CAPITOLO 11

Funzioni Sequenziali MSI Integrate

11.1 Registri

Quando l'informazione binaria è memorizzata in un gruppo di F/F esso è chiamato *registro*. Un gruppo di *n* F/F è chiamato registro ad *n* bit, ed è capace di immagazzinare una informazione di *n* bit. Il registro più semplice da usare e/o realizzare è formato da F/F di tipo D con caratteristiche di tipo Latch e di tipo Edge-Triggered o Master/Slave. In un circuito

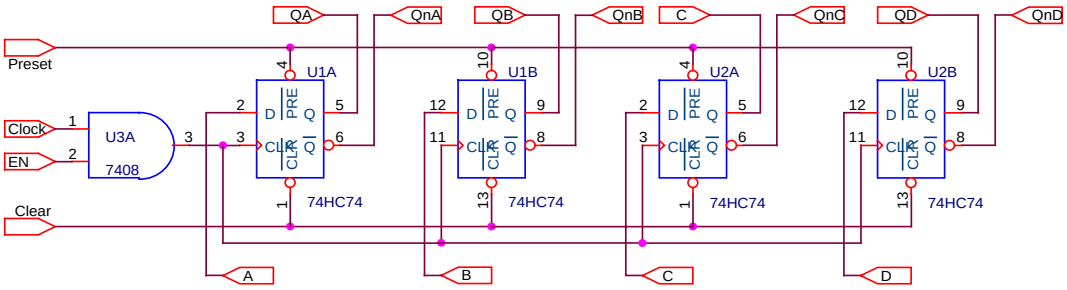


Fig. 11.0

integrato, normalmente, ce ne sono 4 o 8 ed è chiamato Storage Register.
La Fig. 11.0 ne mostra uno a 4 bit. Essi normalmente hanno in comune gli ingressi di clock, di Preset, di Clear e di Enable, mentre, sia gli ingressi (A, B, C, D) che le uscite (Q_A,Q_B,Q_C,Q_D) sono accessibili singolarmente dai pin assegnati dell'integrato. A seconda del numero di piedini, del contenitore dell'integrato, i segnali di uscita e di controllo possono essere parzialmente disponibili. L'ingresso Enable serve ad abilitare o indirizzare il registro: questo si dice abilitato o indirizzato se Enable = 1. L'ingresso clock normalmente viene chiamato impulso di scrittura o strobe. A seconda del tipo di F/F utilizzato il registro è detto di tipo Lacht, o **non** Latch.

- Nel primo caso, la descrizione che lo accompagna dirà su quale stato dello strobe il F/F è trasparente; in questo stato il dato di ingresso viene trasferito nel registro e quindi diventa subito disponibile in uscita, e ne segue le variazione dell'ingresso, quando lo strobe è nello stato complementare (al precedente), nel registro, avviene il congelamento o latching, dello stato dei dati di ingresso.
- Nel secondo caso, la descrizione ci fornisce l'informazione su quale fronte della transizione del clock, i F/F immagazinano i dati d'ingresso. Questi restano congelati nel registro fino all'arrivo del prossimo fronte attivo. Il registro di Fig. 11.0 è un registro **non Latching** e acquisisce o carica i dati di ingresso sui fronti positivi del clock.

11.2 Registri a Scorrimento (Shift Register)

Come già detto, uno shift register è una serie di F/F interconnessi in modo da essere usati per la memorizzazione temporanea di dati. I dati entrano nel primo F/F della serie e procedono da un F/F al successivo alla ricezione di un fronte attivo del clock. Un circuito integrato contenete uno Shift Register consiste di 4, 5, 8,… fino a migliaia di F/F, che sono interconnessi serialmente (in cascata) in modo tale che l'uscita di uno va all'ingresso del successivo. I F/F ricevono simultaneamente l'impulso di clock e pertanto, il dato che è immagazzinato in un F/F e il bit che si trova in ingresso del primo F/F, sono spostati a destra di una posizione a ciascun arrivo dell'impulso di clock.
Per ottener uno Shift Register più lungo di quello che sono normalmente presenti in un circuito integrato possono essere connessi in cascata un numero opportuno di essi. Per costruire Shift Register di una particolare lunghezza o aventi altre particolari caratteristiche si possono anche usare singoli F/F. (Notare che sono anche disponibili shift register di 1000 e più bit. Tali shift register sono chiamati memorie ad accesso sequenziale).

Le equazioni di stato dei F/F di uno shift register possono essere scritte come:

$$Q_1(t+1) = SI;\ Q_2(t+1) = Q_1(t);\ Q_3(t+1) = Q_2(t);\ \ldots\ Q_n(t+1) = Q_{n-1}(t).$$

Notare che con SI (**S**erial-**I**nput) abbiamo indicato la variabile esterna allo Shift Register, essendo il F/F Q_1 il primo F/F della serie. Dalle equazioni di stato si vede che il modo più semplice per realizzare uno shift register è quello di usare dei F/F D, tuttavia essendo l'equazione caratteristica di

- un F/F JK $Q(t+1)= JQ'+K'Q$ se K' = J (cioè si connette all'ingresso K la variabile di ingresso J complementata) si ottiene $Q(t+1)= J(Q+Q')= J$ quindi un F/F D,
- un F/F SR $Q(t+1)= R'Q+S$ con la condizione $RS= 0$ se R'=S (cioè si connette all'ingresso R la variabile di ingresso S complementata) si ottiene Q(t+1)=SQ+S = S quindi un F/F D.

Pertanto gli Shift Register, che si trovano nei circuiti integrati, sono costruiti utilizzando F/F di tipo JK, D e RS indifferentemente. La loro differenza consiste, essenzialmente, in altri fattori e cioè nel modo in cui i dati entrano o escono da essi. In uno Shift Register i dati possono essere caricati in molti modi. Per es. possono essere caricati serialmente, come descritto dalle equazioni di stato sopra riportate, "cloccandoli" all'ingresso del primo F/F e quindi spostandoli a destra attraverso gli altri registri. Possono essere caricati parallelamente in tutti i F/F, connettendo le singole linee di dati ai rispettivi ingressi dei F/F. Infine i dati possono essere caricati in modo asincrono, attraverso linee separate di dati connessi agli ingressi Preset di ciascun F/F.

- Se tutti i F/F di uno Shift Register sono caricati simultaneamente esso è detto ***Parallel-In***.

- Se i F/F sono caricati uno ad uno, con i dati che entrano soltanto dall'ingresso del primo F/F, il registro è chiamato ***Serial-in***.

All'uscita di uno Shift Register i dati memorizzati possono essere disponibili in vario modo.

- Se lo Shift Register ha disponibile, ad ogni istante, i dati di tutti i F/F, è chiamato ***Parallel-Out***.
- Se lo Shift Register ha disponibile in uscita solo il dato dell'ultimo F/F è noto come ***Serial-Out***. È comune trovarlo nei casi in cui non è possibile portare ogni uscita dei F/F in uno dei piedini dell'integrato, per mancanza degli stessi. In questi casi per ottenere l'intero dato è necessario fare scorrere l'intero contenuto attraverso l'unica uscita.

Uno dei più importanti usi di uno Shift Register è nella conversione del formato dei dati. Se è necessario convertire i dati da una sola linea (seriali) in parallelo (molte linee) viene usato uno shift register ***Serial-In/Parallel-Out*** (SIPO). In questa unità logica entra una serie di bit di dati da una sola linea che poi sono messi a disposizione simultaneamente su molte linee di uscita. Inversamente, una conversione Parallelo-Serie è fatta utilizzando uno Shift Register **Parallel-In/Serial-Out** (PISO). In questo caso dopo che il dato, presente sugli ingressi paralleli, è stato caricato nello Shift Register, bisogna applicare un adeguato numero di impulsi di clock perché possa essere ricevuto completamente in uscita.

Alcuni Shift Register sono configurati per permettere sia l'aquisizione dei dati in parallelo sia lo scorrimento degli stessi o destra o a sinistra. Questi Shift Register sono chiamati ***Shift Register Universali***, essendo capaci di:

1) Caricare ed uscire i dati il Parallelo
2) Caricare i dati in serie sia da destra, sia da sinistra
3) Spostare i dati in entrambe le direzioni
4) Uscire i dati serialmente sia da destra, sia da sinistra.

Quando un dato immagazzinato in uno Schift Register è interpretato come un numero gli scorrimenti si distinguono in **logici** e **aritmetici**.

Uno scorrimento, sia a sinistra sia a destra, si dice **logico** quando il bit che entra rispettivamente da destra e da sinistra è uno zero logico. L'interpretazione numerica del risultato (considerato non segnato) ci dice che in tali casi il numero che risulta dopo ogni scorrimento è stato moltiplicato per due (scorrimento a sinistra) o diviso per due (divisione intera). Per esempio:

- uno scorrimento logico a destra di 0111 (7 decimale) produce 0011 (tre decimale); in altri termini si perde il resto.
- Uno scorrimento logico a sinistra dello stesso numero produce 1110 (14 decimale).

Quando, l'interpretazione numerica del dato, ne presume una codifica in complemento a due è necessario, nello scorrimento, conservare le informazioni del segno; se il numero è positivo il

bit di segno deve restare zero, se negativo deve restare 1. In questi casi lo scorrimento di dice **aritmetico**.

Uno scorrimento **aritmetico a destra** è fatto connettendo all'ingresso Serial-In il bit più significativo del numero (che coincide con il bit di segno). Vediamo, con esempi, cosa succede al numero di partenza dopo qualche scorrimento aritmetico:

- Numero negativo (1)010= $(-6)_{10}$=> (1)101= $(-3)_{10}$ => (1)110= $(-2)_{10}$; ad ogni scorrimento abbiamo una divisione intera per due con approssimazione (in modulo) in eccesso.
- Numero positivo (0)110=$(+6)_{10}$=> (0)011= $(+3)_{10}$ => (0)001= $(+1)_{10}$; ad ogni scorrimento abbiamo una divisione intera per due con approssimazione in difetto

Uno scorrimento **aritmetico a sinistra** è fatto connettendo all'ingresso Serial-In, zero e **non** facendo partecipare allo scorrimento il bit di segno (il bit più significativo). Per Es.

- Numero negativo (1)110= $(-2)_{10}$=> (1)100= $(-4)_{10}$ => (1)000= $(-8)_{10}$;
- Numero positivo (0)0001= $(+1)_{10}$=> (0)0010= $(+2)_{10}$ => (0)0100= $(+4)_{10}$;

Riassumendo: ripetuti scorrimenti aritmetici a destra equivalgono a dividere il numero per potenze intere successive di due, ripetuti scorrimenti a sinistra equivalgono al prodotto del numero per potenze intere di due. Tale caratteristica li rende utili nelle **U**nità **L**ogiche

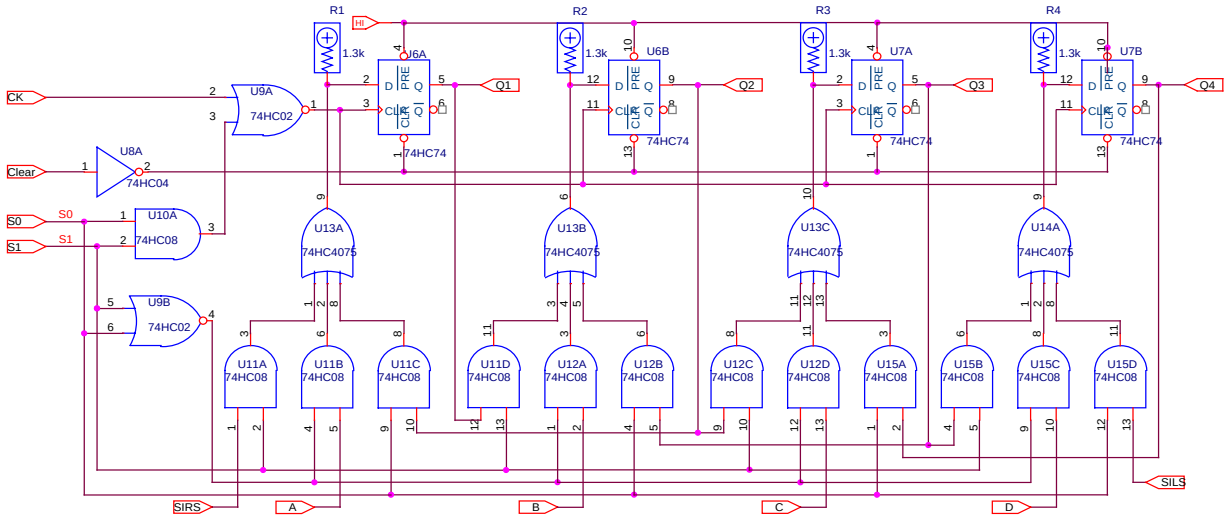


Fig. 11.1

Aritmetiche (ALU), in cui il prodotto tra due numeri può essere effettuato con una serie iterativa di scorrimenti a sinistra del moltiplicatore e la sua somma, con il risultato ottenuto con una precedente, analoga operazione.

La Fig. 11.1 mostra la possibile realizzazione di uno Shift Register Universale a 4 bit. In questo Shift Register notiamo che:

a) Le uscite da Q1 a Q4 sono usate come uscite parallelo, e per le operazioni di shift sia a sinistra, sia a destra

b) L'uscita del F/F Q1 è usata anche per l'uscita seriale a sinistra

c) L'uscita del F/F Q4 è usata anche per l'uscita seriale a destra.

Può essere scelta una operazione tra quattro differenti ed esclusive. Ogni operazione è codificata con un numero binario a due bit, (S0, S1). La Tab. 11.1 ne fornisce la codifica e le funzioni Booleane di decodifica:

Per realizzare le superiore funzioni l'ingresso D di ogni F/F è collegato ad un Select-OR a tre ingressi.

La funzione Booleana all'ingresso D dell'iesimo F/F realizzata dal relativo Select-OR è:

$D_i = P_L\ I_i + S_R\ Q_{i-1} + S_L\ Q_{i+1}$ (i = 1,2,3,4) con.

