Capitolo 6: Reti asincrone

6.1 – Elaborazione asincrona

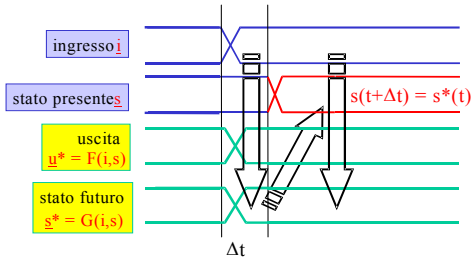
Il modello e le applicazioni

Il modello della rete asincrona prevede che il simbolo d'uscita si possa modificare soltanto quando si verifica una modifica del simbolo d'ingresso.

Caratteristica della macchina asincrona è dunque l'inseguire continuamente il variare dell'ingresso passando da una ad un'altra condizione di stabilità dello stato interno.

Le forme d'onda di figura illustrano sia questo aspetto del comportamento, sia il fatto conseguente che l'uscita non può variare se non varia l'ingresso.

Reti sequenziali asincrone (comportamento)

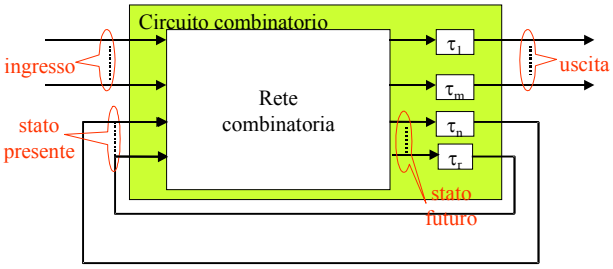


Ogni nuovo simbolo d'ingresso può o lasciare inalterati stato e uscita, o modificare la sola uscita, o generare una transizione di stato, con o senza modifica dell'uscita. Tutto questo può essere ottenuto in modo semplicissimo:

- si realizzano le funzioni F e G in modo da avere per ogni stato presente e per ogni ingresso, i valori appropriati di uscita e di stato futuro;
- si collegano i segnali che codificano lo stato futuro a quelli che codificano lo stato presente.

La **retroazione diretta** dei segnali di stato è consentita dalla intrinseca presenza dei ritardi sulle uscite del circuito combinatorio che realizza la funzione G. Il ritardo permette infatti di distinguere tra uno stato presente all'ingresso della macchina combinatoria G ed uno stato futuro disponibile sulla sua uscita.

Reti sequenziali asincrone (struttura)



In questo modo il circuito può operare correttamente in due passi: dapprima prende atto del valore d'ingresso "a stato costante" e poi, se necessario, aggiorna lo stato "ad ingresso costante". Si noti comunque che tale distinzione serve in realtà solo all'arrivo di un nuovo simbolo di ingresso: appena trascorso il ritardo di propagazione e per tutto il successivo intervallo di tempo in cui l'ingresso rimane costante, il valore dello stato presente coincide sempre con quello dello stato futuro (stabilità).

Una rete logica sequenziale asincrone con **k** retroazioni dispone di $\mathbf{2^k}$ stati interni, che impiega tipicamente per **discriminare** e **ricordare** quali e quante variazioni si sono verificate sui suoi ingressi.

L'applicazione più importante è la **memorizzazione del valore di un bit**: a questo proposito si ricordi che il **flip-flop** posto su ogni anello di retroazione di ogni macchina sincrona è in realtà una rete logica sequenziale asincrona.

L'elaborazione asincrona è inoltre chiamata in causa nella realizzazione e gestione delle **unità funzionali** di ogni macchina digitale complessa.

ESEMPI – La memoria principale di un calcolatore è una complessa rete asincrona. L'unità d'ingresso elabora in modo asincrono le forme d'onda dei segnali che riceve dall'esterno per chiamare in causa la CPU solo quando si verificano eventi significativi. Le comunicazioni tra l'unità di controllo ed il percorso dei dati sono affidate a reti asincrone.

Per la loro velocità nel prendere decisioni, i circuiti con retroazioni dirette sono pericolosamente esposti a **malfunzionamenti**; vediamo come sia possibile eliminarli a priori.

