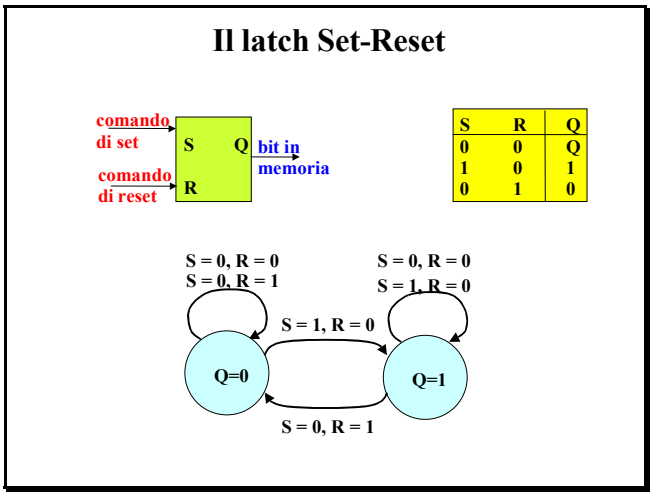
Latch SR

Del latch SR abbiamo già parlato più volte. L'analisi del relè ad autoritenuta, iniziata a pag. 18, è stata poi conclusa a pag. 46; quella di due NOR in retroazione è stata fatta prima a parole (pag. 19) e poi con formule (pag. 72). Vediamo ora il procedimento di sintesi.

1: descrizione del comportamento - Una rete logica deve ricordarsi il valore di un bit **Q** comunicatole tramite due ingressi denominati comando di **Set** e comando di **Reset**:

- attivando il Set si deve avere Q = 1,
- attivando il Reset si deve avere Q = 0,
- tenendoli entrambi a zero Q deve mantenere il valore precedentemente attribuitogli,
- l'attivazione contemporanea di Set e Reset non ha significato e non deve essere fatta.

2: individuazione del grafo degli stati - In figura è mostrato che per ottenere questo comportamento occorrono due stati interni.

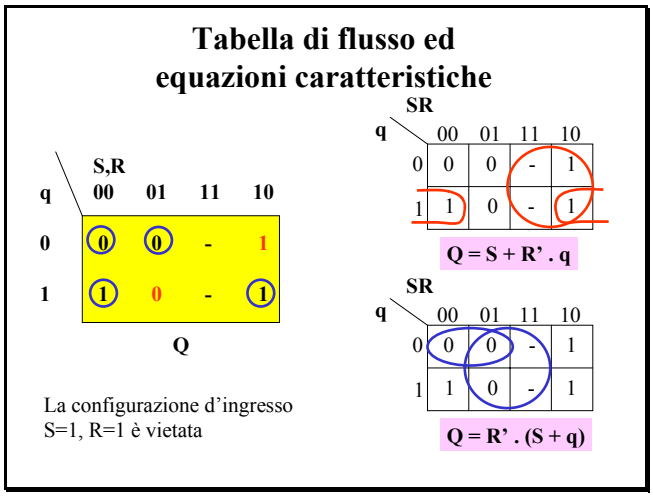


3: individuazione delle tabelle, delle mappe e delle espressioni – Denotiamo con q la variabile di stato presente e con Q quella di stato futuro. La tabella di flusso (quando gli stati interni sono già codificati in binario si parla più propriamente di **tabella delle transizioni**) ha quattro condizioni di stabilità; le due condizioni d'indifferenza discendono dall'ipotesi che la configurazione 11 non sia mai utilizzata in ingresso.

Dalle mappe di figura si ottengono due espressioni minime, dette **equazioni caratteristiche** del latch SR:

(minima SP) **Q = S + R'q**

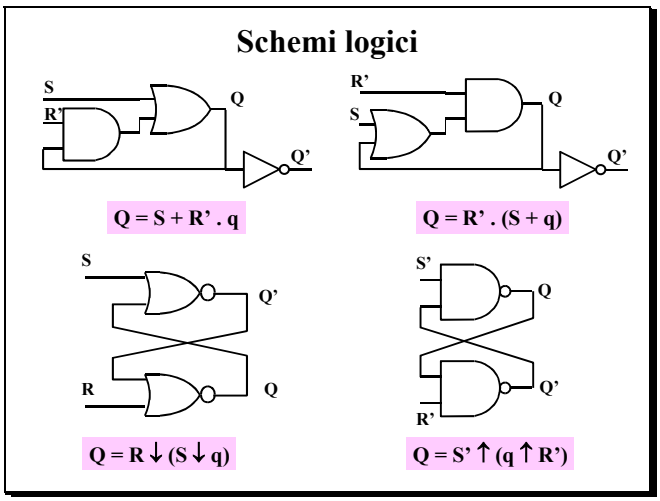
(minima PS) **Q = R'(S + q)**



4: individuazione dello schema logico - Le realizzazioni a AND,OR, NOT, a NOR ed a NAND sono indicate nella figura a lato.