Q_0 = SIRS (Ingresso Seriale per lo Shift a destra),

Q_5 = SILS (Ingresso Seriale per lo Shift a sinistra)

I_i rappresenta l'iesimo bit dell'ingresso parallelo

S0	S1	Descrizione Operazione	Funzione di decodifica
0	0	Parallel Load	$P_L=S_0' S_1'$
0	1	Shift Right;	$S_R= S_1$
1	0	Shift Left;	$S_L=S_0$
1	1	Clock Inhibit	$C_I=S_0 S_1$

Tab. 11. 1

Nota Bene: Nel seguito non riporteremo altri schemi logici circuitali che realizzano il solo scorrimento a sinistra, il solo scorrimento a destra (con caricamento o meno parallelo), etc. Questi circuiti si possono ottenere semplicemente da quello di Fig. 11.1 per es. eliminando alcune linee di ingressi di controllo e/o l'elettronica di decodifica, limitando il numero di ingressi del Select-OR e/o eliminandolo del tutto. Il lettore si eserciti a trovare il circuito logico relativo di una delle tante modalità richiamate.

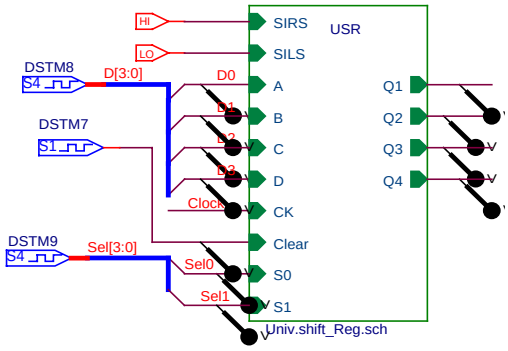


Fig. 11.2

Il timing di Fig. 11.3 è stato realizzato con il circuito di Fig. 11.1, che è il circuito gerarchicamente inferiore al blocco PSPICE di Fig. 11.2.

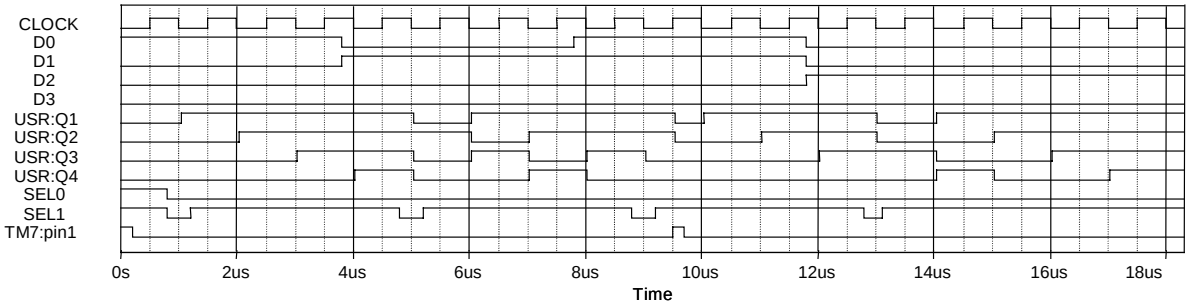


Fig. 11.3

Le traccie del timing si riferiscono ai segnali a cui sono connessi i probe di Fig. 11.2.

Dal timing si noti che sono mostrare alcune delle capacità di programmazione: un Clear iniziale, che azzera tutti i F/F, seguito da un Parallel Load, che carica nei F/F i dati presenti selle linee di ingresso (A, B, C e D) essi sono presenti come stato futuro nelle uscite; rispettivamente (Q1, Q2, Q3 e Q4) al primo impulso di clock. Prima dell'arrivo del successivo impulso di clock la selezione è cambiata in Shift Right ed resta in tale stato per ulteriori tre impulsi di clock. Quindi per altre due volte sono eseguite le operazione di caricamento parallelo e shift seriale. A circa 9,5 μs è dato un altro impulso di clear e l'operazione continua con lo shift right. Notare durante le operazioni di shift a destra, l'1 logico, presente nell'ingresso **S**erial_**I**n/**R**ight-**S**hift (SIRS), entra nel primo F/F e quindi si propaga agli altri della serie.

Si noti che nei circuiti MSI è comune trovare che nello stesso circuito integrato si possano scegliere più funzioni logiche. Ciò accade soprattutto quando, con lo stesso insieme di Flip Flop o registro, variando in modo semplice la parte combinatoria, possono essere realizzate funzione logiche simili. Nel nostro caso nello stesso integrato abbiano uno Shift Register che può fare fino a 4 operazioni diverse. Più avanti si vedrà il caso di un contatore che può contare Up (in aumento) o Down (in diminuizione). I diversi modi di funzionamento sono scelti utilizzando delle linee di controllo o di programmazione. Queste debbono essere mutuamente esclusive in quanto può essere eseguita solo una funzione alla volta. Quando il numero di funzioni è maggiore di due è possibile che la parola di controllo da fornire, per scegliere una funzione, sia codificata allo scopo di diminuire il numero di linee esterne di controllo (come nel nostro caso dell'Universal Shift Register). In tali casi un decoder interno determina la funzione selezionata. È interessante anche notare che le linee di controllo possono essere collegate a dei valori logici fissi o a dei valori logici che possono variare in funzione di variabili esterne o che derivano da funzioni Booleane delle stesse. Questo è il principio su cui si basa la così detta *logica microprogrammata*.

11.3 Forme Speciali di Shift register

Gli shift register ed i contatori hanno molte caratteristiche in comune; premettiamo qui la Definizione di Contatore

Un circuito sequenziale che passa attraverso una fissata sequenza di stati al succedersi degli impulsi di ingresso si chiama contatore. Gli impulsi di ingresso prendono il nome di *impulsi di conteggio*, questi possono essere impulsi di clock o possono provenire da una sorgente esterna e possono verificarsi in modo periodico o casuale. La sequenza di stati di un contatore può seguire un conteggio binario o qualsiasi altra sequenza di stati. I contatori fanno parte di quasi tutti i sistemi contenente logica digitale. Essi sono usati per contare il numero di eventi e sono utili per generare sequenze temporali con le quali un sistema digitale può essere controllato.

Con l'aggiunta di qualche porta logica ad uno Shift Register è possibile modificare il cammino dei dati tra i F/F in modo da farlo diventare un contatore. Un contatore può generare un certo numero di stati diversi è può essere spinto ad andare attraverso questi stati in modo ripetitivo.

I così detti *Ring Counter* e *Johnson Counter* sono circuiti in cui la relazione tra gli Shift Register e contatori è più evidente. Un contatore ad anello è semplicemente uno Shift Register in cui l'uscita dell'ultimo F/F è connessa al suo ingresso seriale (l'ingresso del primo F/F). In altri termini $Q_1(t+1) = Q_n(t)$.

Un **contatore ad anello** è, in genere, caricato con un solo bit ad 1 nel primo F/F e 0 in tutti gli altri F/F e quindi l'uno logico è fatto circolare ad ogni impulso di clock ripetutamente attraverso il registro. Una sua possibile realizzazione, utilizzando solo quattro F/F, si trova in Fig. 11.4, in cui l'ingresso, chiamato !Load (attivo basso) automaticamente presetta il primo F/F e resetta tutti gli altri.

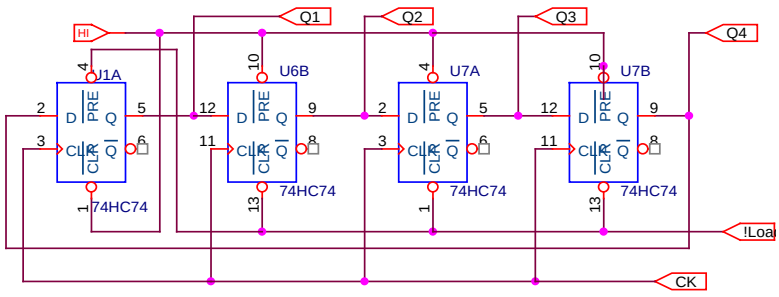


Fig. 11.4 Contatore ad anello

Come risultato di tale configurazione si ha che il contatore passa da un insieme di stati differenti che possono essere usati per determinare una sequenza di operazioni di qualche sistema o di un altro circuito logico che effettua operazioni differenti in tempi diversi. Per es. potrebbe costituire la sequenza di una macchina utensile o di una linea di assemblaggio, o di un calcolatore elettronico che esegue delle somme ripetute. Il maggiore vantaggio di un contatore ad

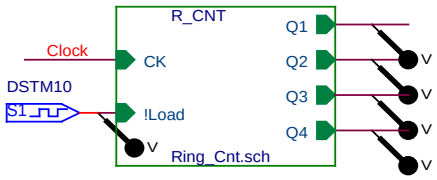


Fig. 11.5

anello è che, contrariamente a qualsiasi altro contatore, non necessita di decodifica: ogni sua linea di uscita può essere connessa direttamente al circuito o all'unità che deve essere attivata. Il timing di Fig. 11.6 dimostra come le uscite sono attive una alla volta sequenzialmente. Tuttavia i contatori ad anello hanno anche molte limitazioni.

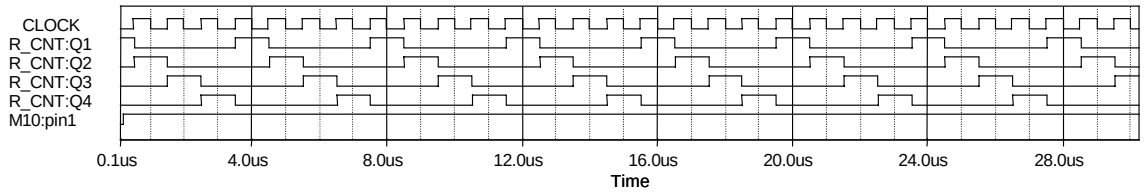


Fig. 11.6

1) Il circuito non fa un uso efficiente di tutti i F/F. Un contatore ad anello a 4 bit può generare soltanto 4 stati diversi, mentre un contatore a 4 bit con codifica binaria ne può generare 16. Per dirla con maggiore generalità, un contatore ad anello ha N stati, ma un contatore binario 8421 ne ha 2^N, dove N è il numero di F/F del contatore.
2) Se un F/F va a finire in uno stato incorretto, a causa di un impulso di rumore o di un malfunzionamento, si verificherà una sequenza erronea e il contatore continuerà a circolare attraverso una sequenza non corretta. Infatti non tutti gli stati possibili sono utilizzati quindi, se il contatore entra in uno di questi stati potrebbe seguire una sequenza tale da non raggiungere mai uno stato della sequenza corretta.

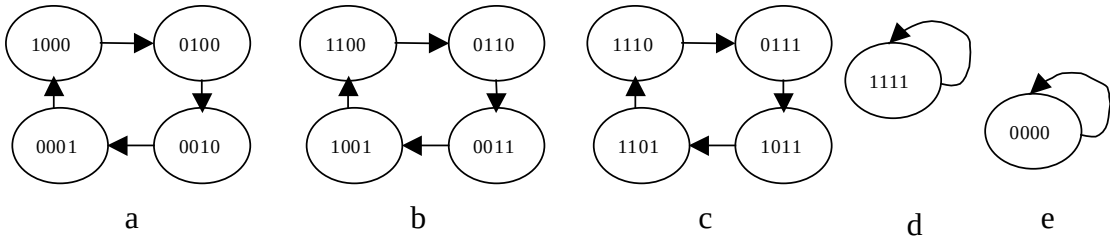


Fig. 11.4bis

Se analizziamo il circuito di Fig. 11.4 a partire da una stato in cui solo uno dei F/F è ad 1 otteniamo il diagramma di stati (a) di Fig. 11.4bis se invece partiamo con due, tre, quatto o zero 1, otteniamo rispettivamente i diagramma di stati (b), (c), (d), (e). Essi, come si vede, sono quattro cicli diversi senza alcuna possibilità di trasferimento da uno all'altro. Pertanto

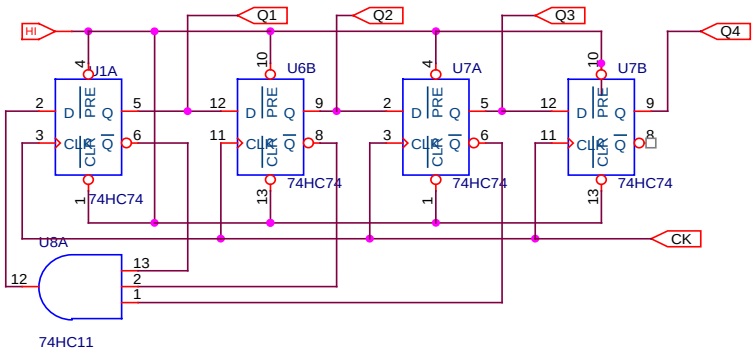


Fig. 11.7 Contatore ad anello modificato

se il ciclo voluto è quello (a) di fig. 11.4bis e se per qualche motivo i contatore ad anello di Fig. 11.4 si trova in uno stato non incluso nel ciclo (a) non ritornerà mai in uno stato della sequenza corretta. Quanto detto giustifica il fatto che il circuito va inizializzato. D'altra parte, se si vuole che il nostro circuito segua, anche a partire da una sequenza errata, una sequenza corretta bisogna modificarlo.

La Fig. 11.7 mostra un contatore ad anello modificato.

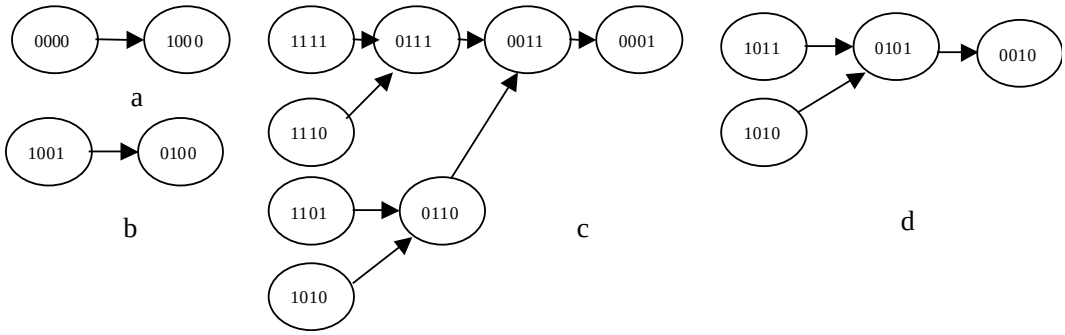


Fig. 11.4 ter

La Fig. 11.4 ter mostra, l'analisi di questo circuito da cui si vede come, a partire da una qualsiasi configurazione iniziale non voluta, si arrivi sempre (dopo un certo numero di passi) ad una delle configurazioni corrette del ciclo desiderato. Infatti, in questo contatore, un bit qualsiasi non corretto sarà semplicemente fatto scorrere attraverso i F/F fino a che tutti saranno azzerati, ad eccezione possibilmente dell'ultimo. Infatti l'ingresso D del primo F/F è la NOR di tutte le uscite dei F/F ad eccezione dell'ultimo. Pertanto esso sarà sempre = 0 fino a che uno di questi F/F (indipendentemente dallo stato dell'ultimo) è ad 1. Poiché il circuito ha questa caratteristica è stato anche eliminato l'ingresso !load che era usato nel caso precedente per imporre un 1 soltanto nel primo F/F. Il timing successivo mostra che, una volta raggiunta la sequenza normale, essa procede come desiderato.