Regole di corretto impiego

Il continuo passaggio da una ad un'altra condizione di stabilità dello stato interno deve essere garantito per **qualsiasi valore dei ritardi** presenti sugli anelli di retroazione.

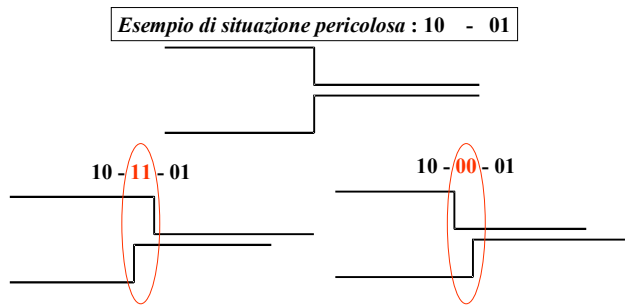
Ciò impone il rispetto di una prima importante regola di corretto impiego:

- **funzionamento in modo fondamentale -** *l'ingresso può essere modificato solo dopo che il circuito ha raggiunto la stabilità.*

Il modo fondamentale è una condizione necessaria, ma non sufficiente, per il corretto funzionamento di un circuito con retroazioni dirette. Ci si deve infatti preoccupare di eliminare a priori l'insorgere di situazioni "non corrette" sugli ingressi della rete.

In figura è ad esempio mostrato il caso di due segnali d'ingresso per cui è stato previsto, ad un certo istante, il cambiamento contemporaneo del valore.

La codifica degli ingressi della funzione G



La cosa è fisicamente impossibile. La variazione di uno dei segnali precederà sicuramente la variazione dell'altro, senza che si possa neppure prevedere in quale ordine, con il risultato che una sequenza di due simboli diventa una sequenza di tre simboli.

Se il fenomeno dura abbastanza, il circuito ha il tempo di prendere decisioni anche sulla configurazione d'ingresso "spuria": da quel momento in poi si comporterà dunque in maniera errata. Ciò ovviamente non capita se non è mai previsto che due o più segnali d'ingresso possano cambiare di valore contemporaneamente, vincolo questo che può essere imposto in fase di codifica binaria dell'alfabeto di ingresso, a patto di conoscere le sequenze di simboli ammissibili.

Da quanto detto discende una seconda regola di corretto impiego:

- **adiacenza delle configurazioni d'ingresso consecutive** - *simboli d'ingresso consecutivi devono essere codificati in modo da differire per il valore di un solo bit.*

Eliminazione a priori delle corse critiche

Anche i segnali in retroazione sono ingressi della funzione di aggiornamento G e chi progetta il circuito deve quindi garantire l'**adiacenza delle configurazioni di stato consecutive**.

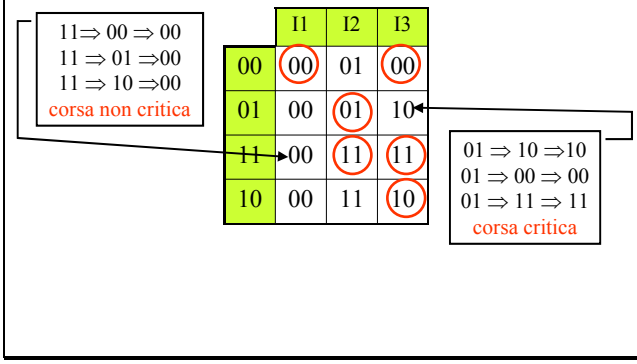
La condizione è piuttosto restrittiva: se **k** sono infatti i bit di codifica dello stato interno, da ogni stato presente variando un solo bit è possibile raggiungere soltanto **k** dei 2^k-1 stati possibili.

Nella tabella delle transizioni indicata in figura esistono due casi in cui è prevista una contemporanea variazione di entrambe le variabili di stato (da 11 a 00 per ingresso I1, da 01 a 10 per ingresso I3).

Nel circuito che dovrà avere questo comportamento è estremamente improbabile che le cose vadano proprio così: a causa dei ritardi interni infatti uno dei due segnali in retroazione cambierà di valore prima dell'altro.

