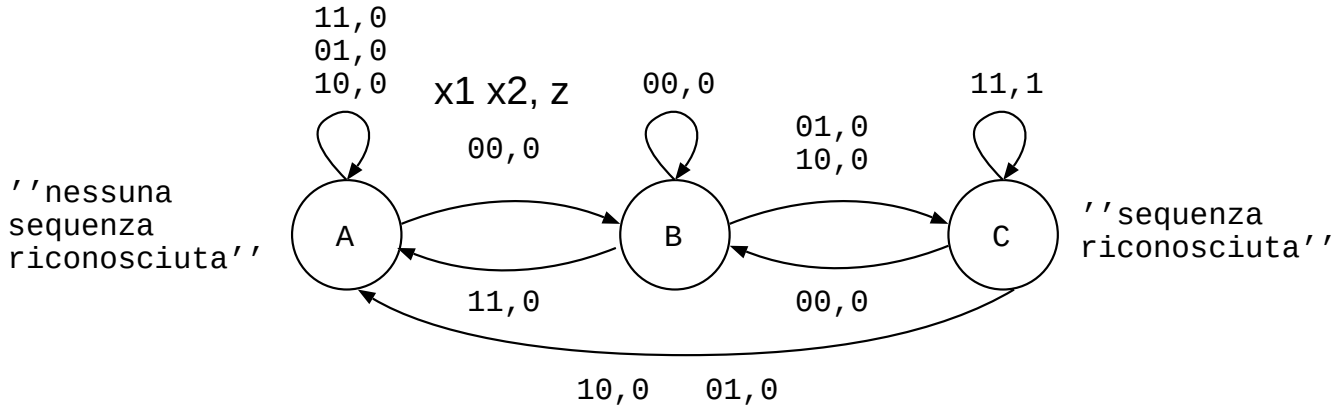
era richiesto: l'uscita sarebbe dovuta rimanere asserita al permanere e non oltre della configurazione finale di entrambe le sequenze.

Rimangono ancora tre configurazioni possibili di ingresso da valutare per lo stato C al fine di aggiungere gli ultimi archi richiesti per completare il grafo.

Tutti e tre questi archi sono relativi a configurazioni differenti da quella terminale delle sequenze e dovranno produrre nel futuro intervallo un'uscita a livello logico basso, ma condurranno a differenti stati.

Le configurazioni 01 e 10 infatti non sono le prime di alcuna sequenza e porteranno allo stato iniziale A mentre la configurazione 00 potrebbe iniziare una sequenza corretta e porterà l'automa nello stato B.

Per ogni stato tutte le transizioni possibili sono state analizzate ed inserite nell'automa, che risulta essere quello della figura seguente:



Dall'automa si giunge direttamente alla tabella di transizione ponendo per ogni stato una riga e per ogni configurazione una colonna; ogni posizione nella tabella identifica lo stato futuro e l'uscita da mantenere nell'intervallo corrente.

Essendo la rete *sincrona* vengono codificati in maniera arbitraria gli stati con l'utilizzo di due variabili di stato presente denominate y_1 ed y_2.

La scelta dei FF-JK porta ad individuare la tabella di eccitazione di figura 9, contenente come ingressi oltre ad x_1 e a x_2 anche le variabili di stato corrente y_1 ed y_2. Due colonne della tabella riporteranno i valori delle variabili di stato futuro $(y_1)^{n+1}$ ed $(y_2)^{n+2}$ ottenute dallo stato futuro indicato nella tabella e dalla codifica degli stati.

La parte finale conterrà la colonna dell'uscita z e le funzioni di eccitazione J_i e K_i dei FF ottenute dalla tabella di mappatura grazie al confronto tra il valore del singolo *bit* di stato presente e quello corrispondente di stato futuro.

2.5.2 Mappe di Karnaugh delle eccitazioni e dell'uscita

Procediamo dunque con la trasposizione di ogni funzione considerata di uscita nella tabella in una mappa di Karnaugh di quattro variabili indipendenti (le stesse quattro della tabella, cioè gli ingressi e i bit di stato

Codifica stati			x1 x2			
y1	y2		00	01	11	10
0	0	A	B,0	A,0	A,0	A,0
0	1	B	B,0	C,0	A,0	C,0
1	0	C	B,0	A,0	C,1	A,0

corrente) ed otteniamo quanto è possibile osservare in figura 10.