Un breve ragionamento sulle condizioni di corretto impiego. La sequenza d'ingresso tipica del latch è formata da una scrittura di 1 (S=1, R=0), un certo tempo di mantenimento di tale valore (S=0, R=0), una scrittura di 0 (S=0, R=1), un certo tempo di mantenimento di tale valore (S=0, R=0), ecc.: è dunque certo che configurazioni d'ingresso consecutive differiscono per il valore di un solo bit.

Per quanto riguarda la durata dei comandi (v. pag. 72), deve essere $\tau_w \geq 2\,\tau_p$ (il valore minimo è indicato sul data sheet).

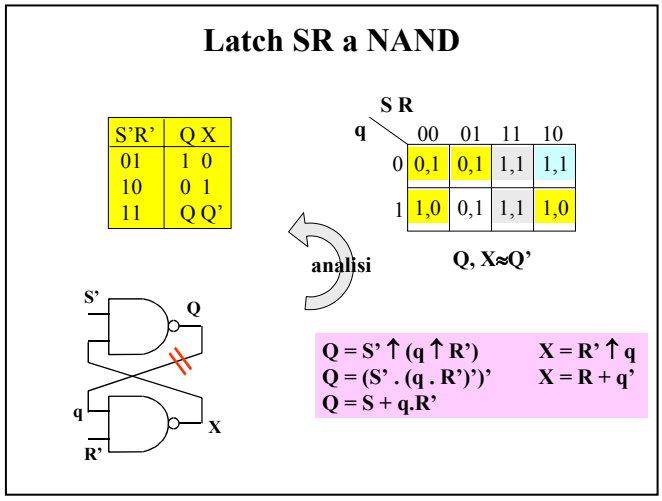


Uscite complementari

Si noti che nelle precedenti realizzazioni a NOR ed a NAND si mette a disposizione il bit Q in forma vera e complementata senza dover ricorrere all'impiego di un NOT, come invece è stato necessario negli schemi a AND, OR.

Questa proprietà merita di essere verificata con un procedimento di analisi.

Consideriamo ad esempio lo schema a NAND. In figura sono riassunti i risultati che discendono dall'esecuzione di 3 passi.



1: individuazione ed eliminazione delle retroazioni - Operando idealmente un taglio subito dopo l'uscita **Q** del circuito, si ottiene una rete combinatoria a tre ingressi e due uscite.

2: individuazione delle espressioni – Indicando con **q** il segnale a valle del taglio e con **X** l'uscita del secondo NAND, si ottengono le espressioni:

- **Q = S' ↑ (q ↑R')** **X = R' ↑ q**.

Trasformandole (v. pag. 86) si ottiene:

- **Q = S + q.R'** **X = R + q'**.

3: individuazione della tabella delle transizioni - Tramite i RR, le otto valutazioni di **Q** e di **X** possono essere agevolmente riportate su una mappa di Karnaugh che rispetti già la forma di una tabella delle transizioni (configurazioni della variabile di stato presente **q** sulle righe e configurazioni degli ingressi **S**, **R** sulle colonne).

N.B. – Si ricordi che uno schema logico combinatorio e la corrispondente espressione descrivono una funzione completamente specificata (T1, pag.58), e questo anche se la specifica del comportamento ha condizioni d'indifferenza.

Sulla precedente mappa è possibile notare che **Q** e **X** hanno valori complementari nelle condizioni di stabilità presenti nelle colonne 00, 01, 10, mentre la proprietà non è vera in colonna 11. La cosa è irrilevante se si rispetta l'ipotesi di non impiegare questa configurazione in ingresso. **Q** e **X** non sono complementari anche nella transizione da **q=0** a **q=1** indicata in colonna 10, ma la cosa questa volta è irrilevante per la breve durata di questo transitorio.

Si può dunque scrivere **X = Q'**.

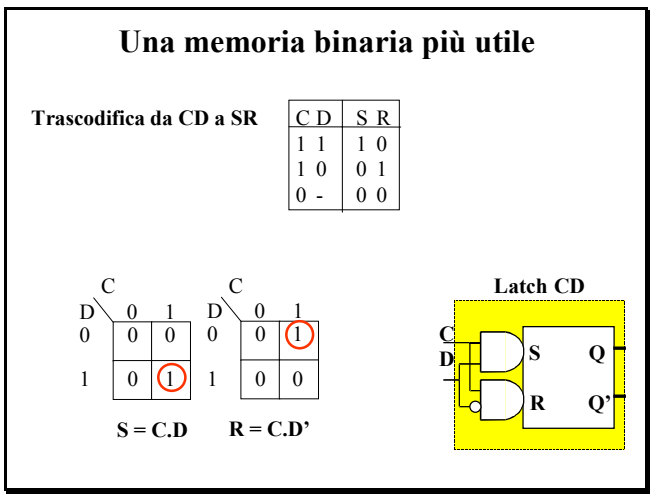
Latch CD

Il latch CD ha ancora due ingressi, C e D appunto, ma li impiega in modo diverso dal latch SR: C=**1** segnala **quando** si vuole scrivere, il contemporaneo valore di D **cosa** si vuole scrivere.

Per la realizzazione si può impiegare un latch SR preceduto da una trascodifica dal codice CD al codice SR.