Nella colonna I1 non succede nulla di male: qualsiasi sia l'ordine con cui si verificano i cambiamenti, il circuito raggiunge

Corse e Corse critiche



infatti la prevista stabilità nello stato 00 dopo essere temporaneamente transitato o per lo stato 01 o per lo stato 10. In casi di questo tipo (caratteristica comune è la presenza nella colonna di **una sola stabilità**) si dice che i segnali di stato si trovano in una situazione di **corsa non critica**: alla fine del transitorio, lo stato è sempre quello desiderato.

Una sequenza di **più stati instabili** tra la stabilità di partenza e quella d'arrivo è detta **transizione multipla**. Questa deviazione dal modello teorico della macchina asincrona è considerata lecita, data la grande velocità con cui il circuito percorrerà la sequenza. Naturalmente è necessario che durante la transizione multipla

- l'uscita non presenti andamenti diversi da quelli previsti per la transizione originaria,
- la durata del simbolo d'ingresso sia sufficiente al raggiungimento della nuova stabilità.

Ben diversa è la situazione nella colonna I3: lo stato 10 viene raggiunto solo nel caso improbabile di variazione contemporanea dei segnali in retroazione; molto più probabile è che il circuito si trovi ad operare temporaneamente nello stato spurio 00 o nello stato spurio 11, ciascuno dei quali, a differenza del caso precedente, non conduce a sua volta allo stato finale corretto, ma è stabile e quindi porta la macchina ad assumere un riassunto errato della storia passata. In questi casi si parla di **corsa critica** (caratteristica comune è la presenza nella colonna di **più stabilità**).

Le situazioni di corsa critica possono essere eliminate a priori se si riesce ad individuare una codifica degli stati che soddisfi la seguente regola:

- **adiacenza delle configurazioni di stato consecutive** – *nelle colonne in cui sono previste più condizioni di stabilità, i simboli di stato futuro devono essere codificati in modo da differire per il valore di un solo bit dalle configurazioni dei corrispondenti simboli di stato presente; nelle colonne in cui è prevista una sola stabilità ed in cui tutte le righe contengono l'indicazione dello stato da raggiungere, è lecito invece avere transizioni che richiedono la variazione contemporanea di più variabili di stato.*

Esiste un procedimento a quattro passi che, data una qualsiasi tabella di flusso, consente di arrivare sicuramente ad una tabella delle transizioni priva di corse critiche. Cominciamo a discutere i primi due passi.

1: analisi delle esigenze di adiacenze – Dopo aver idealmente eliminato dalla tabella di flusso le colonne in cui è presente una sola condizione di stabilità, si traccia un **grafo delle adiacenze**, in cui tutti gli stati della macchina sono connessi a tutti i loro stati futuri da un ramo non orientato.

2: individuazione del codice privo di corse critiche – Si costruisce una mappa per il n° minimo di variabili di stato e le si sovrappone il grafo delle adiacenze (così com'è, o anche ruotato e distorto): se ogni ramo del grafo collega due celle adiacenti, la mappa indica la codifica cercata.

Riprendiamo l'esempio precedente e costruiamoci la tabella di flusso sostituendo le quattro configurazioni di stato con i numeri decimali corrispondenti.

Con il primo passo del procedimento si determina il grafo delle adiacenze indicato in alto a destra nella figura a lato.

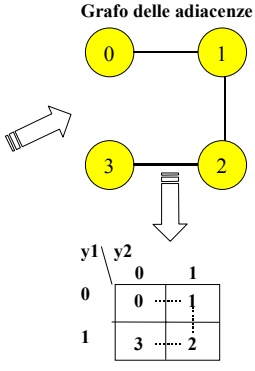
Con il secondo passo (v. mappa a 2 bit sottostante) si individua una codifica non ridondante degli stati, atta a garantire nelle colonne I2,I3 transizioni che richiedano la modifica di un solo bit di stato.

Non è però detto che si arrivi così sempre al risultato desiderato.