Le espressioni SP ottime risultano essere le seguenti:

- $J_1 = \overline{x_1} \cdot x_2 \cdot y_2 + x_1 \cdot \overline{x_2} \cdot y_2$
- $K_1 = \overline{x_2} + \overline{x_1}$
- $J_2 = \overline{x_1} \cdot \overline{x_2}$
- $K_2 = x_2 + x_1$
- $z = x_1 \cdot x_2 \cdot y_1$

2.5.3 Schema logico

La disposizione grafica classica dei blocchi della rete prevede che le uscite dei Flip Flop rappresentanti i bit di stato corrente vengano poste sullo stesso lato degli ingressi in modo da potere utilizzare questi segnali per l'ingresso alla MFN ed alla SFN esattamente come viene fatto per gli ingressi ordinari (vedi figura 11).

Si usa tracciare per ogni "ingresso" due linee di segnale verticali, una per la forma diretta ed una per la forma complementata.

Le due reti combinatorie prelevano i segnali dalle linee di ingresso poste alla sinistra e li producono:

- in uscita alla rete se la funzione è quella del calcolo delle variabili per la codifica dei simboli appartenenti all'alfabeto di uscita
- in ingresso agli elementi di memoria se la funzione è quella di presentare i segnali codificanti lo stato futuro

bit i	bit i+1	J	K
0	0	0	d
0	1	1	d
1	1	d	0
1	0	d	1

x1	x2	y1	y2	(y1)	(y2)	J1	K1	J2	K2	z
0	0	0	0	0	1	0	d	1	d	0
0	0	0	1	0	1	0	d	d	0	0
0	0	1	0	0	1	d	1	1	d	0
0	0	1	1	d	d	d	d	d	d	d
0	1	0	0	0	0	0	d	0	d	0
0	1	0	1	1	0	1	d	d	1	0
0	1	1	0	0	0	d	1	0	d	0
0	1	1	1	d	d	d	d	d	d	d
1	0	0	0	0	0	0	d	0	d	0
1	0	0	1	1	0	1	d	d	1	0
1	0	1	0	0	0	d	1	0	d	0
1	0	1	1	d	d	d	d	d	d	d
1	1	0	0	0	0	0	d	0	d	0
1	1	0	1	0	0	0	d	d	1	0
1	1	1	0	1	0	d	0	0	d	1
1	1	1	1	d	d	d	d	d	d	d

Figura 9: Tabella di eccitazione del riconoscitore

OSS: è sufficiente porre tante RSS a più ingressi ed una sola uscita come nel caso attuale in parallelo per avere una forma generale di RSS a più ingressi e a più uscite.

2.6 Esercizio: moltiplicatore per tre di numeri di lunghezza arbitraria

Si progetti un circuito che moltiplichi per tre un numero N_2 espresso con una rappresentazione binaria e di lunghezza arbitraria acquisito serialmente in ingresso a partire dal bit meno significativo.

In uscita si presenti il risultato $3 \cdot N_2$ serialmente e partendo dal bit meno significativo.

Svolgimento:

Per effettuare questo tipo di moltiplicazione si sfrutta una caratteristica della numerazione binaria: si ricorda infatti che un numero binario moltiplicato per due ha tutte le cifre traslate di una posizione ponderale a sinistra e presenta come cifra meno significativa uno zero, in quanto pari. Infatti:

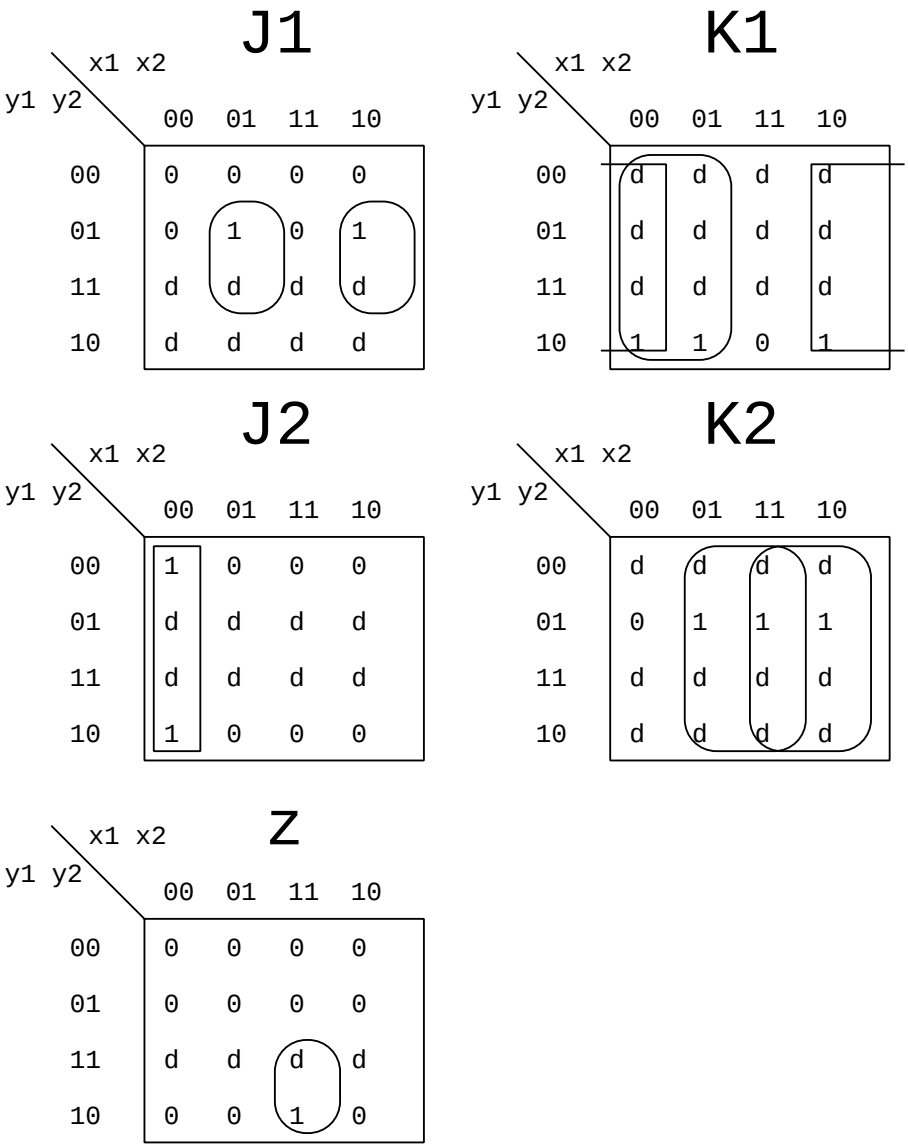


Figura 10: Riconoscitore: mappe e copertura ottima

$$(\ldots b_3 \cdot 2^3 + b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0) \cdot 2 = \ldots b_2 \cdot 2^3 + b_1 \cdot 2^2 + b_0 \cdot 2^1 + 0 \cdot 2^0$$

È possibile sfruttare questo per effettuare una moltiplicazione per tre: il risultato voluto è $N + 2N$ ed è possibile addizionare al numero ricevuto serialmente in ingresso il numero stesso ritardato di un ciclo di c_k, cioè il numero $2N$. Si sommerà cioè al simbolo ricevuto nell'intervallo corrente il simbolo binario ricevuto nell'intervallo precedente, assumendo come bit meno significativo del numero $2N$ il bit zero.