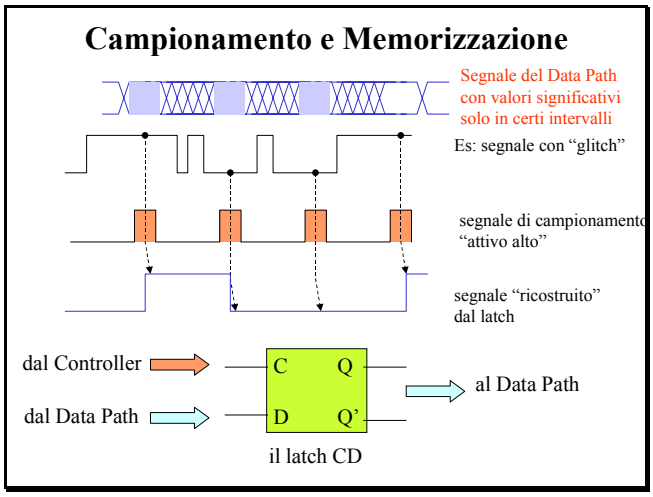
Le mappe di figura indicano che occorre eseguire i prodotti logici **C.D** e **C.D'**.

La struttura può essere utilmente interpretata in modo diverso: un primo "stadio", i due AND, ha il compito di **campionare** il valore di D nell'intervallo di tempo indicato da **C=1**; un secondo "stadio", il latch SR, ha il compito di prendere in consegna il campione raccolto e di **memorizzarlo** per tutto il successivo intervallo di tempo **C=0**.



Questo comportamento è particolarmente utile nella decomposizione Controller-Data Path: spesso infatti i dati da memorizzare hanno valori significativi solo in brevi intervalli del tempo di funzionamento della macchina ed è quindi importante lasciar stabilire al Controllo, cioè a chi conosce lo sviluppo temporale del processo, quando devono o non devono essere presi in considerazione.

In figura è mostrata un'applicazione tipica del latch CD: stabilendo opportunamente gli intervalli di campionamento la forma d'onda di un segnale può essere ripulita da indesiderati andamenti in transitorio.

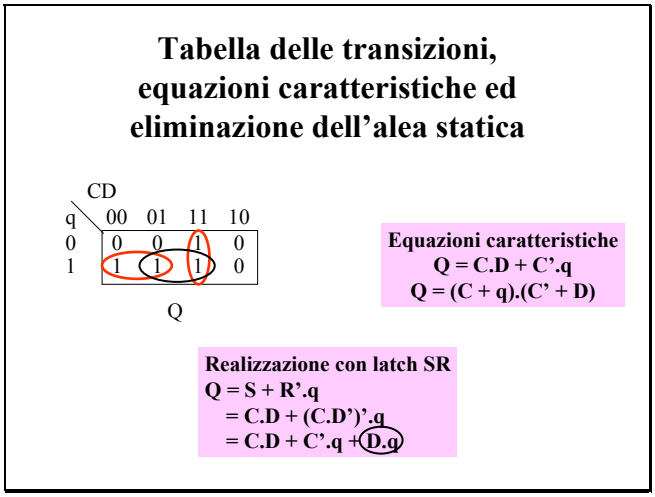


Alea statica

Il latch CD ha anche altre realizzazioni: in tutte è stato necessario preoccuparsi del fenomeno dell'alea statica.

In figura è indicata la tabella delle transizioni che discende dalle specifiche, unitamente alle due **equazioni caratteristiche** che da essa discendono:

- (minima SP) **Q = CD + C'.q**
- (minima PS) **Q = (C + q).(C' + D)**.



A parte una ridenominazione dei segnali, la tabella è proprio quella che abbiamo discusso a pag. 105: l'uso delle espressioni minime nella realizzazione può determinare comportamenti aleatori. Ciò non è vero nel caso del latch SR preceduto da due AND: bastano i pochi passaggi indicati in figura per convincersi che il comportamento è stato infatti ottenuto con una **espressione ridondante**.

Per rispettare **T14**, la funzione di aggiornamento dello stato interno deve dunque essere descritta o dalla somma di tutti gli implicanti primi o dal prodotto di tutti gli implicati primi:

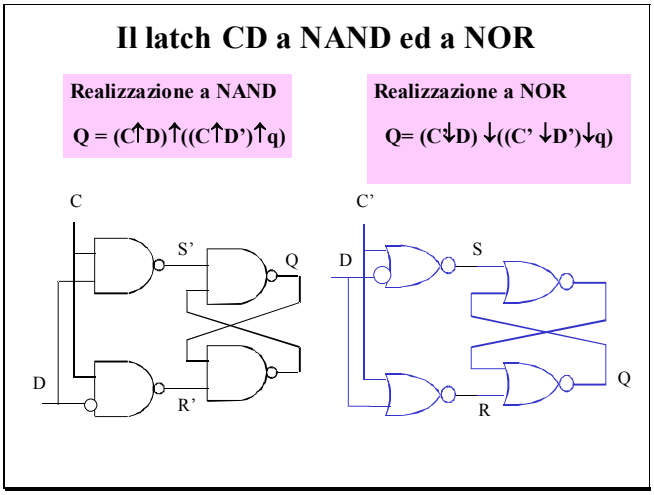
Q = C.D + C'.q + D.q **Q = (C + q).(C' + D).(D+q).**

Nelle usuali realizzazioni a NAND ed a NOR vengono in realtà impiegate espressioni a tre livelli:

- **Q = C.D + (C'+D).q**
- **Q = (C' + D).((C.D)+q).**

Tali espressioni consentono infatti di impiegare solo gate a due ingressi e richiedono un minor numero di interconnessioni. Il tutto però va a spese della rapidità di risposta.