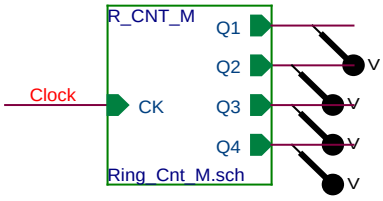


Fig. 11.8

11.3.1 Il Contatore Johnson

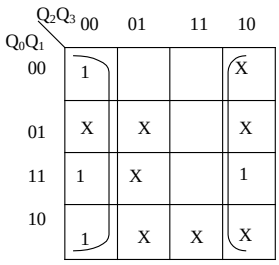
Il contatore Johnson è un contatore che esegue la sequenza descritta dalla tabella di stato, Tab. 11.1. La tabella mostra soltanto la sequenza di stati utilizzati; tutti gli altri sono quindi stati di don't care.

Notare che mentre i F/F Q_1, Q_2, Q_3 seguono la sequenza canonica degli shift register : $Q_1(t+1)=Q_0(t)$; $Q_2(t+1)= Q_1(t)$; $Q_3(t+1)= Q_2(t)$; e lo stato futuro del F/F Q_0 è quello complementare di Q_3. cioè $Q_0(t+1)=Q'_3(t)$;

SP				SF			
Q_0	Q_1	Q_2	Q_3	Q_0	Q_1	Q_2	Q_3
1	0	0	0	1	1	0	0
1	1	0	0	1	1	1	0
1	1	1	0	1	1	1	1
1	1	1	1	0	1	1	1
0	1	1	1	0	0	1	1
0	0	1	1	0	0	0	1
0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0

Tab. 11.1

Malgrado è non necessario, vogliamo dare qui un esempio di minimizzazione delle funzioni Booleane, con il metodo delle mappe, di Fig. 11.10. Notare che in essa figurano tutti gli stati don't care. La derivazione delle altre funzioni Booleane può essere fatta analogamente.



$Q_0(t+1)=Q_3'$

Fig. 11. 10

Una possibile realizzazione di questo contatore con l'uso di F/F JK, in configurazione D, è presentata in Fig. 11.11. Come si vede questo circuito può essere ricavato dal precedente contatore ad anello, con una piccola modifica: il Qbar dell'ultimo stadio è connesso all'ingresso (D equivalente) del

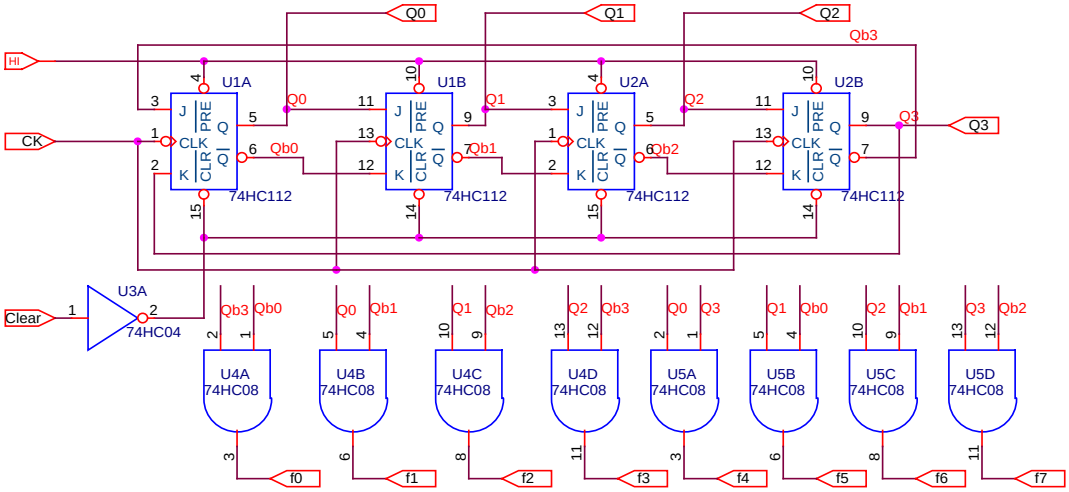


Fig. 11.11

primo.

A causa di questa commutazione tra l'uscita dell'ultimo F/F e l'ingresso del primo, il contatore passa attraverso 2N stati diversi, dove N è il numero di F/F nel contatore. Così, un contatore Johnson a 4 bit ha il doppio di stati di un contatore ad anello, sebbene ne ha ancora la metà di un contatore binario 8421. Il contatore Johnson tuttavia necessita di una decodifica in uscita per ottenere segnali separati da ciascuno stato. La decodifica presente nella stessa figura è ricavabile facilmente dalla tabella di stato (considerata come tabella di verità) e ponendo a destra le otto funzioni di decodifica (mutuamente esclusive). Considerati gli 8 stati di don't care si ricavano le funzioni Booleane delle uscite decodificate.

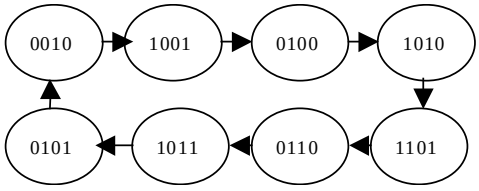


Fig. 11.11bis

Il contatore Johnson, come quello ad anello, può restare intrappolato in una sequenza di stati sbagliati. La Fig. 11.11bis mostra come basta che entri in uno stato sbagliato perché resti in un ciclo non voluto. Ovviamente è necessario che venga modificato in qualche modo il

circuito fondamentale di Fig. 11.11 in modo tale, per es. che non appena viene rivelato uno stato non voluto il circuito venga azzerato (essendo lo stato 0000 uno delle sequenza voluta).

Il timing di Fig. 11.13 è relativo al circuito di Fig. 11.12 che è il blocco gerarchico del circuito di Fig. 11.11. Le tracce presentate sono relative ai probe indicati in Fig. 11.12. In generale, se interessa realizzare un certo numero di funzioni, che indicano, in modo mutuamente esclusivo, le diverse fasi in cui un processo si deve evolvere si usano contatori ad anello o di tipo Johnson perché, con essi, si realizzano le funzioni richieste che sono caratterizzate da completa assenza di false commutazioni. Infatti sia nell'uno che nell'altro caso quando si passa da uno stato al successivo vi è sempre la variazione di un solo bit nello stato del contatore e questo assicura l'assenza di false decodifiche.

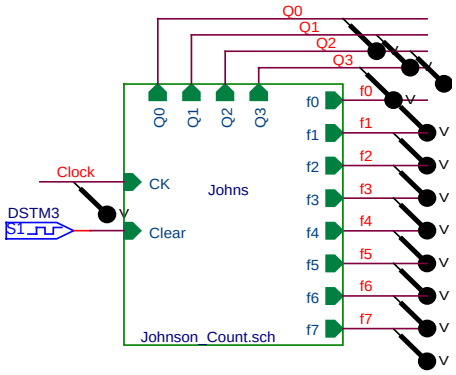


Fig. 11.12

Questo stesso obbiettivo può essere anche raggiunto con contatori che, pur utilizzando tutti gli stati che possono essere codificati con in numero di Flip-flop usati, si evolvono in modo che lo stato binario successivo differisce dal precedente per la variazione di un solo bit (come accade in un contatore che segua la sequenza riflessa Gray già a noi nota).

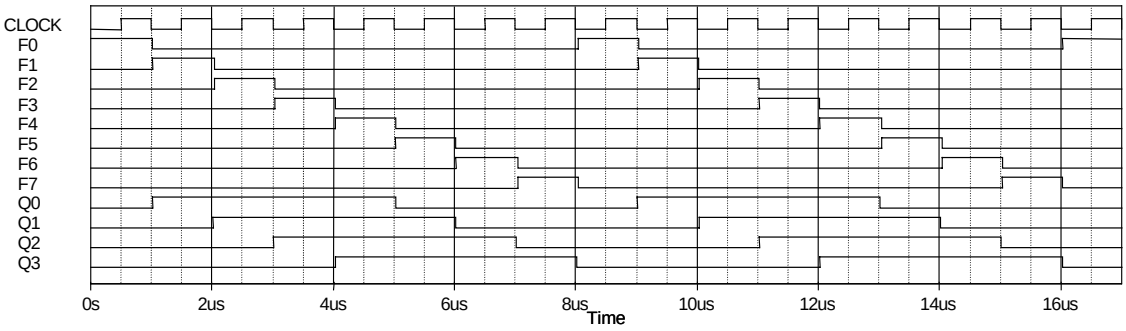


Fig. 11.13

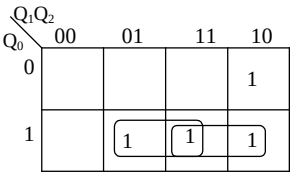
Qui di seguito è presentato il progetto di un contatore a 3 bit che segue la sequenza Gray.

11.4 Contatore Gray con uscite decodificate

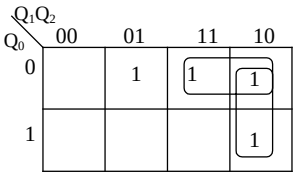
La Tab. 11.2 mostra la sequenza di un contatore gray a 3 bit. Essa è anche una tabella di stato pur mancando della sezione "stato futuro"; poiché essa si trova nella riga successiva. In questo caso abbiano anche messo le 8 funzioni di uscita f_0 –f_7.

Q_0	Q_1	Q_2	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	1	0	0	1	0	0	0	0	0
0	1	0	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	1	0	0	0
1	1	1	0	0	0	0	0	1	0	0
1	0	1	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	1

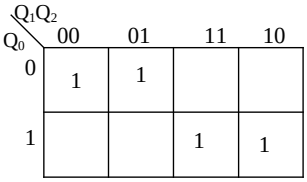
Tab. 11.2



$Q_0(t+1)=$
$Q'_0Q_1Q'_2+Q_0(Q_1+Q_2)$
Fig. 11. 14



$Q_1(t+1)=$
$Q'_1Q'_0Q_2+Q_1(Q'_0+Q'_2)$
Fig. 11. 15



$Q_2(t+1)=$
$Q'_2(Q_0\odot Q_1)+Q_2(Q_0\odot Q_1)$
Fig. 11. 16

Le Fig. 11.14, 11.15 e 11,16 sono le mappe per lo stato futuro dei tre F/F e forniscono le relative funzione di stato minimizzate.

Utilizzando F/F di tipo JK e progettando con l'uso delle funzioni di stato, abbiamo per:

$JQ_0= Q_1Q'_2$; $KQ_0= = Q'_1Q'_2$
$JQ_1= Q'_0Q_2$; $KQ_1= = Q_1Q_2$
$JQ_2= Q_0\odot Q_1$; $KQ_2= (Q_0\odot Q_1)'$

Il circuito che scaturisce dalle funzione di ingresso ai F/F, ora trovate, è presentato in Fig.

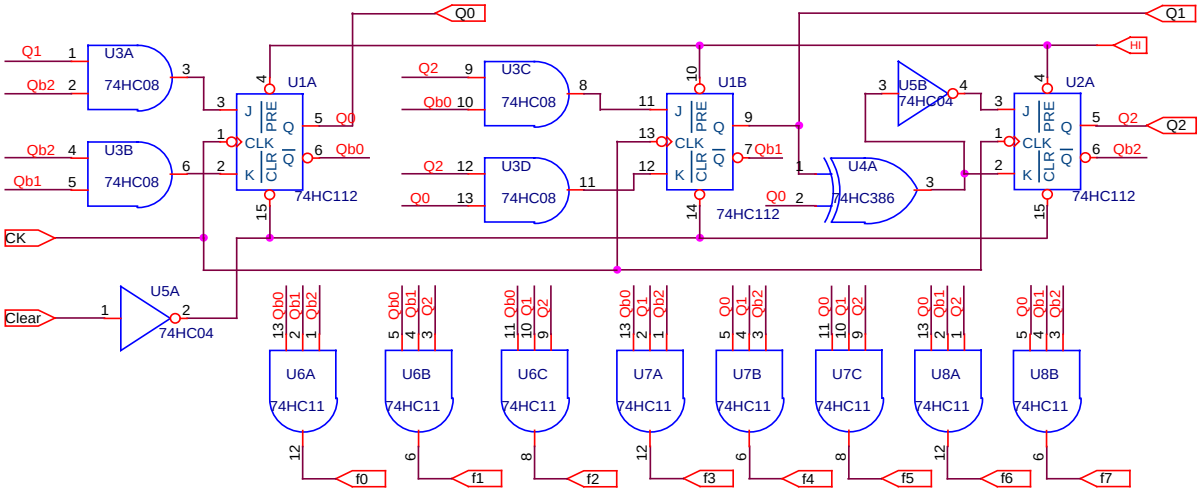


Fig. 11.17

11.17 ed è richiamato dal blocco gerarchico di Fig. 11.18. In questo compaiono tutti i probe utilizzati per ricavate il timing di Fig. 11.19.