Autore: Amos Tibaldi e-mail address: tibaldi@CE.UniPr.IT

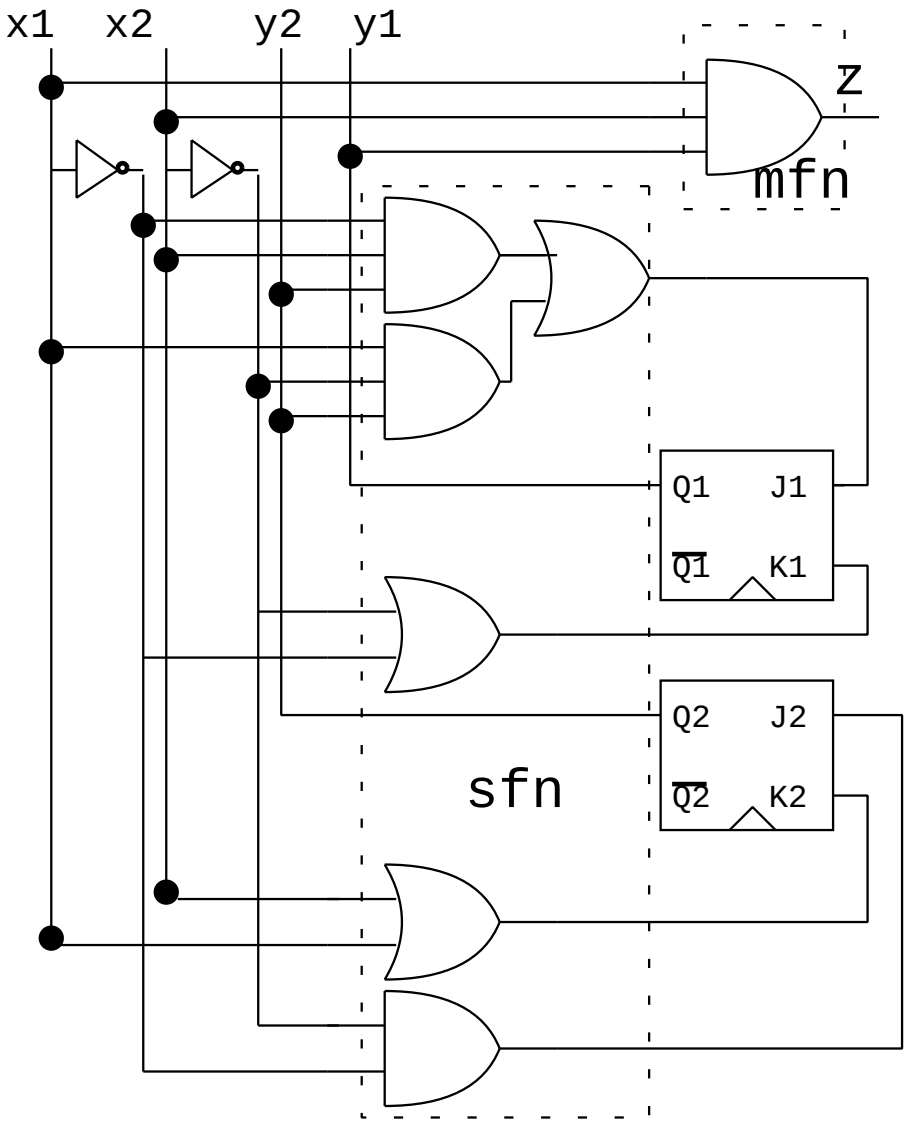


Figura 11: Riconoscitore: schema logico

È necessario inoltre tenere traccia indefinitamente del riporto generato in una somma di qualsiasi livello ponderale ed anche della relativa ripristinata condizione di assenza di riporto. Ad ogni somma effettuata alla ricezione dell'ingresso si addizionano tre bit: l'ingresso corrente, quello dell'intervallo precedente ed il riporto.

Si prosegue con l'individuazione dell'automa: ogni stato deve codificare in sè l'informazione utile per effettuare la somma nell'intervallo corrente, cioè il **numero di bit aggiuntivi** da addizionare dovuti allo stato passato.

Dal momento che la cifra binaria ricevuta nell'intervallo precedente come ingresso deve essere ad-

dizionata nell'intervallo corrente, questa è *in tutto e per tutto* **equivalente** ad un bit di "riporto".

2.6.1 Macchina a stati

Lo stato iniziale A sarà tale da sommare ad uno dei due possibili valori in ingresso il simbolo binario zero nella posizione di peso zero del numero binario $2N$; essendo appena iniziata l'operazione di somma non si ha riporto da considerare e quindi il primo stato individuato produrrà in uscita il bit in ingresso ricevuto.

La RSS presenta solo due archi uscenti da A e da ogni altro nodo, corrispondenti a transizioni effettuate per le sole due configurazioni possibili (alta o bassa) dell'unico ingresso.

Lo stato A codifica in sè la situazione

> Numero di bit asseriti da sommare (o da "riportare"): 0

Questo è necessariamente dovuto al solo caso:

- simbolo zero da sommare in questo intervallo all'ingresso (cioè zero ricevuto nell'intervallo precedente)
- non vi è riporto pendente tramandato da somme effettuate in precedenza

Lo stato A per ingresso $x = 0$, non sommerà a questo alcunché (assenza di riporto e simbolo precedente 0) e dunque produrrà in uscita zero. Per il prossimo intervallo sarà inoltre verificata la proposizione "il simbolo ricevuto nell'intervallo precedente è zero e non è stato tramandato alcun riporto", dunque per tale ingresso lo stato dell'automa presenterà un autoanello (zero bit asseriti da sommare nel prossimo intervallo).

Dallo stato A l'ingresso $x = 1$ deve portare ad uno stato che codifichi la situazione

> Numero di bit asseriti da sommare: 1

che rispecchia una delle seguenti due casistiche:

1. - il simbolo da sommare in questo intervallo è il valore binario 1
   - non vi è riporto pendente
2. - il simbolo ricevuto in precedenza è uno 0
   - vi è riporto pendente

e che verrà etichettato con la lettera B; l'uscita deve tener conto del significato associato allo stato A di partenza (niente riporto, simbolo precedente da addizionare 0). Al valore in ingresso 1 andranno cioè sommati due zeri e sull'arco della transizione verrà riportata l'uscita z al valore logico 1.