Si noti che la struttura è anche in questo caso formata dalla disposizione in serie di uno "stadio" di campionamento e di uno "stadio" di memorizzazione.



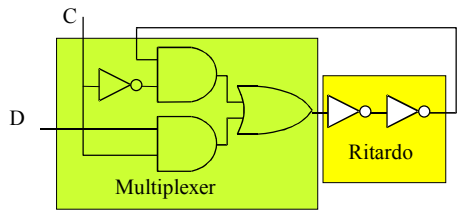
E' interessante prendere atto che la risoluzione del problema dell'alea statica può essere delegata al livello fisico.

Nella famiglia TTL ad esempio la funzione di aggiornamento è stata realizzata anche con un semplice selettore a due vie; il ritardo inerziale (v. pag. 74) dei due NOT posti a valle del MUX è l'accorgimento messo in campo per eliminare l'eventuale *glitch* generato dall'alea statica del selettore.

La complessità del circuito non è per questo maggiore del necessario: i due NOT infatti servivano comunque per fornire all'esterno, in forma vera e complementata, il bit memorizzato.

Una quarta soluzione

$Q = C.D + C'.q$



Durata del transitorio

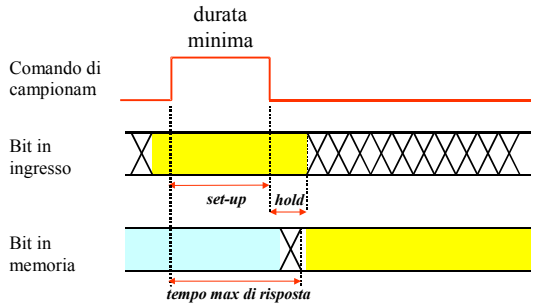
Per la maggiore lunghezza dei percorsi di elaborazione, il transitorio nel latch CD ha una durata superiore a quello del latch SR. Tutto comincia quando C diventa 1.

Il valore di D passa attraverso due o tre gate (a seconda della realizzazione) ed innesca poi la retroazione. Durante questo intervallo di tempo D deve essere **costante**: il tempo necessario per la prima fase è detto **setup time**, quello per la seconda fase **hold time**.

La **durata minima** del comando di campionamento deve essere almeno pari al tempo di setup indicato nel data sheet.

Per poter disporre dell'uscita in forma vera e complementata occorre aspettare ancora un po' di tempo: il dato fornito dai Costruttori di circuiti integrati è detto **response time** ed è misurato a partire dall'inizio del transitorio (scandito o da un fronte di salita di C, o, quando C è già a 1, da un fronte di salita o di discesa di D).

I tempi di *set-up*, di *hold* e di *risposta*



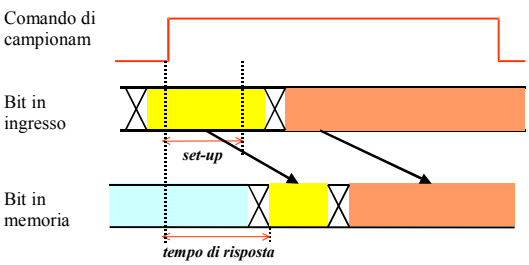
Uscite trasparenti

Se il comando di campionamento ha una durata più lunga della minima e se, contemporaneamente, D modifica il suo valore, tale valore si riproduce anche sull'uscita.

Il fenomeno, detto delle **uscite trasparenti**, è giustificato in figura tramite la tabella delle transizioni.

Per evitare malfunzionamenti (il valore in uscita, a causa della durata del transitorio, può infatti essere o l'ultimo od il penultimo valore assunto da D nell'intervallo C=1) è pressoché indispensabile garantire **la costanza di D durante tutto il tempo di campionamento**.

L'uscita "trasparente"



q \ CD	00	01	11	10
0	0	0	1	0
1	1	1	1	0

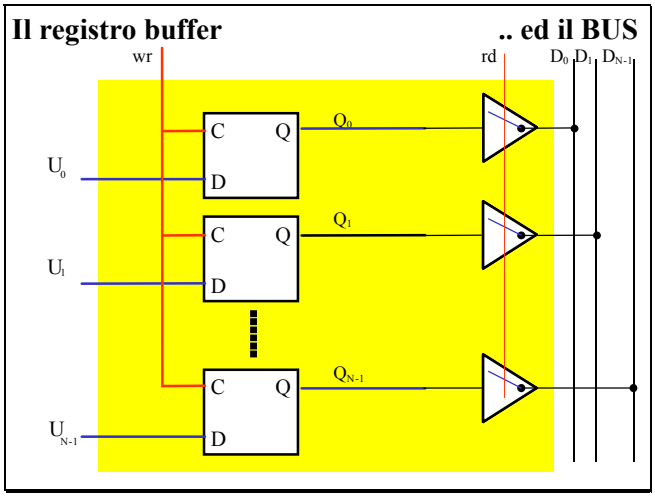
Q

CASO DI STUDIO - A livello architettonico il latch CD viene normalmente impiegato come componente del **registro buffer** (una memoria temporanea che le unità funzionali del calcolatore impiegano per "parcheggiare" dati in arrivo ed in partenza).