Si paragoni ora i due contatori con uscita decodificata Johnson e Gray. Entrambi decodificano una sequenza di 8 stati ma il primo utilizza 4 F/F e 8 porte AND a **due** ingressi mentre il secondo utilizza 3 F/F e 8 porte AND a **tre** ingressi. Si noti infine che la logica combinatoria per costringere i tre F/F a passare nella sequenza degli 8 stati è più complicata e quindi più costosa di quella del contatore Johnson. Questa osservazione può essere considerata come una dimostrazione che in generale quando si diminuisce il numero di Flip-Flop aumenta la complessità della logica combinatoria

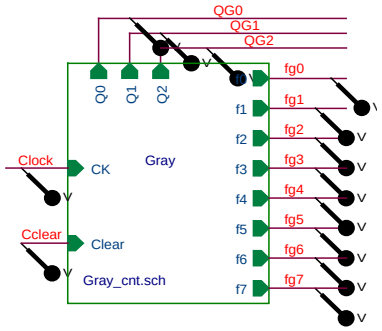


Fig. 11.18

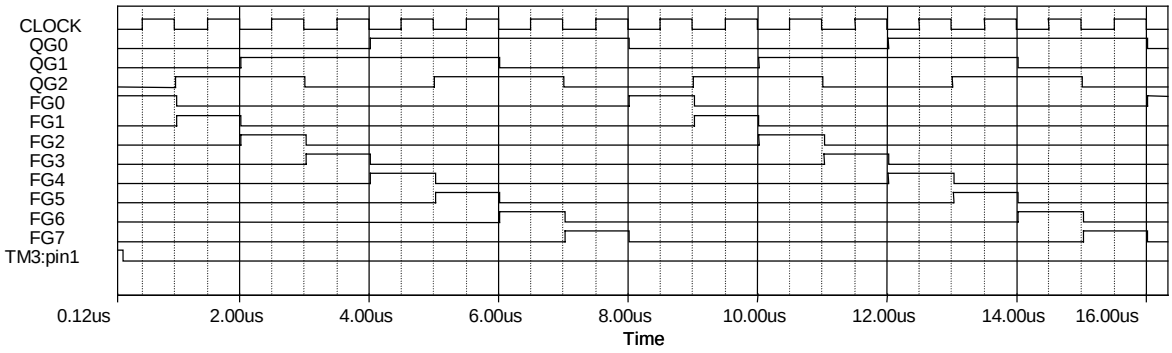


Fig. 11.19

(per ottenere la stessa funzione con un numero di F/F maggiore) e ciò a causa della diminuzione o completa assenza di stati don't care. Si faccia, tuttavia, attenzione che la mancanza di stati don't care assicura la non esistenza di stati non utilizzati e quindi l'impossibilità, per il circuito, di restare intrappolato (per un motivo qualsiasi) in una sequenza di stati sbagliata.

11.5 Contatori Binari e BCD

L'obbiettivo di un circuito contatore è quello di seguire una specifica sequenza di stati o di massimizzare il numero di stati diversi che possono essere ottenuti con un dato numero di F/F. La maggior parte dei contatori contano nella sequenza 8421, 2421, Eccesso di 3, o qualsiasi altro codice binario. I contatori sono usualmente usati come mattoni per costruire altri tipi di circuiti logici. Essi sono usati per: contare, stabilire una sequenza di operazioni di processamento, misurare e dividere frequenza, per effettuare manipolazioni aritmetiche, misurare intervalli di tempo e per molti altri scopi. Esistono diverse varianti di circuiti di conteggio. Tutti sono fatti con F/F JK, T, RS o D e possono essere classificati in due grosse categorie:

Asincroni (chiamati anche contatori **seriali** o **ripple**)

Sincroni (chiamati anche contatori **paralleli**)

Nei contatori sincroni, tutti i F/F cambiano stato simultaneamente; nei contatori asincroni il cambiamento di stato di un F/F può provocare (triggerare) il cambiamento di stato di un secondo F/F, che a sua volta triggera un terzo F/F e così via.

All'interno di ciascuno di queste categorie un contatore può essere progettato per avanzare in uno qualsiasi dei codici binari (fino ad un massimo) per poi ritornare a ripetere la sequenza quante volte si vuole. Il numero di stati consecutivi di una particolare sequenza di conteggio prima che il ciclo si ripeta si chiama **modulo**. I contatori con modulo 2, 4, 8, 16…. o con qualsiasi altra potenza di 2 sono più semplici da costruire; tuttavia è possibile avere contatori con moduli 6 o 10.

I contatori possono anche essere classificati in accordo con il codice pesato con cui essi contano (per es. il contatore binario 8421 o eccesso di 3). Inoltre un contatore può contare in aumento o **UP** o in diminuzione o **DOWN** o in tutte e due i modi, a secondo dei livelli logici che vengono forniti alla logica di controllo.

Una definizione più generale di contatore è la seguente: un circuito che, attraversa M differenti stati in un preciso e specificato ordine, dove M è il modulo del contatore. Un contatore è fatto cambiare da uno stato al successivo da un segnale di clock che proviene da una sua linea di ingresso. Nel seguito descriveremo degli esempi fondamentali sia di contatori sincroni che asincroni. Questi circuiti sono anche disponibili come funzione di uscita di circuiti integrati.

11.5.1 Contatori Binari Asincroni

Un contatore binario, 8421, che conta UP è un contatore che segue la numerazione crescente binaria pura, arriva al suo massimo valore e cicla ritornando al valore di partenza. Per un contatore a 4 bit si ha la tabella di stato di Tab. 1.3. In questa è assente la sezione dello stato futuro in quanto si trova nel rigo successivo. Notare che bisogna considerare il primo rigo come successivo all'ultimo.

Q_3	Q_2	Q_1	Q_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

Tab 11.3

Un contatore binario ripple (asincrono) è il più fondamentale di tutti i circuiti contatori. Una possibile realizzazione è presentata in Fig. 11.20. L'idea, per il suo progetto, può venire osservando la tabella di stato e notando che il primo F/F commuta ad ogni arrivo dell'impulso di clock, il secondo, e tutti i F/F successivi, commutano soltanto quando il F/F che lo precede, nell'ordine, passa dallo stato 1 ad 0. La figura mostra una serie di F/F JK (la cui transizione attiva di clock è quella negativa) tutti in configurazione toggle (J = K=1). Notare che solo il primo F/F ha la sua linea di clock connessa ad un trigger esterno mentre tutti gli altri l'hanno connessa all'uscita Q del F/F che lo precede. Il primo F/F eseguirà una commutazione all'arrivo di ogni impulso di clock, il secondo quando il primo ritorna a zero (dopo essere andato a uno) e così avviene per tutti gli altri F/F. Ciò è coerente con la sequenza di stati in tabella.

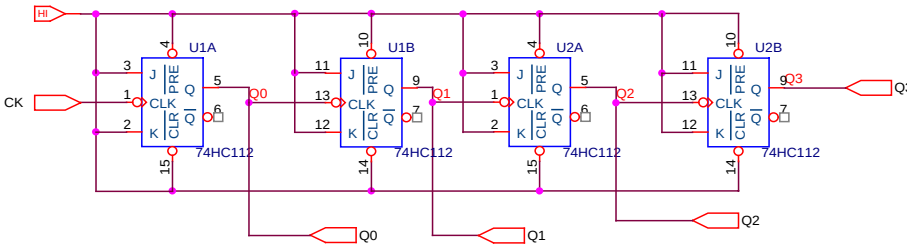


Fig. 11.20

La stessa tabella di stato può essere letta in senso inverso, cioè in modo decrescente (da 15 a 0) o Down. Si noti che, a parte il primo F/F che esegue sempre una variazione di stato per ogni impulso di clock, il secondo e i successivi F/F eseguono una variazione di stato quando il precedente F/F fa una transizione da 0 a 1. Il che suggerisce l'idea, per ottenere un contatore down, di apportare una semplice

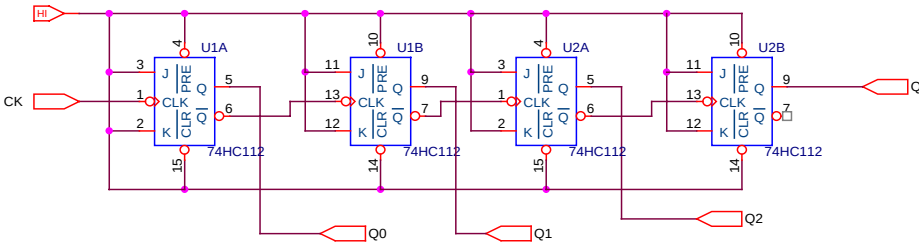


Fig. 11.21

modifica al precedente circuito: connettere l'uscita complementare di ogni F/F con l'ingresso clock del successivo (invece che l'uscita normale). Il circuito che riporta questa variazione è presentato in Fig. 11.21.

I circuiti ora descritti hanno una caratteristica in comune fondamentale che li classifica come contatori asincroni, e che ci permetteranno di identificare tutti i circuiti di questo tipo. L'uscita del primo F/F triggera il secondo F/F, al suo ingresso di clock, il secondo a sua volta triggera il terzo e così via. In questo modo l'effetto di un impulso di clock introdotto all'ingresso del primo F/F si propagherà (ripple) da un F/F ad un altro, fino a che non raggiunge l'ultimo della serie (da cui il nome di **Ripple Counter** o **Contatore Seriale**).

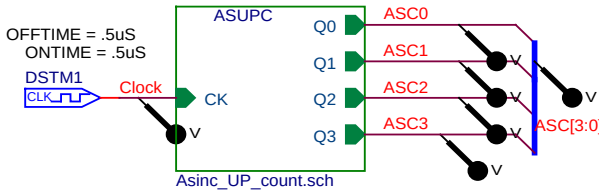


Fig. 11.22

Il timing di Fig. 11.23 è stato ottenuto usando la configurazione UP di Fig. 11.21. Il suo

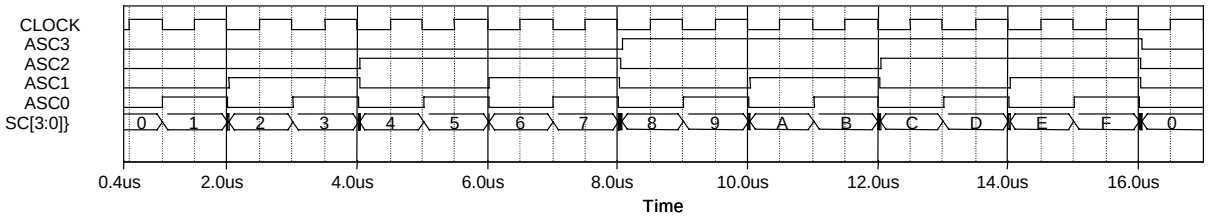


Fig. 11.23

blocco gerarchico (ASUPC) ed i probe con cui sono state prese le tracce è rappresentato in Fig. 11.22. Notare che, siccome l'uscita di ciascun F/F è connessa con l'ingresso clock del successivo F/F, questo cambia stato con una frequenza metà del F/F precedente (divisione di frequenza).

Il timing di Fig. 11.25 si riferisce invece al contatore DOWN asincrono. Esso è stato ottenuto utilizzando il blocco gerarchico di Fig. 11.24 che punta al circuito di Fig. 11.21. Ovviamente presenta le stesse caratteristiche essenziali del contatore Up.

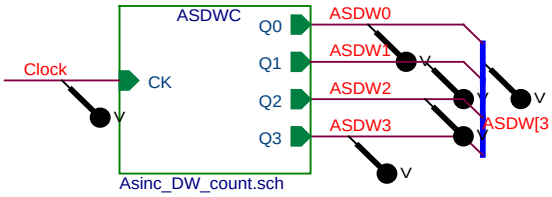


Fig. 11.24

Una importante caratteristica di qualsiasi contatore è la velocità massima a cui può operare.

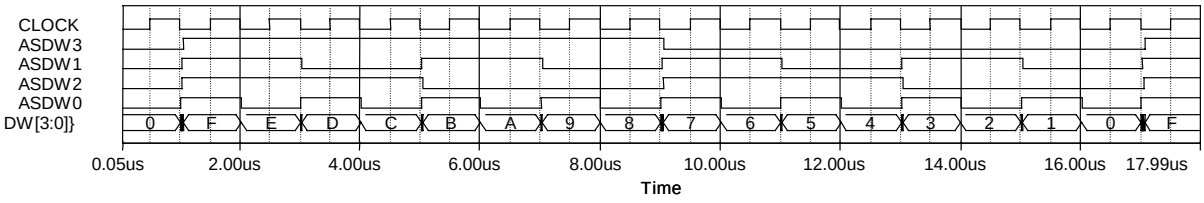


Fig. 11.25

Se ciascun F/F rappresentato in Fig. 11.20 o 11.21 ha un ritardo, dovuto al tempo di propagazione, di 25 ns, il ritardo totale, dall'istante d'inizio del primo fronte attivo dell'impulso di clock a quando l'ultimo F/F ha completata la sua variazione di stato, è di 100 ns. Quindi, il successivo impulso di clock, normalmente, non può avvenire prima di 100 ns. Durante questo tempo i F/F stanno cambiando stato e l'uscita del contatore non è corretta. Questo può essere visto espandendo il timing attorno a quei punti di transizione nella traccia che legge lo stato del contatore come un numero, in cui non ci sono transizioni nette. Ad ogni impulso di clock non tutti i F/F debbono cambiare stato ma quando il conteggio passa da 7 a 8 o quando ricicla da 15 a 0 tutti i F/F debbono cambiare stato ed il segnale di clock deve essere sufficiente lento da permettere il completamento di queste variazioni.
La limitazione della massima frequenza degli impulsi di clock è il principale svantaggio dei contatori asincroni. Nell'esempio precedente, un ritardo di 100 ns tra due successive transizioni attive del clock comporta una frequenza massima di clock pari a 10 MHz (l'inverso di 100 ns)

11.5.2 Contatori Binari Asincroni Up/Down

I due circuiti precedenti possono essere combinati in un solo circuito che può contare UP o Down. Il risultato di questa combinazione produce il circuito presentato in Fig. 11.26. Come si vede tra un F/F e il successivo è stata interposta una logica combinatoria aggiuntiva che realizza, per l'ingresso clock di ogni F/F, il mutiplexing di due funzioni. La prima, quando (l'ingresso di controllo del MUX) U/D è alto seleziona le uscite Q dei F/F realizzando così un contatore UP, la seconda, U/D = 0, seleziona le uscite complementari realizzando un contatore down. In questo circuito è stata inoltre inclusa una linea di abilitazione, CE.