In questo nuovo stato B un ingresso $x = 0$ sarà addizionato con un solo bit asserito (B: numero di bit asseriti da sommare = 1). Il risultato da presentare in uscita è dunque 1, e si transiterà verso una situazione in cui si tramanderà il simbolo corrente in ingresso 0 senza necessità di addizionare riporti: condizione espressa già dallo stato iniziale A (nessun bit riportato da sommare).

Da B un “uno” in ingresso vede sommato a sé un solo valore binario a livello logico alto (sia questo dovuto al simbolo ricevuto in precedenza da sommare o al valore del riporto); il risultato (due in decimale, zero con riporto di uno in binario) è espresso con la produzione di un simbolo zero sull’uscita z e con la transizione verso un nuovo stato che codifichi la necessità di sommare “due riporti” alla somma ternaria da effettuare al livello temporale (e ponderale) futuro, dovuti in questo caso allo stato di cose “simbolo da sommare nello stato: 1” (x è asserito) ed alla contemporanea presenza del riporto propriamente detto (scaturito dall’operazione di somma precedente).

Il numero da addizionare in un certo intervallo è 0_{10} se l’automa si trova nello stato A, 1_{10} se l’automa si trova nello stato B e 2_{10} se l’automa si trova in questo ultimo stato che indicheremo con la lettera C.

Quando l’automa si trova nello stato C, la rappresentazione della sua memoria passata è la seguente

Numero di bit asseriti da sommare: 2

situazione necessariamente dovuta a

- simbolo binario asserito ricevuto in ingresso nell’intervallo temporale precedente
- presenza di un riporto pendente a tutti gli effetti da somma precedente.

Da C la configurazione di ingresso $x = 1$ porta a sommare l’ingresso stesso col simbolo 1 di ingresso precedente e col riporto del precedente intervallo; il risultato della somma 3_{10} viene suddiviso in un bit ad 1 da produrre in uscita sull’arco relativo ed in un bit da riportare in futuro. Dal momento che anche l’ingresso corrente è asserito, si transiterà verso lo stato C esprimente la condizione *due riporti.*

La configurazione $x = 0$ invece va a sommarsi con i due riporti dello stato C ed il risultato 2_{10} così ottenuto determina un’uscita non asserita ed un riporto futuro; essendo l’ingresso a valore logico basso non dovrà essere considerato nell’intervallo futuro come ulteriore riporto e dunque lo stato al quale l’arco volgerà sarà B, codificante **un solo** riporto “*esteso*”.

Per ogni stato individuato abbiamo identificato gli archi relativi alle transizioni dovute ai due valori possibili per il segnale in ingresso e questi sono stati tutti diretti verso nodi già identificati; possiamo dunque dedurra che la macchina a stati è completamente definita come si presenta in figura 12.

2.6.2 Tabelle, eccitazioni e mappe

Viene presentata in figura 13 la tabella delle transizioni con la codifica arbitraria degli stati. Vengono scelti per il progetto della rete i FFJK e questo determina il tipo di tabella eccitazione—uscita presentata; infine si può osservare in figura 14 la trasposizione delle funzioni di uscita su mappe per la sintesi SP a due livelli.

Le espressioni per le eccitazioni dei JK risultano essere

$$J_1 = x \cdot y_2$$

$$K_1 = \bar{x}$$

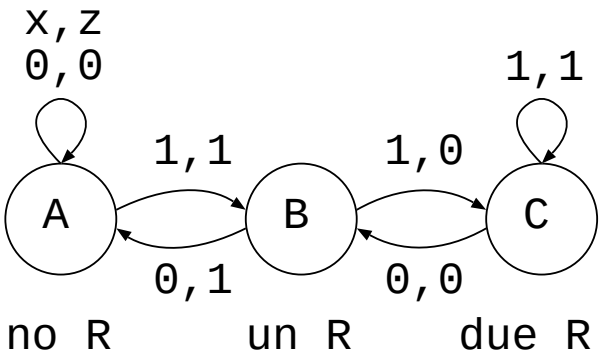


Figura 12: Automa del moltiplicatore $3 \cdot N$

y1 y2	x = 0	x = 1
00 A	A,0	B,1
01 B	A,1	C,0
10 C	B,0	C,1

S	S++	J	K
0	0	0	d
0	1	1	d
1	0	d	1
1	1	d	0

x	y1	y2	y1 n+1	y2 n+1	J1	K1	J2	K2	z
0	0	0	0	0	0	d	0	d	0
0	0	1	0	0	0	d	d	1	1
0	1	0	0	1	d	1	1	d	0
0	1	1	d	d	d	d	d	d	d
1	0	0	0	1	0	d	1	d	1
1	0	1	1	0	1	d	d	1	0
1	1	0	1	0	d	0	0	d	1
1	1	1	d	d	d	d	d	d	d

Figura 13: Tabella di transizione automa $3 \cdot N$

$$J_2 = x \cdot \overline{y_1} + \overline{x} \cdot y_1$$

$$K_2 = 1$$

e la funzione di uscita vale $z = \overline{x} \cdot y_2 + x \cdot \overline{y_2}$.