In figura è mostrato un **buffer di n bit** interposto tra le uscite U_0, U_1, U_{N-1} di una unità funzionale ed il bus che consente di inoltrarne il contenuto ad una qualsiasi delle altre unità funzionali.

L'unità che deve trasmettere il dato attiva il comando di scrittura **wr** quando è certa che è "valido" ed attiva il comando di lettura **rd** quando è certa di poter disporre del bus.

Anche l'unità che riceve il dato si avvale di un buffer: in questo caso gli ingressi sono connessi ai segnali del bus ed il comando **wr** viene attivato solo quando si è certi che trasportino valori validi.

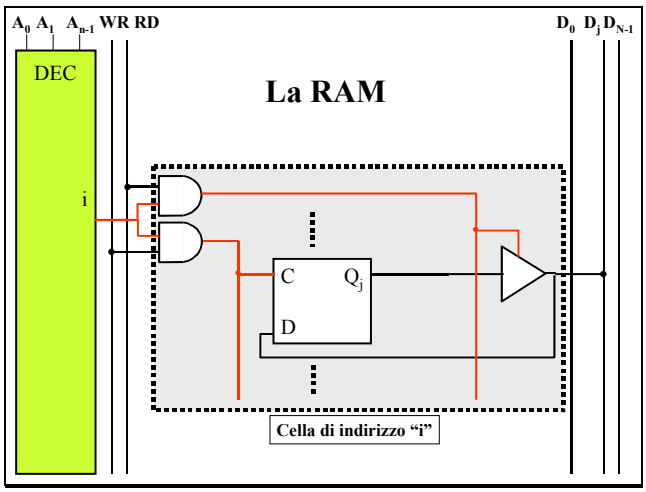


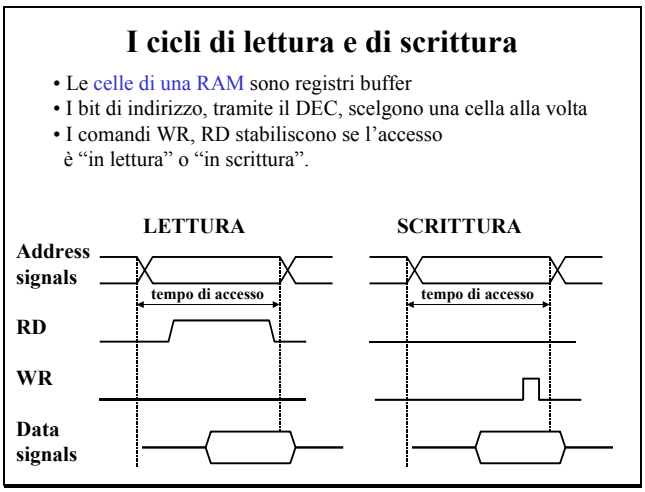
APPROFONDIMENTO – La RAM (*Random Access Memory*), il componente fondamentale della memoria principale di un calcolatore, è un caso particolarmente importante di "parcheggio" di dati. Vediamo uno schema di principio della organizzazione interna nel caso di **n** bit d'indirizzo e **N** bit di dato.

La memoria è suddivisa in $\mathbf{2^n}$ **celle**, contenenti ciascuna **N latch CD**.

Un **Decoder** riceve i bit d'indirizzo e genera in corrispondenza il comando con cui viene selezionata la cella su cui si vuole eseguire o un'operazione di scrittura (comando **WR** attivo alto), o un'operazione di lettura (comando **RD** attivo alto).

Per il corretto impiego del componente occorre assegnare una opportuna temporizzazione a indirizzi, dati e comandi: un possibile sviluppo temporale dei due cicli d'accesso è indicato in figura.





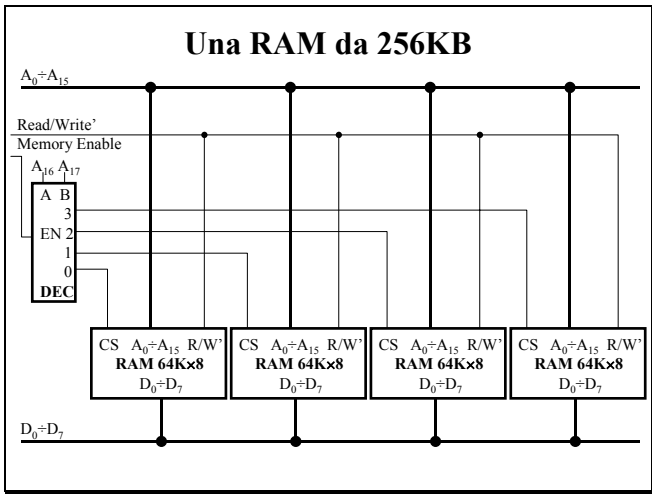
In realtà, come abbiamo già visto per le ROM, anche per i chip di RAM è previsto un comando CS (*chip select*) per semplificare la connessione ad un bus di più componenti.