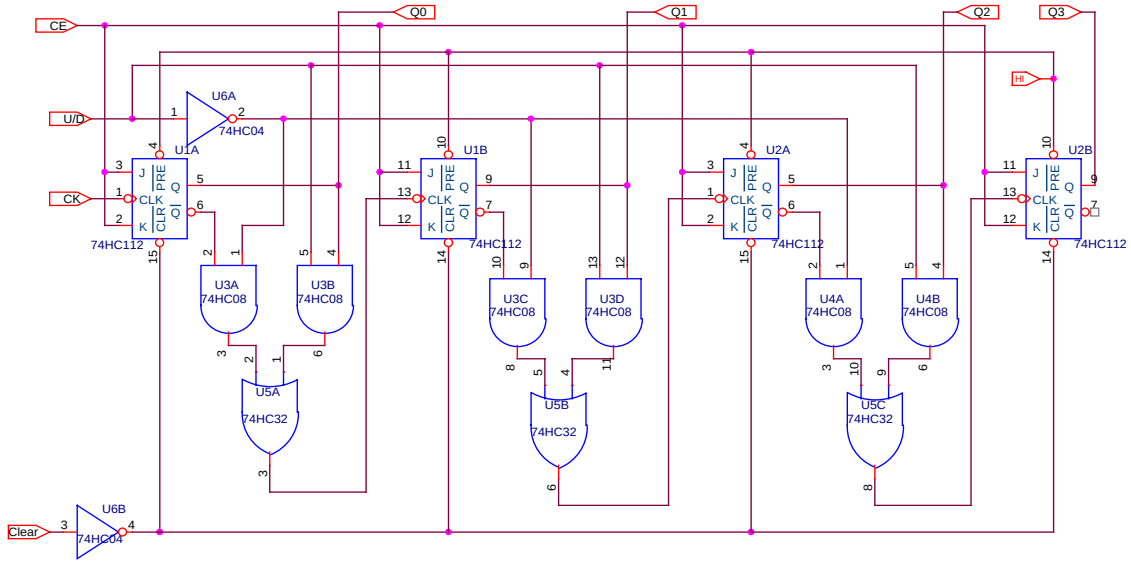


Fig. 11.26

Tale linea di abilitazione agisce sugli ingressi J e K e può essere commutata da 0 ad 1 invece che connetterla stabilmente ad 1 logico. Quando arriva l'impulso di clock, se il livello è a zero nessun F/F cambierà stato. Un contatore disabilitato rimane nel conteggio a cui si trova al momento della disabilitazione.

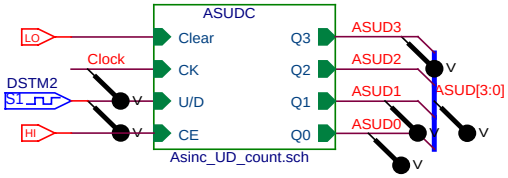


Fig. 11.27

Il timing di Fig. 11.28 è stato ottenuto utilizzando il blocco gerarchico di fig. 11.27 che punta al circuito di Fig. 11.26. Esso mostra come la sequenza di conteggio è controllata dalla linea U/D. Nella prima metà del timing il livello è alto ed il conteggio è crescente, nella seconda parte, U/D=0, ed il conteggio è decrescente.

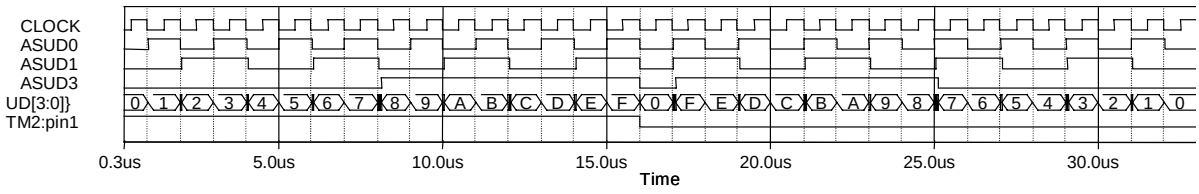


Fig. 1.28

11.6 Contatori Binari Sincroni

In un Contatore Binario sincrono tutte le uscite debbono variare contemporaneamente all'arrivo dell'impulso di conteggio. Un contatore di questo tipo può essere progettato con i metodi già visti. Ovviamente anche adesso questo circuito non ha variabili di ingresso esterne né di uscita, infatti l'uscita di un contatore è lo stato in cui esso si trova. I contatori sincroni, come quelli asincroni, sono costruiti usando F/F JK (o T) con la differenza che tutti i F/F sono triggerati da un segnale di clock comune e, quindi, tutti cambiano stato nello stesso tempo. Supponiamo di dover progettare un circuito contatore up binario a 4 bit. utilizzando F/F T.

SP				Ingressi F/F			
A	B	C	D	TA	TB	TC	TD
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	1
0	0	1	0	0	0	0	1
0	0	1	1	0	1	1	1
0	1	0	0	0	0	0	1
0	1	0	1	0	0	1	1
0	1	1	0	0	0	0	1
0	1	1	1	1	1	1	1
1	0	0	0	0	0	0	1
1	0	0	1	0	0	1	1
1	0	1	0	0	0	0	1
1	0	1	1	0	1	1	1
1	1	0	0	0	0	0	1
1	1	0	1	0	0	1	1
1	1	1	0	0	0	0	1
1	1	1	1	1	1	1	1

Tab. 11.4

La Tab. 11.4 mostra la tabella di eccitazione. In questa manca la sezione stato futuro in quanto questa può essere derivata dalla riga successiva della sezione dello stato presente. In Fig. 11.29 sono mostrate le mappe delle singole funzioni di ingresso ai F/F con le relative funzioni semplificate.

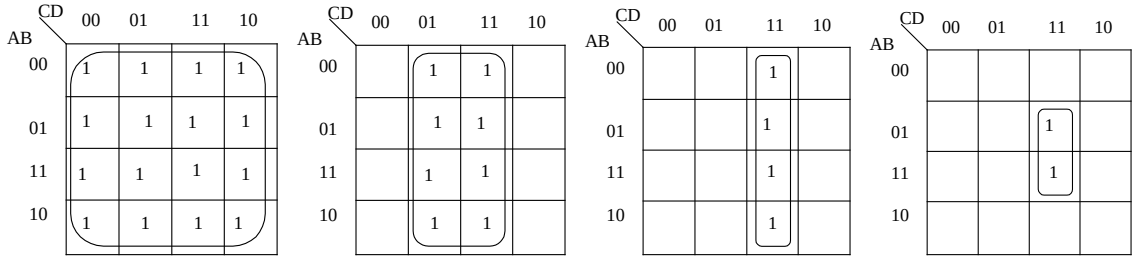


Fig. 11.29

Notare che, mentre la funzione di ingresso del primo F/F è una costante = 1 logico, quella di tutti gli altri F/F è uguale al prodotto logico dello stato di tutti i F/F che lo precedono.

Questa osservazione ci fornisce un'indicazione di cosa fare per aumentare la capacità di conteggio di questo tipo di contatore, aggiungendo degli altri F/F. La Fig. 11.30 mostra la realizzazione dettata dalla prescrizione trovata. Gli ingressi J e K di tutti i F/F sono connessi alle uscite Q di tutti i F/F, che lo precedono nella catena, per mezzo di porte AND. In questo modo ogni F/F cambierà stato quando tutti i F/F che lo precedono sono tutti ad 1.

Una lettura della tabella di eccitazione in direzione opposta rispetto alla precedente cioè in senso decrescente di conteggio, invece che crescente, ci fornisce la conclusione che un F/F cambia stato quando lo stato dei F/F che lo precedono sono tutti a 0.

Ciò suggerisce l'idea che usando gli stessi componenti e connettendo, agli ingressi delle porte AND, le uscite complementate, Q', invece che le uscite Q di tutti i F/F che lo precedono, la

sequenza di conteggio risulta invertita (DOWN). Notare che poiché tutti i F/F ricevono lo stesso clock e quindi cambiano stato contemporaneamente, il ritardo totale è indipendente dal numero di F/F che costituiscono la catena di conteggio ed è pari a quello di un solo F/F.

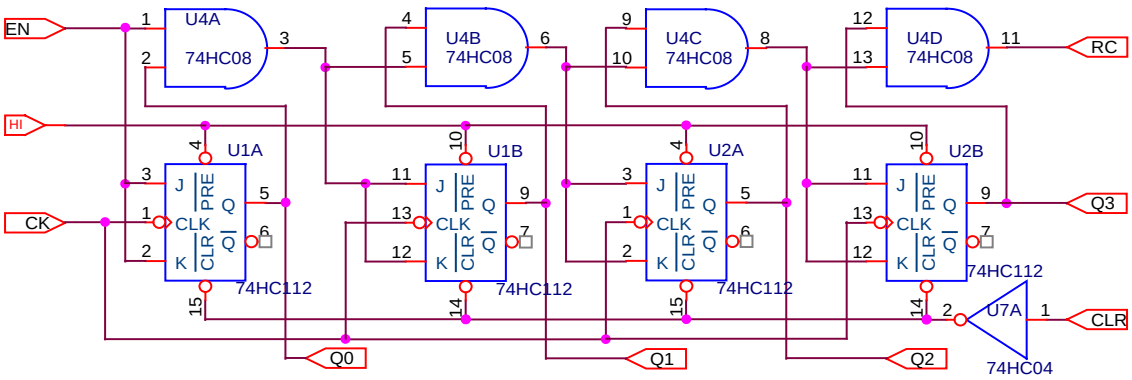


Fig. 11.30

In questo modo quando il conteggio cambia da uno stato al successivo non ci sono stati intermedi durante i quali le uscite non sono corrette.

Però anche i contatori sincroni hanno delle limitazioni.

1) Essi richiedono, per la loro realizzazione, una logica combinatoria più complicata e quindi più costosa.
2) Notare che l'ultima porta AND, U5A, di Fig. 11.30 ha quattro linee di ingresso (ne avrebbe un'altra in più, se avessimo usato una ulteriore linea di abilitazione di conteggio). Per evitare ciò, nel nostro circuito, abbiamo usato la porta AND, U4B, per bloccare il clock e disabilitare il conteggio e abbiamo chiamato la linea di controllo En (Enable). Se il contatore dovesse essere espanso ulteriormente, il numero di ingressi delle porte successive aumenterebbe portando ad una limitazione pratica. Per ovviare a tale inconveniente è stata usata una porta AND, U6A, la cui uscita diventa 1 quando tutti le uscite Q sono ad uno (il conteggio ha il suo massimo valore). Questa uscita è chiama **Ripple Count** ed è usata, assieme all'ingresso di abilitazione, per costruire contatori con maggiore capacità di conteggio. Nel caso di contatore di tipo Down la stessa funzione può essere ottenuta facendo l'AND di tutte le uscite complementari dei F/F (il conteggio ha il suo minimo valore).

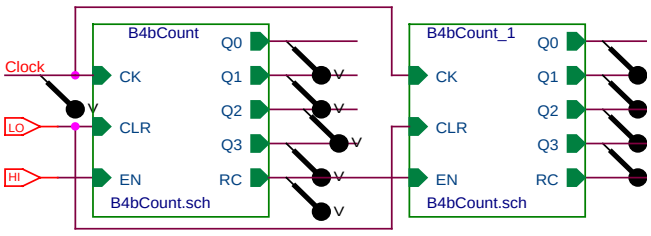


Fig. 11.31

Le connessioni necessarie, per aumentare la capacità di conteggio di un contatore (usando più esemplari dello stesso tipo), sono mostrate in Fig. 11.31. I blocchi gerarchici, ivi rappresentati, puntano al circuito rappresentato in Fig. 11.30; il primo contatore è sempre abilitato mentre il secondo viene abilitato quando l'uscita del primo, RC è al livello 1. Poiché ciò accade una sola volta per ciclo, il secondo contatore conta il numero di cicli del primo. In altri termini ogni conteggio del secondo contatore vale 16. Il circuito risultante è ancora sincrono in quanto il secondo contatore è già abilitato quando il conteggio del primo è 15 e quindi non provoca ritardi di commutazione. Il funzionamento è illustrato dal timing di Fig.

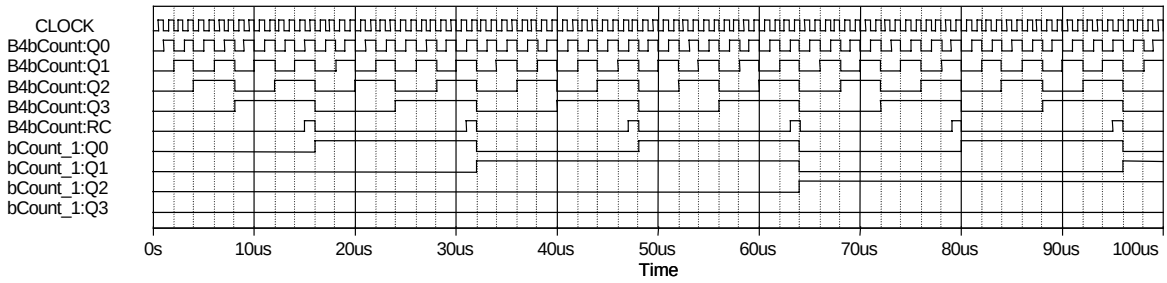


Fig. 11.32

11.32.

Nel caso si desideri una capacità di conteggio ancora maggiore basta connettere un terzo contatore al secondo con le stesse modalità con cui il secondo è connesso al primo.

11.7 Contatori BCD Sincroni

Un contatore BCD è un contatore che conta fino a dieci (i dieci stati vanno da 0 a 9). Dopo che ha raggiunto 9 ritorna a 0 e riparte un altro ciclo. Questo tipo di contatore deve avere almeno 4 F/F per rappresentare una cifra decimale, poiché questa è rappresentata con almeno 4 bit. La sequenza di stati di un contatore decimale è dettata dal codice binario usato per rappresentare la cifra decimale.

SP				Ingressi F/F			
Q_3	Q_2	Q_1	Q_0	T_3	T_2	T_1	T_0
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	1
0	0	1	0	0	0	0	1
0	0	1	1	0	1	1	1
0	1	0	0	0	0	0	1
0	1	0	1	0	0	1	1
0	1	1	0	0	0	0	1
0	1	1	1	1	1	1	1
1	0	0	0	0	0	0	1
1	0	0	1	1	0	0	1

Tab. 11.5

Esso è molto simile a quella di un contatore binario ad eccezione che dopo lo stato 1001 (9 decimale) si passa a 0000 (0 decimale) Anche se il circuito è simile a quello già studiato, esso assume una importanza particolare perché è comunemente usato sia nei computer sia nelle calcolatrici sia in tutti gli altri circuiti in cui è necessario un conteggio decimale. Un contatore decimale, sia esso sincrono sia asincrono conta nel codice binario 8421. Quindi è fondamentalmente lo stesso di qualsiasi altro contattore binario a 4 bit ad eccezione che ha un circuito speciale che

rivela il conteggio limite (9 binario o $(1001)_2$) e limita così il massimo conteggio. Esso è un contatore modulo 10.