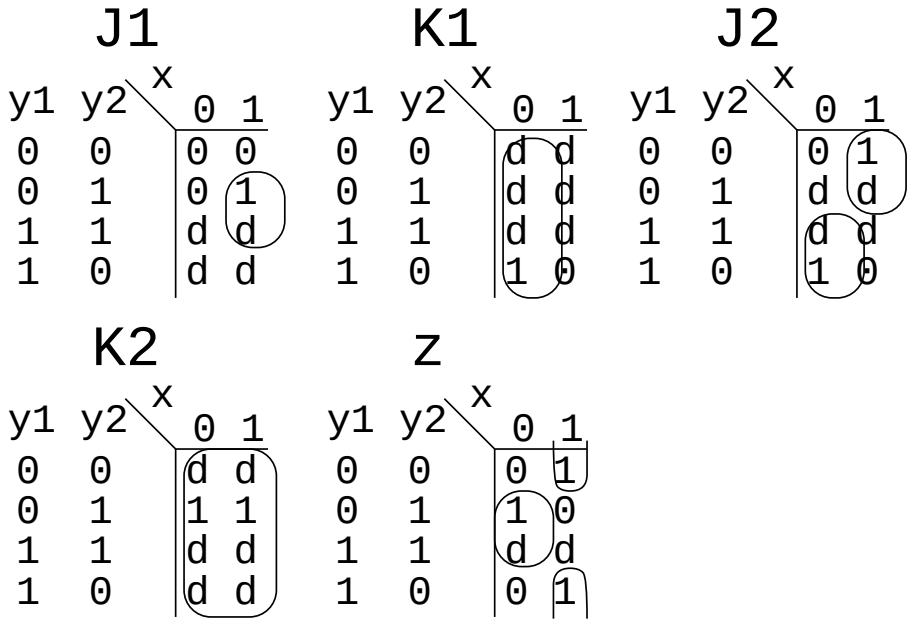


Figura 14: Mappe di Karnaugh automa $3 \cdot N$

3 Esercizi suggeriti

- Progettare una RSS a due ingressi mai attivi contemporaneamente e con due uscite che segnalano se e quale dei due ingressi è rimasto affermato per almeno due intervalli consecutivi. Automa in figura 15.

- Progettare una RSS a due ingressi ed una uscita, affermata se il primo ingresso è affermato e nel periodo precedente in cui non lo era si era verificata almeno una variazione del secondo ingresso. L'uscita torna al valore logico basso quando il primo ingresso non risulta più essere affermato e se la variazione del secondo ingresso coincide con la salita del primo, tale variazione va ignorata. Automa in figura 16.

- Progettare una RSS che date due sequenze di *nibble* [1] presentati in ingresso serialmente e dal bit **meno** significativo, nell'intervallo di **ricezione** del MSB, asserisca l'unica uscita nel solo caso entrambi hanno rispettato la codifica Binary Coded Decimal; in ogni altro caso l'uscita è indifferente. Automa in figura: 17.

- Progettare una RSS che date due sequenze di *nibble* presentati in ingresso serialmente e dal bit **più**

[1] sequenza ordinata di 4 bit (es: 0010) ovvero parola binaria formata da 4 bit; la concatenazione di due nibble può formare due diversi byte in base all'ordine con cui vengono posti in sequenza e se alle posizioni di ogni singolo bit del nibble era associato un determinato valore ponderale, quando questo viene posto nella posizione più significativa "di sinistra" del byte, il valore di ogni bit di questo viene moltiplicato per 2^4