In questo caso il secondo comando è indicato con R/W' e genera, unitamente a CS, tre differenti modalità di funzionamento del componente:

- **CS = 0 e R/W' = -** lo isola dal bus,
- **CS=1 e R/W'= 1** lo abilita alla lettura,
- **CS=1 e R/W'= 0** lo abilita alla scrittura.

CASO DI STUDIO - Supponiamo che un processore con 8 bit di dato e 18 bit d'indirizzo debba essere dotato di una memoria principale di 256 KB.

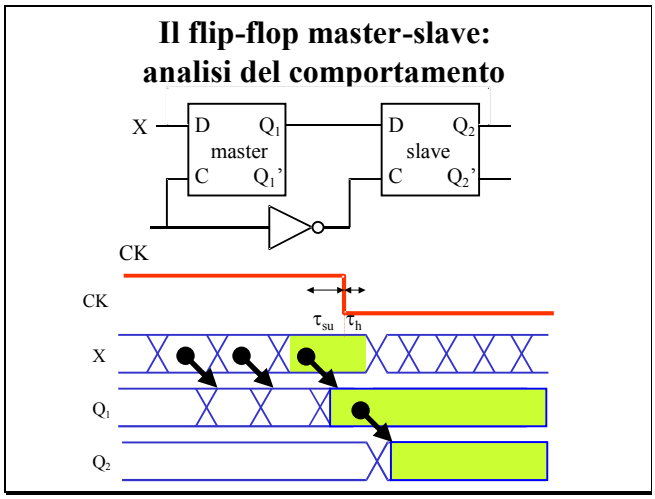
In figura è mostrato come ciò possa essere agevolmente conseguito usando 4 chip con 16 bit d'indirizzo e 8 bit di dato.



Flip-flop master-slave

Il latch CD, **da solo**, non può essere impiegato sulle retroazioni di un circuito sequenziale sincrono: in questa situazione non è infatti possibile garantire che le variabili di stato restino costanti durante l'intervallo in cui è attivo il comando di campionamento.

Due latch CD, disposti in cascata ed azionati da un segnale CK in forma vera ed in forma complementata, vanno invece bene. Lo schema è detto **flip-flop master-slave** e non presenta più il difetto dell'uscita trasparente: con CK alto è solo l'uscita Q_1 del master a riprodurre le variazioni dell'ingresso X; non appena CK diventa basso il master memorizza l'ultimo valore campionato e lo consegna per il campionamento allo slave. L'uscita Q_2 si può dunque modificare solo in corrispondenza dei fronti di discesa di CK (istanti di sincronismo).

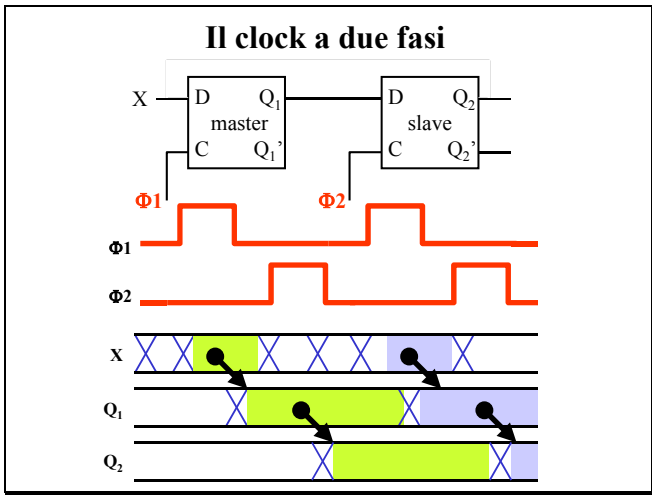


APPROFONDIMENTO - Il flip-flop master-slave ha un potenziale malfunzionamento: se il ritardo del NOT su CK è maggiore del tempo di risposta del master, un valore appena campionato dal master può infatti immediatamente propagarsi sull'uscita dello slave.

Il problema è stato affrontato e risolto sia a livello tecnologico, sia a livello logico.

Nel primo caso è stata posta ogni cura nel ridurre al minimo il ritardo di propagazione del NOT.

Nel secondo caso, evidenziato in figura, è stato introdotto il cosiddetto principio del **clock a due fasi**, oggi molto impiegato nella realizzazione dei µprocessori: i campionamenti del master e dello slave sono affidati a due segnali impulsivi distinti, di uguale periodo, ma anche adeguatamente sfasati uno dall'altro per garantire che i due comandi non si sovrappongano mai.