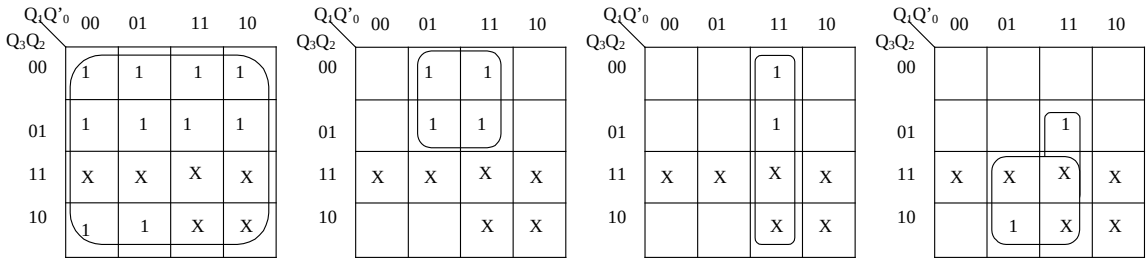


Fig. 11.33

Supponiamo di dover progettare un circuito utilizzando F/F di tipo T. La Tab. 11.5 mostra anche la tabella di eccitazione. Come in precedenza manca la sezione stato futuro in quanto questa può essere derivata dalla riga successiva della sezione dello stato presente. In Fig.

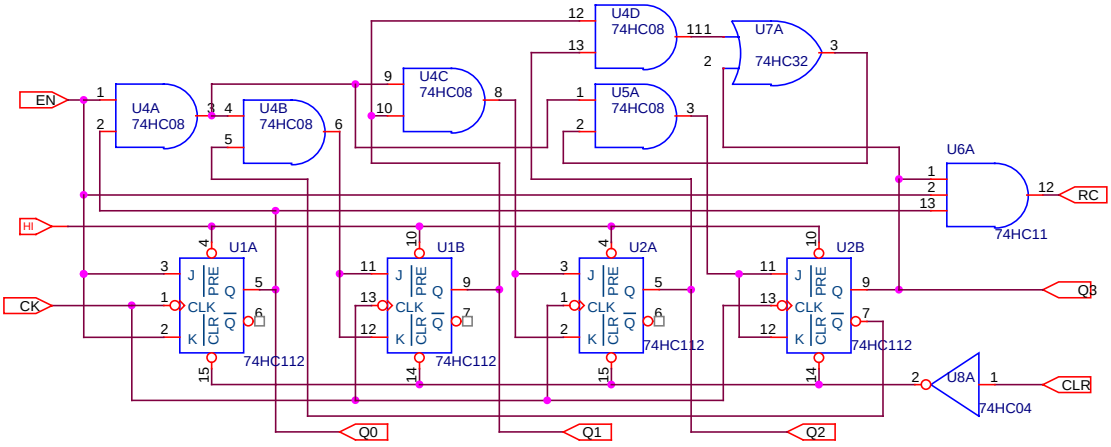


Fig. 11.34

11.33 sono mostrate le mappe delle singole funzioni di ingresso ai F/F con le relative funzioni semplificate. I sei stati, non compresi nel codice BCD, sono stati considerati di don't care: il contatore non può andare mai in uno di questi stati.

Malgrado ciò notare, dallo schema logico di Fig. 11.34, quanto questo circuito è più complicato del precedente.

Rispetto alle equazioni trovate, nel circuito di Fig.11.34, troviamo la porta AND, U6A, in più. La sua uscita diventa 1 quando il conteggio in binario è 1001. Esso rivela che il massimo conteggio del contatore ed è stato aggiunto ed indicato, come per il contatore binari, RC, ed usato nel stesso modo.

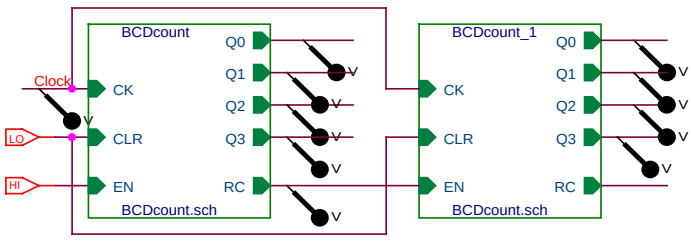


Fig. 11.35

La Fig. 11.35 mostra due blocchi gerarchici (entrambi si riferiscono al contatore BCD di Fig. 11.34) che connettono in cascata due contatori BCD per realizzare una capacità di conteggio che va da 0 a 99. Come è evidente i collegamenti tra i due contatori sono gli stessi di quelli illustrati precedentemente. Qui basta dire che ogni conteggio del secondo contatore vale 10. L'uso di altri contatori fornirebbe la possibilità di contare le centinaia, le migliaia di conteggi e così via.

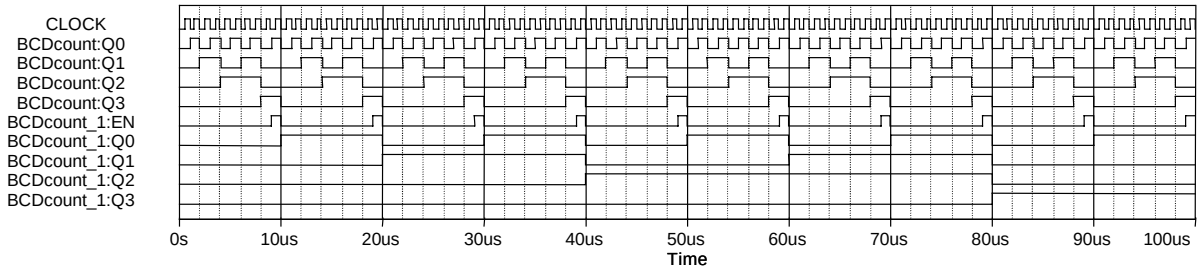


Fig.. 11.36

Il timing di Fig. 11.36 mostra il funzionamento dei due contatori sopra descritti.

11.8 Contatori Modulo N

Un contatore modulo N è un contatore che ha N stati differenti. Un contatore modulo N può essere sia di tipo sincrono o asincrono purché abbia dei circuiti che controllano il numero totali di stati che esso può avere. Un contatore BCD, per esempio, è un contatore che potrebbe contare da 0 a 15 ma che ha dei circuiti speciali che limitano il massimo conteggio a 9. Se dotiamo il contatore binario a 4 bit, di un circuito che decodifica lo stato 1010, e lo stesso è usato per azzerare il contatore (attraverso la linea di clear asincrona) allora i suoi stati andranno da 0 a 9. Ovviamente cambiando il circuito di decodifica potremmo attenere un numero di stati qualsiasi (< 16).

La definizione ora data è molto generale e contiene tutti i contatori con moduli di conteggio grande o piccolo. In pratica un mezzo per ottenere un contatore con modulo qualsiasi è quello di interconnettere molti contatori in cascata. In questo tipo di connessione, come già da noi sperimentato, i moduli dei contatori separati vengono moltiplicati e determinano il modulo desiderato. Per esempio un contatore modulo 105 può essere ottenuto connettendo in cascata tre contatori di modulo rispettivamente 3, 5 e 7 poiché 3 x 5 x 7 =105.

Contatori modulo 3, 5, 7, 11 possono essere ottenuti con le stesse tecniche di progetto usate per i contatori modulo 16 e 10.

11.9 Contatori Programmabili

Un contatore programmabile è qualsiasi contatore il cui modulo o figura di conteggio può essere modificato in qualche modo usando segnali di controllo anziché apportando delle modifiche circuitali.
Il modo più comune è quello di usare segnali di programmazione che presettano il contatore a certi valori di conteggio (cambiando così il modulo del contatore) o di controllare il contatore in modo che esso si fermi quando viene raggiunto un certo conteggio e/o si resetti al valore prestabilito e lo faccia ripartire.

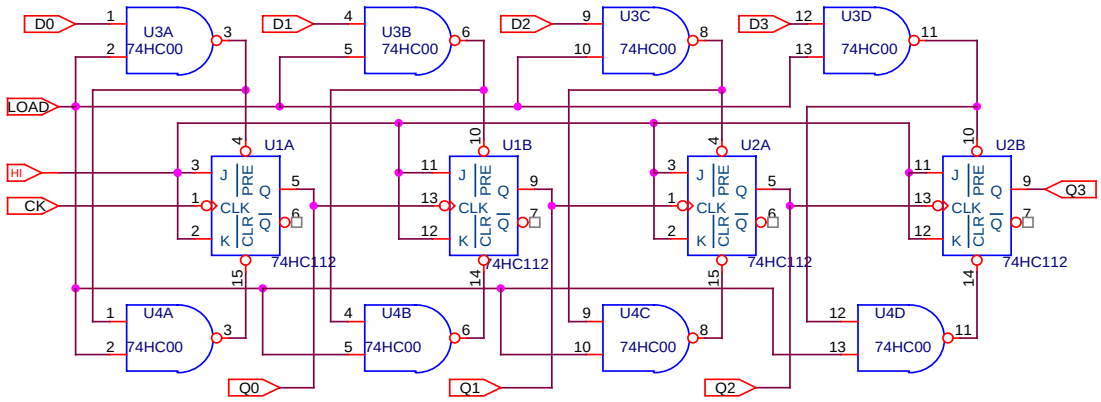


Fig. 11.37

Un circuito di conteggio può essere programmato e controllato in molteplici modi, noi ne discuteremo alcuni che mostrano le idee fondamentali per la programmazione del conteggio. La Fig 11.37 mostra un contatore asincrono a 4 bit che può essere presettato ad un qualsiasi valore tra 0 e 15 attraverso le porte AND U3(A-D) abilitando la linea di controllo LOAD.

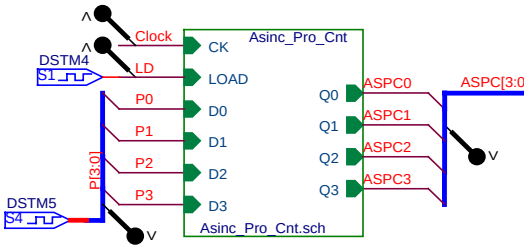


Fig. 11.38

Il timing di Fig. 11.39 è stato ottenuto utilizzando il circuito ora descritto, richiamato dal blocco gerarchico “Asin_Pro_cnt” di Fig. 11.38. Inizialmente il contatore viene presettato a 5 attraverso il segnale “Load”=1. In questo modo il contatore attraversa la sequenza 6, 7,..:,15.

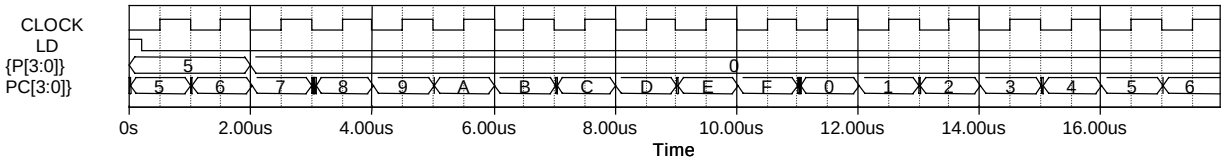


Fig. 11.39

Quando il conteggio è arrivato a 15. poiché, al contatore, non arriva il segnale di Load, riparte da zero. Dal timing si vede chiaramente che il Load è fatto solo all'inizio.
La Fig. 11.40 mostra un contatore sincrono a 4 bit che avanza da 0 ad un valore massimo e che raggiuntolo si ferma. Se resettato riparte a fare la stessa sequenza.

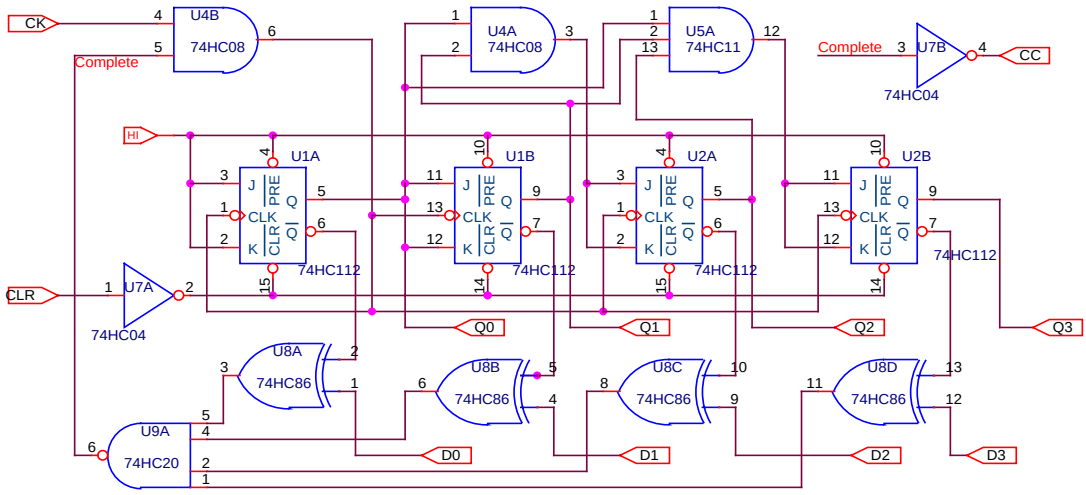


Fig. 11.40

È facile verificare che questo circuito è fondamentalmente lo stesso di quello usato per il contatore sincrono a 4 bit, modificato dall'aggiunta di un poco di logica di controllo. Infatti, il modulo è scelto attraverso le porte XOR U8(A-D) e la NAND U9A. L'uscita di ogni una delle porte U8(A-D) è normalmente a zero e diventa 1 quando il Q del F/F, a cui è collegato uno dei suoi ingressi, uguaglia l'ingresso di programmazione relativo, (D0-D3). La porta U9A è sempre ad 1 logico fino a che tutti i suoi ingressi non sono diventati 1 (uguaglianza tra il contenuto del contatore e il dato di programmazione). Quando l'uscita di U9A diventa 0 il clock viene disabilitato e il conteggio si blocca. A questo punto il segnale Count-Complete (CC) è attivo e può esser usato per resettare il contatore o per innescare qualche altra azione.

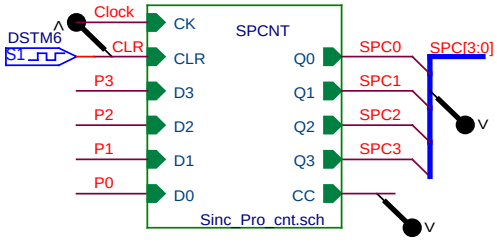


Fig. 11.41

Per provare il funzionamento di questo circuito sono stati usati due modi.