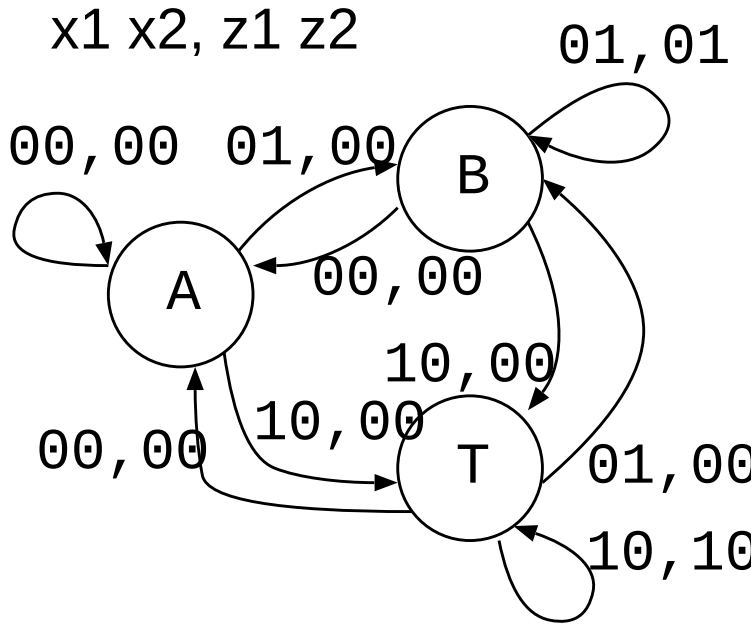


Figura 15: Automa segnalante uno dei due ingressi affermato per 2 intervalli

significativo, nell'intervallo di ricezione del LSB, asserisca l'unica uscita nel solo caso in cui almeno uno di essi è primo e l'altro è diverso dal numero quattro; in ogni altro caso l'uscita è indifferente.

- Progettare una RSS che date due sequenze di *nibble* presentati in ingresso serialmente e dal bit **più** significativo, nell'intervallo di ricezione del LSB, asserisca l'unica uscita nel solo caso in cui la somma sia dispari e solo uno dei due sia maggiore di 8_{10}; in ogni altro caso l'uscita è indifferente.
- Progettare una RSS con un ingresso seriale ed un'uscita mantenuta a zero se non è vero che il nibble costituito dal valore di ingresso corrente e dai suoi precedenti è una palindrome; le prime tre uscite sono indefinite.
- Progettare una RSS che riceve serialmente in ingresso una sequenza di nibble e che produce alla ricezione dell'ultimo bit di ognuno l'uscita asserita nel solo caso in cui erano presenti in questo almeno tre bit asseriti; in ogni altro caso l'uscita va mantenuta al livello logico basso.

Fine esercitazioni integrative [2]

[2] La corrente versione di questa esercitazione non è ancora definitiva; per informazioni o per riportare errori individuati: tibaldi@ce.unipr.it