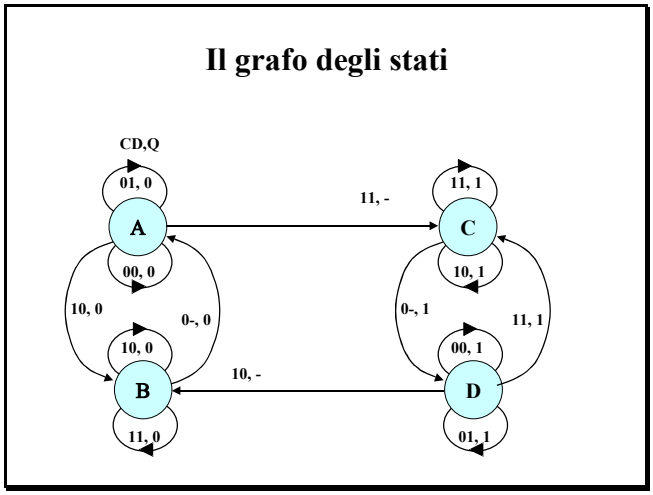
Flip-flop edge-triggered

Una diversa e più sicura realizzazione del flip-flop è stata individuata con l'aiuto del calcolatore a partire dal grafo di Moore che abbiamo esaminato nel cap.3, pag. 42.

In figura è mostrato il grafo di Mealy equivalente. Il campionamento di D avviene in corrispondenza dei fronti di salita di C; si noti la condizione d'indifferenza apposta sui rami che comportano una modifica di Q.

E' istruttivo esaminare i passi di progetto che hanno portato ad individuare lo schema a NAND di questa importante memoria binaria.

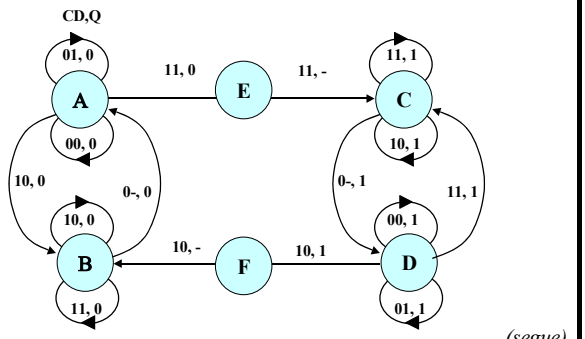
Dal grafo degli stati si evince immediatamente che è possibile codificare gli stati con due soli bit, eliminando a priori il pericolo di corse critiche: associando ad esempio 00 ad A, 01 a B, 11 a D e 10 a C, tutte le transizioni di stato richiedono la variazione di un solo bit.



Pur non essendo teoricamente necessario, il CAD ha dimostrato conveniente l'uso di un codice ridondante a 3 bit. In questa maniera infatti ci si può avvalere di due ulteriori stati (E ed F

nella sottostante figura) per compiere due **transizioni multiple** in corrispondenza dei campionamenti che comportano una variazione d'uscita (colonna 10 e colonna 11). La tabella delle transizioni ha dunque otto righe, due delle quali corrispondono a stati non utilizzati (000 e 100) e contengono quindi solo condizioni d'indifferenza.

Le transizioni multiple



(segue)

La tabella delle transizioni

stato		CD 00	01	11	10
	000	---,-	---,-	---,-	---,-
E =	001	---,-	---,-	101,-	---,-
A =	011	011,0	011,0	001,0	010,0
B =	010	011,0	011,0	010,0	010,0
	100	---,-	---,-	---,-	---,-
C =	101	111,1	111,1	101,1	101,1
D =	111	111,1	111,1	101,1	110,1
F =	110	---,-	---,-	---,-	010,-
	$y_1y_2y_3$	$Y_1Y_2Y_3$,Q			

(segue)

Dalla tabella si ottengono quattro mappe, rispettivamente per Q (la variabile d'uscita) e per le tre variabili di stato futuro Y_1, Y_2, Y_3.

Sintesi a NAND (1)

$y_1y_2y_3$	CD 00	01	11	10
0 0 0	-	-	-	-
0 0 1	-	-	-	-
0 1 1	0	0	0	0
0 1 0	0	0	0	0
1 0 0	-	-	-	-
1 0 1	1	1	1	1
1 1 1	1	1	1	1
1 1 0	-	-	-	-

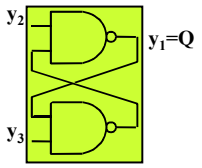
Q

$Q = y_1$

Sintesi a NAND (2)

$y_1y_2y_3$	CD 00	01	11	10
0 0 0	-	-	-	-
0 0 1	-	-	1	-
0 1 1	0	0	0	0
0 1 0	0	0	0	0
1 0 0	-	-	-	-
1 0 1	1	1	1	1
1 1 1	1	1	1	1
1 1 0	-	-	-	0

Y_1



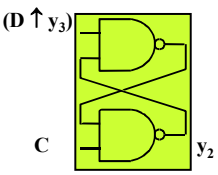
$Y_1 = y_2' + y_3.y_1$
$Y_1 = y_2 \uparrow (y_3 \uparrow y_1)$

Sempre il CAD ha suggerito di impiegare una copertura non minima per Y_2: in questo modo infatti è possibile ottenere un'espressione a tre livelli per la cui realizzazione ci si può avvalere della uscita di un NAND già presente nella realizzazione di Y_3.