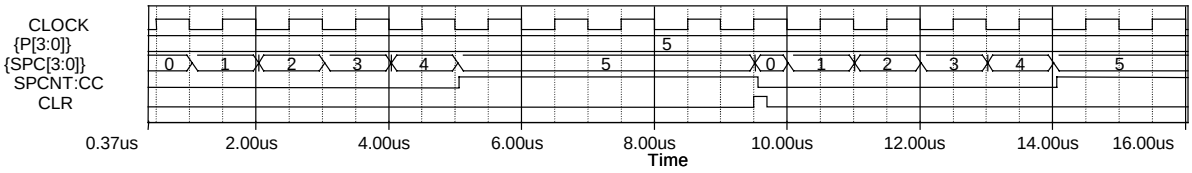


Fig. 11.42

1) Il primo modo usa il blocco gerarchico SPCNT (Fig. 11.41), che punta al circuito di Fig. 11.40. Il timing ottenuto è quello di Fig. 11.42. Il contatore parte da zero, arriva a 5 e si ferma. Un successivo impulso di clear esterno, asincrono, lo azzera e quindi causa l'inizio di un nuovo ciclo. La durata dello stato 0 dipende dal tempo che intercorre tra l'istante di azzeramento e quello d'arrivo del primo fronte attivo del clock.

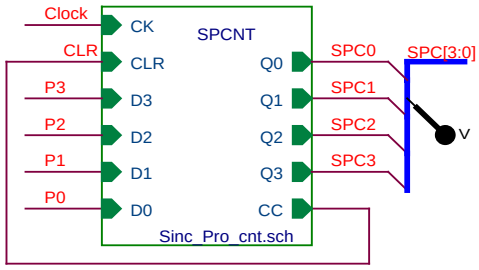


Fig. 11.43

2) Il secondo modo usa il blocco gerarchico SPCNT (Fig. 11.43) che, come il precedente, punta al circuito di Fig. 11.40. Questa volta, come è evidente dalla stessa Fig. 11.43 il segnale di clear è sincrono (essendo connesso con l'uscita "Count-Complete" (CC)).

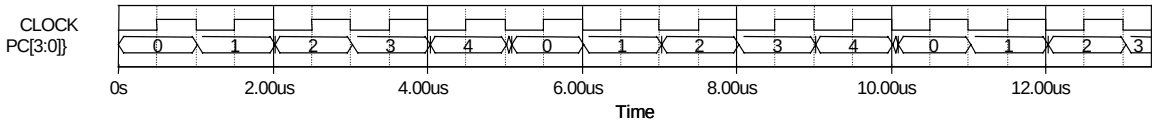


Fig. 11.44

Pertanto ci dobbiamo aspettare che, non appena il conteggio arriva al valore programmato, immediatamente il contatore riparte da zero. Questo è evidente dalla Fig. 11.44 che ne è il timing. Come si vede adesso il contatore cicla tra gli stati 0 e 4 in quanto l'azzeramento del conteggio avviene praticamente nell'istante in cui viene rivelato 5 e la durata di questo stato è pari al tempo di propagazione di una porta logica. Si vede chiaramente che c'è un brevissimo istante in cui il conteggio è pari a 5 (in coincidenza del non netto passaggio da 4 a 0 nella traccia PC[3:0].

11.10 Somma Binaria Seriale

In questo paragrafo e nel successivo, dopo avere introdotto tutte le funzioni sequenziali necessarie, riprendiamo i circuiti aritmetici seriali che erano stati qui rimandati alla fine del Cap. 7. Si ricorda che un breve cenno alla soluzione del problema è stato dato nel Par. 10.4, Es. 10.4 in cui era stato affrontato solo una soluzione parziale del problema. Adesso vogliamo invece fornire una soluzione completa e realizzare circuiti che possano funzionare da soli.

Per realizzare un sommatore seriale (vedi Fig. 11.45) si utilizza un solo circuito full adder, un F/F D (memorizzerà il carry), una coppia di registri a scorrimento del tipo Parallel-In/Serial-Out (PISO) per caricare e trasferire serialmente la coppia di bit da sommare, un registro Serial-In/Parallel-Out (SIPO) (a cui sarà trasferito il risultato), uno Storage Register utilizzato per trasferire il risultato (memoria esterna di uscita) ed un contatore down (per contare il numero di coppie di bit ancora da sommare).
Il full adder somma sequenzialmente ciascuna coppia di bit provenienti dall'uscita dei due PISO ed il bit di carry contenuto nel F/F D, a partire dalla coppia di bit meno significativa. Con il primo impulso di clock avviene

a) Il contatore down viene presettato ad un valore binario pari al numero di coppie di bit da sommare.
b) il caricamento parallelo dei due PISO, con i due numeri da sommare prelevati da una memoria esterna di ingresso (non presentata). Alle loro uscite sarà presente la coppia di bit meno significativa che costituisce anche l'ingresso del full adder.
c) viene azzerato il F/F D, la sua uscita costituisce l'ingresso carry del full adder.

Alla fine del primo impulso di clock (passato un tempo pari a quello di propagazione del full adder), la somma della prima coppia di bit ed il nuovo carry sono pronti.
Utilizzando un secondo impulso di clock vengono effettuate le seguenti operazioni in parallelo:

1) il bit di somma viene fatto scorrere dal full adder al SIPO,
2) il bit di carry viene fatto scorrere dal full adder al F/F D. La sua uscita, collegata all'ingresso carry del full adder, costituisce il bit di carry per la somma successiva,
3) il contatore di bit da sommare viene decrementato,
4) la nuova coppia di bit, immediatamente più significativa, viene fatta scorrere sui F/F di uscita dei PISO e quindi presentata all'ingresso del full adder.

L'uscita del full adder, passato il tempo di propagazione, contiene la somma della seconda coppia di bit e del carry. Al terzo impulso di clock vengono ripetute le operazioni 1)-4) ottenendo la memorizzazione della seconda somma e del carry. La sequenza di operazioni ora descritta viene ripetuta con i successivi impulsi di clock fino a che il contatore non ha

raggiunto il valore zero. A questo punto la somma dei due addendi è stata effettuata e un ulteriore impulso di clock è utilizzato per la memorizzazione della somma nella memoria esterna di uscita.

Una nuova somma può essere iniziata nel seguente modo: con il primo impulso di clock avviene la sequenza di operazioni a)-c) e quindi, con gli altri impulsi di clock, ciclicamente le operazioni 1)-4) seguite da una nuova memorizzazione del risultato in una nuova memoria esterna.

La Fig. 11.45 mostra lo schema logico con il quale è stato simulato il sommatore seriale. Notare che, poiché sia i contatori sia gli shift register hanno la stessa transizione attiva del clock (positiva) il clock per lo shift register è stato invertito. Ciò consente il caricamento del nuovo dato al centro dell'intervallo in cui il contatore ha il bit di peso 8 alto; questo evita problemi di set-up time per lo shift register. Il presetting del contatore è ad 8 dovendo sommare dati ad 8 bit. Un altro punto degno di nota è che, prima di azzerare il F/F carry e il

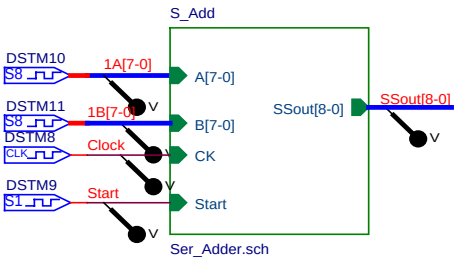


Fig. 11.46

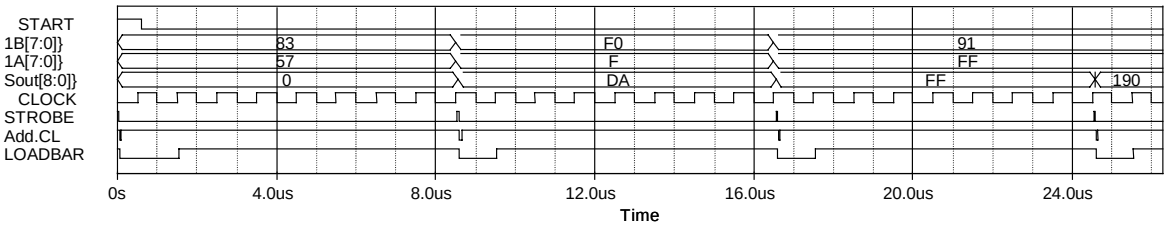


Fig. 11.47

SIPO, e necessario trasferire il dato somma nella memoria esterna (U8, U9) e quindi caricare una nuova coppia di dati da sommare. Ciò è stato realizzato utilizzando un rivelatore di fronti positivi in modo tale che il clear avvenga certamente dopo che il dato è stato caricato e prima della caricamento dei nuovi addendi. Il circuito è stato provato usando il blocco gerarchico di Fig. 11.46 che punta al circuito di Fig. 11.45. Il timing ottenuto (vedi Fig. 11.47) mostra il funzionamento del sommatore utilizzando i dati prelevati dai punti in cui sono mostrati i probe in fig. 11.45 e Fig. 11.46. Notare infine che per rendere più evidente il valore della somma il bit di carry è stato salvato in un ulteriore FF D la cui uscita va a far parte del risultato, come bit più significativo. Il risultato è pronto dopo che sono passati 8 impulsi di clock ed è stabile in memoria tra una somma e la successiva.

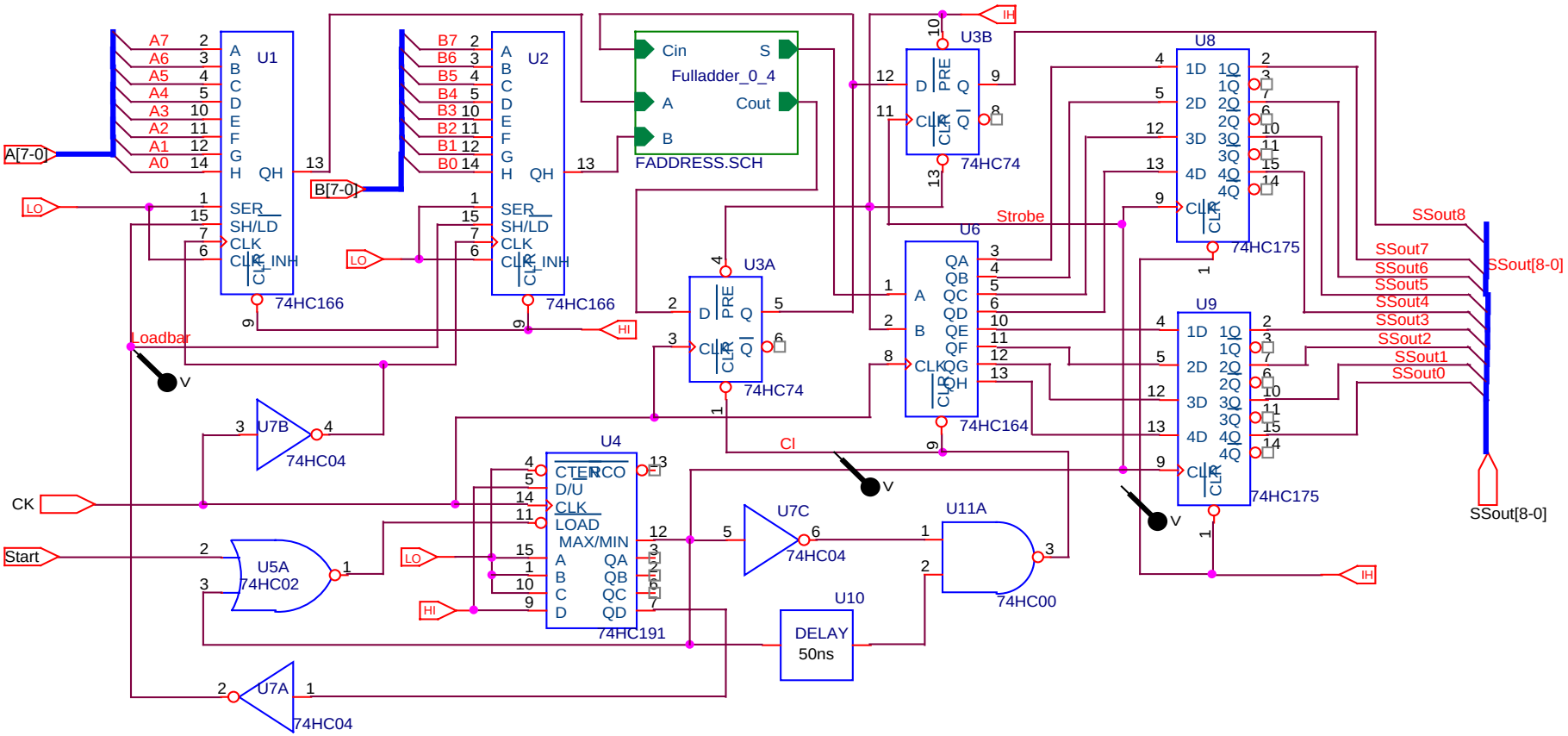

A[7-0]
B[7-0]
U1
U2
74HC166
FADDRESS.SCH
Cin
S
Fulladder_0_4
A
Cout
B
U3B
74HC74
U3A
74HC74
U6
74HC164
U8
74HC175
U9
74HC175
Strobe
SSout8
SSout7
SSout6
SSout5
SSout4
SSout3
SSout2
SSout1
SSout0
SSout[8-0]
Loadbar
U7B
74HC04
U4
74HC191
CI
U7C
74HC04
U11A
74HC00
U10
DELAY
50ns
CK
Start
U5A
74HC02
U7A
74HC04
LO
HI
IH

Fig. 11.45

11.11 Somma Seriale di una sequenza di Numeri Binarie

Al circuito descritto nel paragrafo precedente, che risulta essere fondamentale per questo tipo di operazioni, possono essere apportate molte varianti allo scopo di ottenere risultati di tipo differente.