Una codifica priva di corse critiche

Tabella di flusso

	I1	I2	I3
0	0	1	0
1	0	1	2
3	0	3	3
2	0	3	2



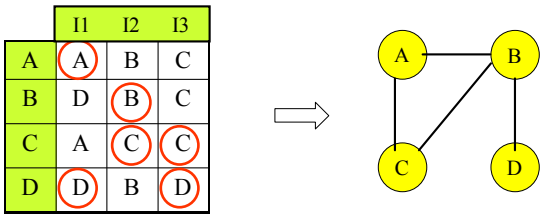
Vediamo subito un caso in cui, apparentemente, non è possibile individuare un codice non ridondante privo di corse critiche.

Le esigenze di adiacenza poste dalla tabella di flusso di figura sono evidenziate nel grafo indicato a lato

Nel caso in esame è inutile tentare di sovrapporre il grafo ad una mappa di due bit.

Ogni cella di questa mappa infatti ha due sole celle adiacenti, mentre la configurazione da associare allo stato B dovrebbe risultare adiacente a quella degli stati A (esigenza posta dall'ingresso I2), C (esigenza posta dall'ingresso I3) e D (esigenza posta dall'ingresso I2).

Analisi di esigenze di adiacenza



In situazioni di questo tipo occorre eseguire il terzo passo del procedimento.

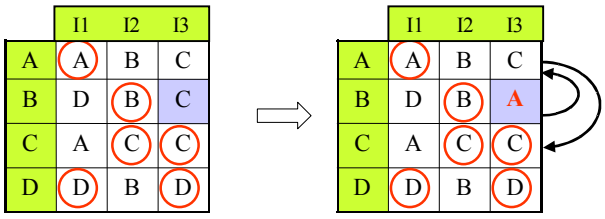
3: modifica della tabella di flusso - Si introducono, se possibile, **transizioni multiple** atte a ridurre le esigenze di adiacenza tra gli stati e si ripetono i passi 1 e 2.

Nel caso precedente l'accorgimento è utile. La transizione B → C, richiesta solo nella colonna I3 in cui è presente anche una transizione A → C, può essere sostituita dalla transizione B → A → C, riducendo così di un'unità le esigenze di adiacenza dello stato B.

In figura è evidenziato come deve di conseguenza essere modificata la tabella.

Ciò fatto è possibile impiegare due sole variabili di stato e garantire l'adiacenza di ogni coppia stato presente-stato futuro sulla tabella delle transizioni: un codice corretto è, ad esempio, A = 00, B = 01, C = 10, D = 11.

Il progetto di una transizione multipla

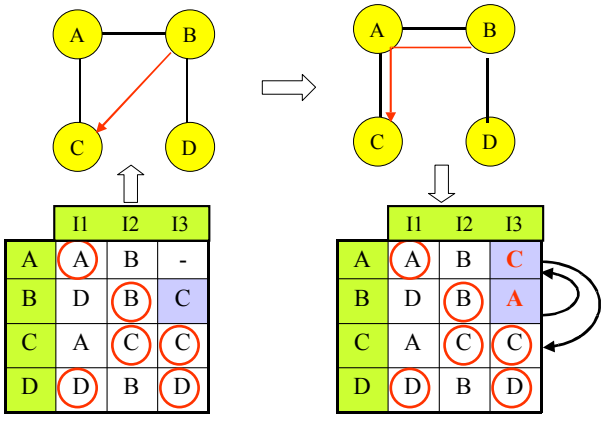


L'accorgimento della transizione multipla è lecito anche nel caso in cui esistano stati con stato futuro indifferente: l'impossibilità che in presenza di quello stato si verifichi un certo ingresso ci consente di imporre uno stato futuro di nostro gradimento al posto della indifferenza.

Riprendiamo in considerazione l'esempio precedente, prevedendo però stato futuro indifferente nell'incrocio A, I3. Per eliminare la corsa critica nella transizione B → C devono essere apportate due modifiche:

- A al posto di C nell'incrocio B, I3;
- C al posto di – nell'incrocio A, I3.

Uso di condizioni d'indifferenza



Esistono però anche tabelle di flusso in cui i due accorgimenti visti non sono applicabili. Per risolvere queste situazioni il procedimento prevede un quarto passo.