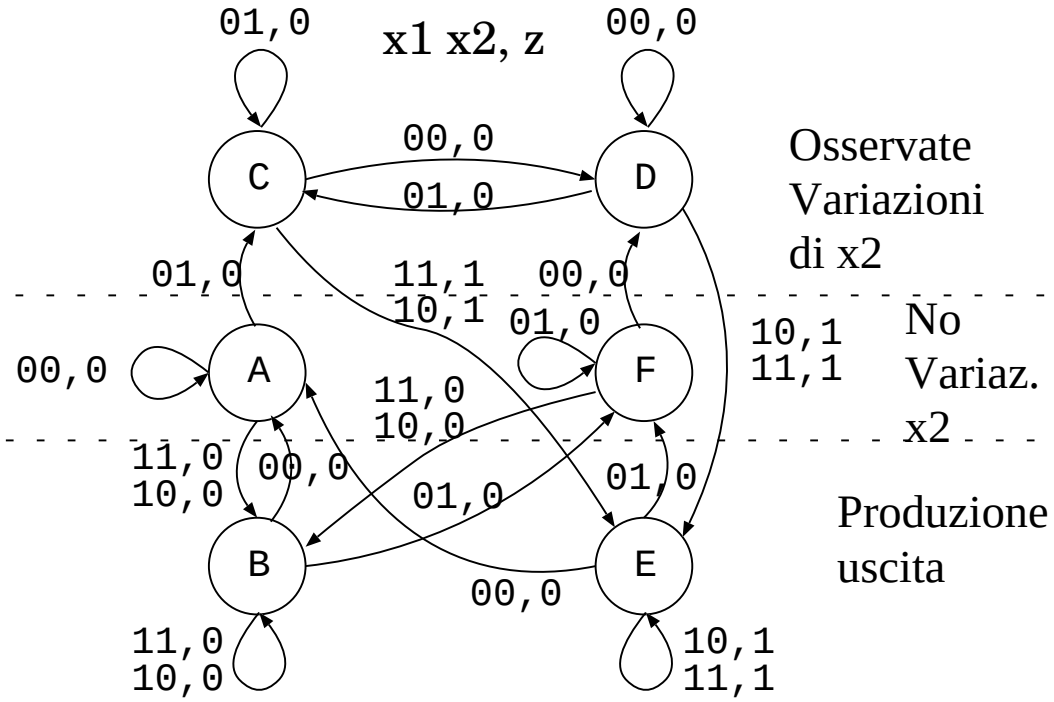


Figura 16: Automa segnalante una variazione di x_2 nel periodo in cui $x_1 = 0$

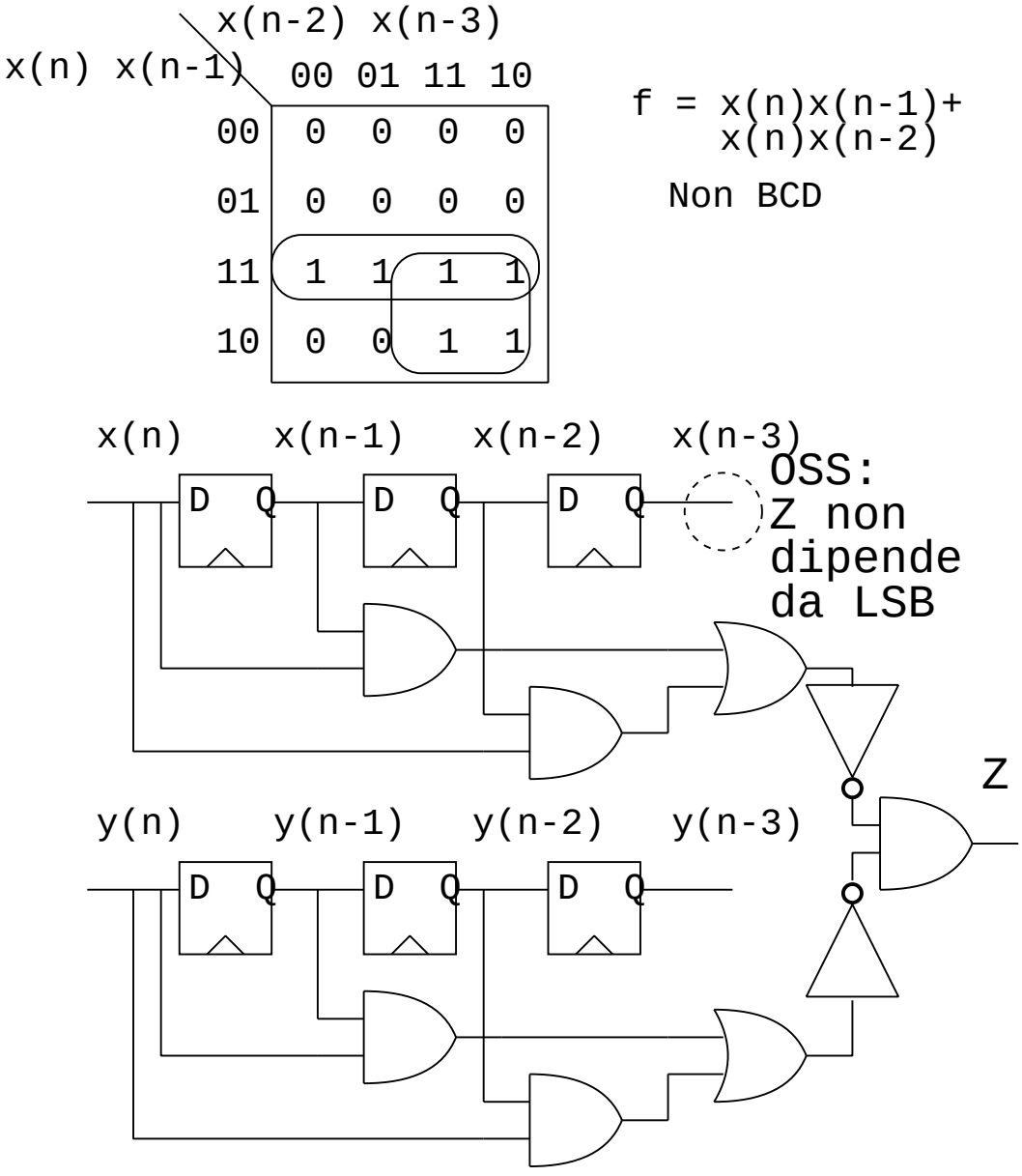


Figura 17: Automa riconoscente due BCD in serie