Sintesi a NAND (3)

$y_1y_2y_3$	CD 00	01	11	10
0 0 0	-	-	-	-
0 0 1	-	-	0	-
0 1 1	1	1	0	1
0 1 0	1	1	1	1
1 0 0	-	-	-	-
1 0 1	1	1	0	0
1 1 1	1	1	0	1
1 1 0	-	-	-	1

Y_2

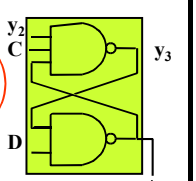


$Y_2 = C' + D'.y_2 + y_2.y_3' = C' + y_2.(D'+y_3')$
$Y_2 = C \uparrow (y_2 \uparrow (D \uparrow y_3)$

Sintesi a NAND (4)

$y_1y_2y_3$	CD 00	01	11	10
0 0 0	-	-	-	-
0 0 1	-	-	1	-
0 1 1	1	1	1	0
0 1 0	1	1	0	0
1 0 0	-	-	-	-
1 0 1	1	1	1	1
1 1 1	1	1	1	0
1 1 0	-	-	-	0

Y_3



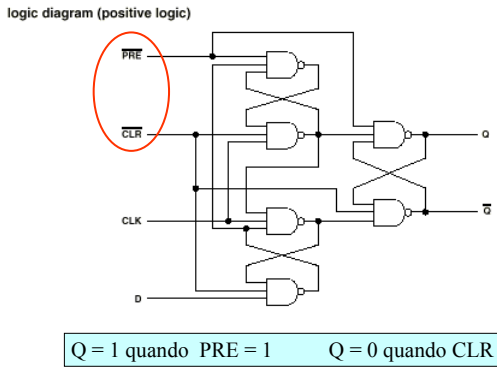
$Y_3 = C' + D.y_3 + y_2'$
$Y_3 = C \uparrow y_2 \uparrow (D \uparrow y_3)$

La realizzazione di un flip-flop edge-triggered impiega dunque sei NAND (4 per il campionamento e 2 per la memorizzazione) contro gli otto necessari nella struttura master-slave; il clock non richiede inoltre la presenza di un NOT, aspetto che abbiamo detto essere critico nel master-slave. Un ultimo vantaggio è che il segnale D entra in ingresso ad un solo NAND.

Nella figura a lato, tratta da un *data sheet* della famiglia TTL, è rappresentata l'intera struttura. Per aumentare la flessibilità d'impiego, il Costruttore ha messo a disposizione del progettista logico anche due comandi **asincroni**, detti **preset** e **clear**.

Tali comandi sono **prioritari** rispetto ai segnali di clock e di "dato": quando uno dei due ha valore 1 (per mantenere le uscite una il complemento dell'altra non devono mai essere attivati contemporaneamente) l'uscita Q è immediatamente portata e poi mantenuta o al livello H o al livello L.

Il flip-flop di tipo D della famiglia TTL: comandi sincroni e asincroni ('74)



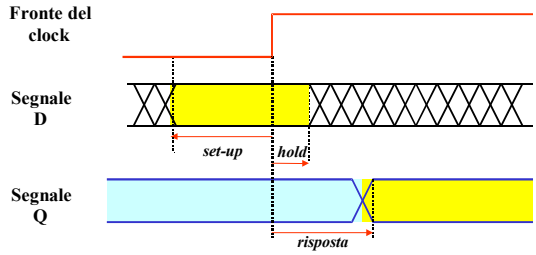
Q = 1 quando PRE = 1 Q = 0 quando CLR = 1

Il transitorio del flip-flop è un po' più lungo di quello del latch CD. Per un corretto impiego sono significativi i seguenti tempi:

- **setup** – intervallo minimo di tempo precedente il fronte del clock, durante il quale il segnale D deve essere costante;
- **hold** – intervallo minimo di tempo successivo al fronte del clock, durante il quale il segnale D deve ancora mantenere costante il suo valore;
- **response** – intervallo massimo di tempo successivo al fronte del clock durante il quale non è ancora garantito che il circuito presenti valori complementari in uscita.

I valori di setup, di hold e di response sono indicati nel data sheet

I tempi di *set-up*, di *hold* e di *risposta*



In certe realizzazioni del flip-flop occorre anche preoccuparsi della "ripidità" del fronte del clock; in questi casi il data sheet fornisce la massima durata ammessa per questo transitorio.

Per non complicare inutilmente lo schema logico delle reti sequenziali sincrone il dettaglio della struttura interna viene nascosto all'interno di un blocco. Sono in uso due differenti simboli.

L'ingresso del clock è sempre indicato da un piccolo **triangolo** e ciò consente di differenziare a colpo d'occhio il flip-flop dal latch CD; abbiamo già visto infatti che in questo secondo caso l'ingresso del segnale di campionamento è indicato solo dalla lettera C.

Il triangolo può poi essere o meno preceduto da un **pallino**.

Se il campionamento è fatto dai **fronti di discesa** del clock (*negative edge triggering*) il pallino è presente; se al contrario vengono usati i **fronti di salita** (*positive edge triggering*) il pallino è assente.

Simboli logici del flip-flop