4: uso di un codice ridondante - Si incrementa il n° di bit di codifica e si ritorna al passo 3.

Si noti che una nuova variabile di stato raddoppia gli stati a disposizione con il doppio beneficio di aumentare di un'unità il numero di configurazioni adiacenti a ciascuno stato presente e di mettere a disposizione un grande numero di indifferenze su cui impostare transizioni multiple.

Eliminazione a priori delle alee statiche

Un circuito funzionante in modo fondamentale, con segnali d'ingresso e di stato che si modificano uno solo alla volta può ancora presentare malfunzionamenti.

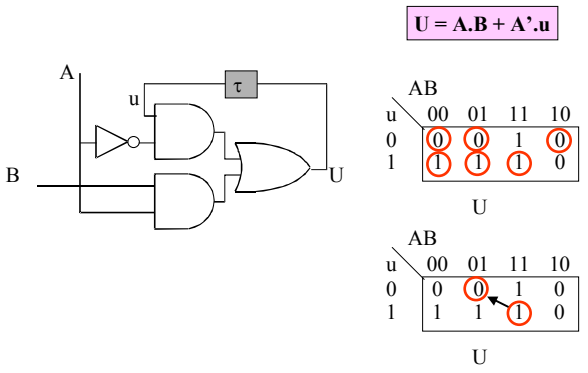
Consideriamo la rete con retroazione diretta indicata in figura. Dallo schema è agevole dedurre una tabella delle transizioni con sei condizioni di stabilità.

Supponiamo ora che il circuito si trovi nella situazione stabile A=B=U=u=1 e che l'ingresso A assuma il valore 0.

Come sappiamo da pag.75, con questi valori d'ingresso la parte combinatoria presenta un'alea statica.

In questo caso però il valore 0 di U può non essere un fenomeno transitorio: se infatti il NOT su A ha un ritardo maggiore di τ, lo "zero" fa a tempo a propagarsi fino ad u ed a stabilizzarsi poi sull'anello di retroazione. In conclusione il circuito raggiunge la situazione di stabilità A=0, B=1, U=u=0 e non quella prevista dal suo progettista: A=0, B=1, U=u=1.

Un circuito asincrono con alea statica



Durante il progetto occorre dunque rispettare un'ulteriore regola:

- **eliminazione a priori delle alee statiche** – *il ritardo deve essere l'unico fenomeno transitorio consentito alla rete combinatoria che calcola lo stato futuro di una macchina asincrona.*

Il rispetto di **T14** (v. pag. 89) è un modo usuale per non avere transitori pericolosi.

6.2 - Memorie binarie

La denominazione di **memoria binaria** viene attribuita a tutte le reti asincrone che sono state adibite a ricordare il valore di un bit; l'importanza di questo comportamento le rende **componenti primitivi** per ogni macchina digitale, come dimostra il fatto che la loro realizzazione è inclusa in ogni famiglia di circuiti integrati.

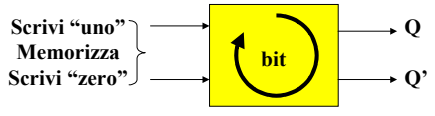
La memoria binaria ha:

- **due segnali d'uscita** per comunicare il bit in forma vera e complementata;
- **due segnali d'ingresso** per sapere **quando** e **cosa scrivere** al suo interno; l'alfabeto d'ingresso contiene 3 ordini: "*metti in memoria un 1*", "*metti in memoria uno 0*", "*mantieni in memoria il valore che c'è*".

Sono stati messi a punto tre differenti tipi di memoria binaria:

1. il **latch SR**,
2. il **latch CD**,
3. il **flip-flop**.

Memorie binarie



- Latch SR
- Latch CD
- Flip-flop edge-triggered

complessità strutturale

Semplicità d'uso

Le differenze dipendono dal **tipo di sequenza** impiegata per individuare il "quando" ed il "cosa" scrivere in memoria, aspetti che influenzano poi la **complessità logica interna** del circuito e la sua **facilità d'impiego**. Nel seguito studieremo i tre casi dal più semplice al più complesso, per potere così acquisire gradualmente familiarità con il progetto di ogni altra rete asincrona.