Per es. Il circuito di Fig. 11.48 realizza una macchina che somma una sequenza di numeri. Allo scopo di utilizzare quanto più possibile gli integrati già usati per il circuito precedente, nel progetto di questo circuito si è supposto che il numero di bit, di cui sono formati gli addendi, sono 4 e che il risultato necessita al più di 8 bit (oltre eventualmente il bit di carry). In questo modo il numero di shift necessari per effettuare la somma resta pari ad 8. I PISO, da noi usati, hanno anche un ingresso seriale. Uno di essi può essere usato sia come registro PISO sia come Shift Register con ingresso seriale. Questa funzione è richiesta perché, nel nostro caso, è necessario conservare il risultato della somma parziale in modo seriale (bit a bit). Per il fatto che questo registro è destinato ad assolvere questa funzione prende il nome di ***accumulatore***. Il SIPO è ancora usato per raccogliere il risultato seriale della somma. Poiché bisogna tener conto del numero di addendi da sommare si usa un secondo contatore (inizialmente presettato al numero di addendi). All'inizio della serie di somme l'accumulatore e il PISO vengono caricati con la prima coppia di numeri. La somma procede come indicato precedentemente con la sola

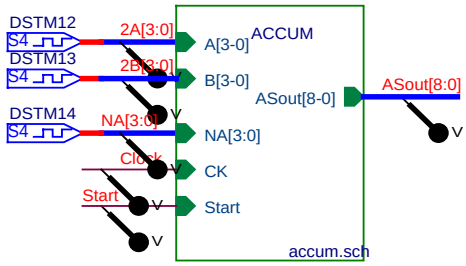


Fig. 11.49

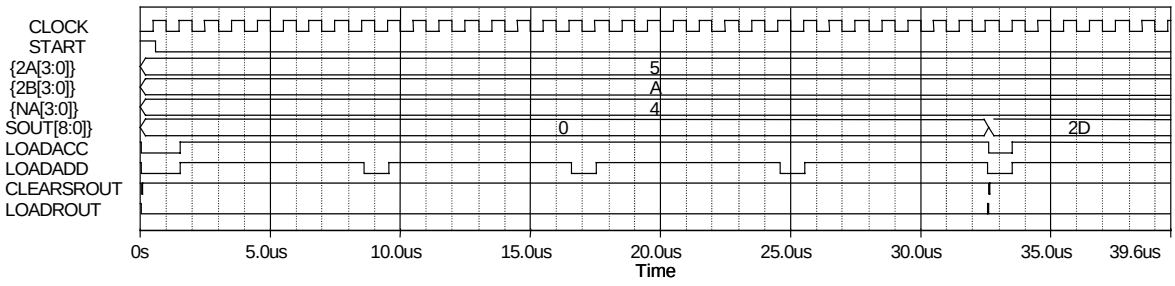


Fig. 11. 50

differenza che il bit di somma viene fatto entrare serialmente nell'accumulatore (nel nostro circuito questo bit è prelevato dall bit meno significativo del SIPO, per accordare le fasi in cui avvengono gli shift nel SIPO e nell'Accumulatore). Quando la somma dei primi due numeri è finita, viene decrementato il contatore di addendi (prima dell'inizio della somma successiva), l'accumulatore conterrà il risultato nella giusta posizione pronto per la prossima somma, il F/F D conterrà il bit di carry. La nuova somma procede con il caricamento del PISO con il terzo addendo (senza l'azzeramento del F/F D). La sequenza ora descritta procede fino a che

il contatore di addendi non si azzera. Quando ciò accade la sequenza di somme è finita, il risultato è contenuto nell'accumulatore e nel F/F D e, come descritto in precedenza, il loro contenuto viene spostato in una memoria esterna. Notare che il F/F D U13A è usato per inizializzare correttamente la prima somma, indipendentemente dal numero di addendi.
Dopo di ciò un nuovo ciclo può avere inizio.
Il timing di Fig. 11.50 è stato ottenuto usando il blocco gerarchico di Fig. 11.49 che punta al circuito di Fig. 11.48. Le traccie qui presentate provengono dai probe connessi come si vede dalla Fig. 11.48 e 11.49.
Il timing mostra che il circuito somma al numero 7, quattro addendi uguali a 10.

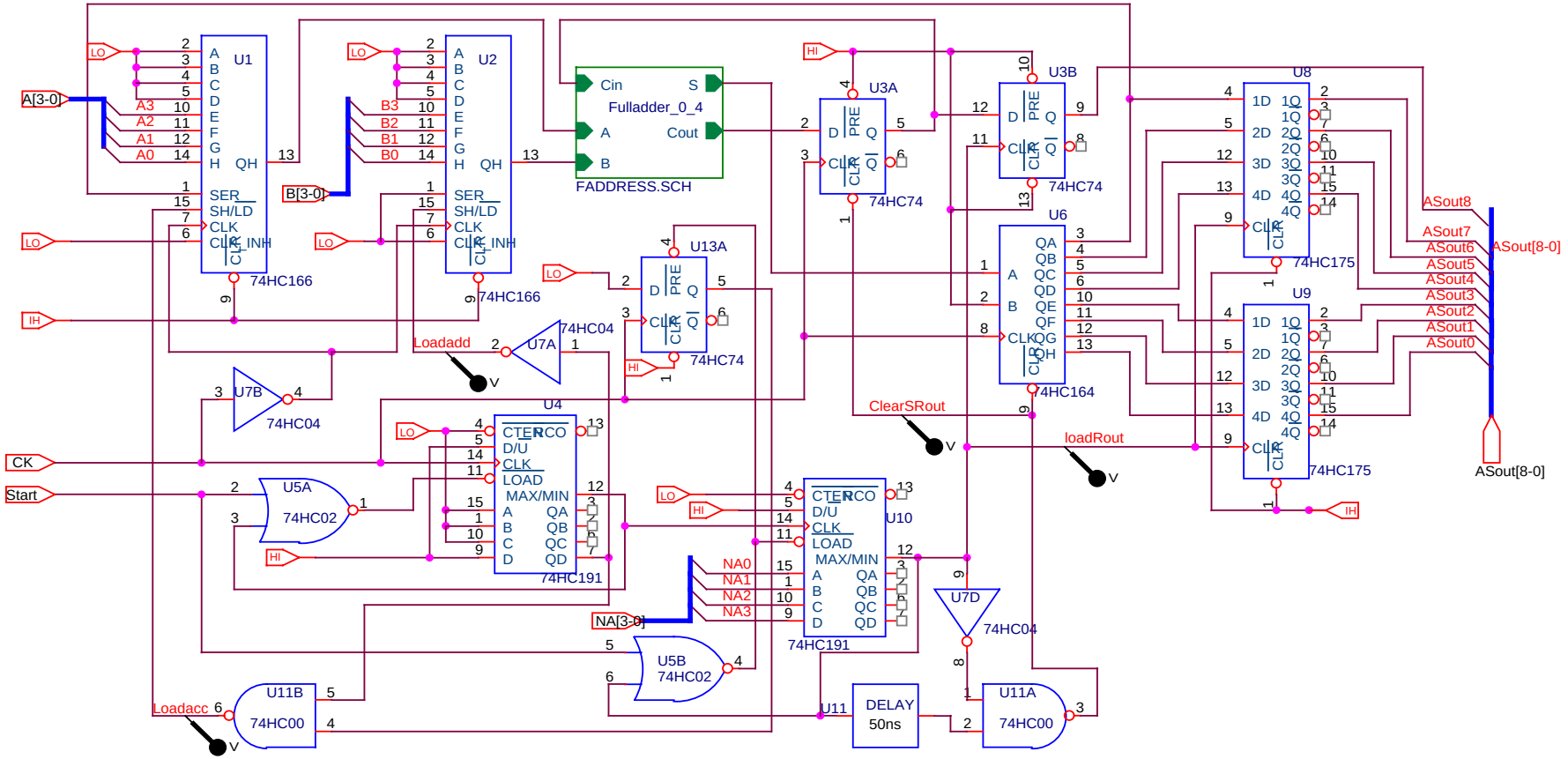

A[3-0]
B[3-0]
U1
U2
74HC166
FADDRESS.SCH
Fulladder_0_4
U3A
U3B
74HC74
U6
74HC164
U8
U9
74HC175
ASout8
ASout7
ASout6
ASout5
ASout4
ASout3
ASout2
ASout1
ASout0
ASout[8-0]
U13A
74HC04
Loadadd
ClearSRout
loadRout
CK
Start
U4
74HC191
U5A
74HC02
U7A
U7B
U7D
U10
NA[3-0]
NA0
NA1
NA2
NA3
U5B
Loadacc
U11B
74HC00
U11
DELAY
50ns
U11A

Fig. 11.48

PROBLEMI

P11.1 Per il contatore sincrono di Fig. P11.1 determinare la sequenza degli stati a partire da ABC= 000. Cosa accade se lo stato iniziale è 101, 110 o 111?. Fare anche il diagramma di stati e il diagramma temporale.

P11.2 Verificare che la forma d'onda di uscita del F/F, con periodo maggiore, di un contatore BCD ha duty cycle <> 50%. Progettare e disegnare gli schemi logici di un contatore sincrono up modulo 10 in modo che la forma d'onda di uscita del F/F con periodo maggiore sia simmetrica cioè con duty cycle =50%.

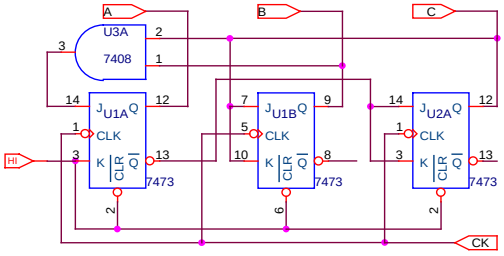


Fig. P11.1

P11.3 Progettare e disegnare gli schemi logici dei seguenti contatori sincroni dotati di ingressi di controllo di abilitazione, di uscita di decodifica del conteggio (massimo, per i contatori UP, minimo per i contatori Down e massimo/minimo per i contatori Up/Down). Inoltre i contatori UP/Down siano dotati di una linea di controllo che ne determina il modo di conteggio. Si esegua il progetto sia a partire alla tabella di stato sia dal timing sia dal diagramma di stato e utilizzare sia F/F JK, sia D.

a) Contatore binario 8421 Down a 4 bit; b) Contatore binario 8421 Up/Down a 4 bit;

c) Contatore Decimale BCD Down; d) Contatore Decimale BCD Up/Down

e1) Contatore binario Up/Down che conta da 0 a 2 e2) che conta da 0 a 3

f1) Contatore binario Up/Down a 3 bit che conta da 0 a 4 f2) che conta da 0 a 5

f3 che conta da 0 a 6 f4) che conta da 0 a 7

g1) Contatore binario a 4 bit che conta da 0 a 10 g2) che conta da 0 a 11

g3) che conta da 0 a 12 g4) che conta da 0 a 13

g5) conta da 0 a 14

P11.4. Utilizzando, come blocchi funzionali, i contatori progettati in P11.2 e/o quelli del testo (che trovate necessari) fare lo schema a blocchi di un Orologio/Cronometro che a partire da un generatore di clock con frequenza pari a 1.000.000 Hz (1 Mega Hertz) fornisce

a) il numero di ore, b) di minuti, c) di secondi, e d) il numero di millisecondi.

P11.5. Modificare lo schema realizzato per la soluzione di P11.3 in modo che possano essere modificati lo stato

a) delle ore, b) dei minuti, c) dei secondi, e d) dei millisecondi.

NOTA: nel mentre si fa l'operazione di setting l'orologio non deve avanzare. Finita la modifica l'orologio riprenda la marcia normale.

P11.6. Modificare il progetto, realizzato per la soluzione di P11.5, in modo che, utilizzando dei comandi e degli impulsi esterni, si possa fare avanzare o diminuire, in modo indipendente,

a) le ore, b) i minuti, c) i secondi, e d) i millisecondi.

NOTA nel mentre si fa l'operazione di setting l'orologio non deve avanzare. Finita la modifica l'orologio riprenda la marcia normale.

P11.7. Modificare il progetto, realizzato per la soluzione di P11.6, in modo che, gli avanzamenti o le diminuzioni delle ore e dei minuti avvengano con il clock di avanzamento dei secondi e le variazioni dei secondi avvengano con il clock dei millisecondi. (durante il tempo in cui il relativo comando è attivo), I millisecondi possono essere soltanto azzerati.

NOTA: nel mentre si fa l'operazione di setting l'orologio non deve avanzare. Finito la modifica l'orologio riprenda la marcia normale.

P11.8. Progettare un circuito che a partire da un'onda quadra a una data frequenza determini una forma d'onda il cui periodo è 11 volte quello della forma d'onda di ingresso e che il rapporto tra la durata del livello alto a quello basso è:

a) di 10 a 1; a1) di 100 a 1. a2) di 1000 a l.
b) quale deve essere il valore di frequenza nei tre casi se il la durata allo stato alto deve essere di 1) un secondo; 2) 0.1 secondo ; 3) 0.001 secondo?

P11.9 Progettare e disegnare un circuito monostabile NON retriggerabile, attivo sul fronte positivo di un impulso di trigger e la cui durata di monostabile è pari a:

a) 5 periodi del clock di sistema; b) 10 periodi del clock di sistema; c) 50 periodi.

NOTA: Dato che il clock di sistema ed il trigger sono asincroni si realizzi il monostabile con le due seguenti caratteristiche: 1) l'impulso di monostabile inizia con l'arrivo del fronte del trigger e ha durata pari a quella richiesta a meno di un periodo; 2) la durata di monostabile è pari al numero di periodi richiesto ma l'istante di inizio dell'impulso è ritardato rispetto all'arrivo dell'impulso di trigger di meno di un periodo di clock.

P11.10 Trasformare i circuiti realizzati con la soluzione del problema 11.8 in modo tale che il monostabile diventi Retriggerabile.

P11.11 Progettare un circuito che misura la frequenza di una forma d'onda esterna (frequenza minima: 1Hz, Massima: 1 MHz).

NOTA: Si usi una base dei tempi (il tempo durante il quale vengono contati i fronti della forma d'onda di ingresso) generata localmente.

P11.12 Progettare un circuito che misura il periodo una forma d'onda esterna.

NOTA: Si usi la forma d'onda esterna come base dei tempi ed un clock locale come generatore della unità di tempo.

P11.13 Progettare e disegnare lo schema di un registro a scorrimento a destra con ingresso e uscita seriale a 4 bit.

P11.14 Progettare e disegnare lo schema di un registro a scorrimento a sinistra con ingresso e uscita seriale a 4 bit.

P11.15 Progettare e disegnare lo schema di un registro a scorrimento a sinistra o a destra con ingresso e uscita seriale a 4 bit.

P11.16 Progettare e disegnare lo schema di un registro a scorrimento a destra con ingresso parallelo e uscita seriale a 4 bit.

P11.17 Progettare e disegnare lo schema di un registro a scorrimento a sinistra con ingresso parallelo e uscita seriale a 4 bit.

P11.18 Progettare e disegnare lo schema di un registro a scorrimento a destra con ingresso seriale e uscita parallelo a 4 bit.

P11.19 Progettare e disegnare lo schema di un registro a scorrimento a sinistra con ingresso seriale e uscita parallelo a 4 bit.

P11.20 Progettare e disegnare lo schema di un registro PISO che faccia uno scorrimento aritmetico a destra del numero (a 8 bit) d'ingresso positivo o negativo codificato in complemento a 2.

P11.21 Progettare e disegnare lo schema di un registro PISO che faccia uno scorrimento aritmetico a sinistra del numero (a 8 bit) d'ingresso positivo o negativo codificato in complemento a 2.

P11.22 Progettare e disegnare lo schema, completo di registri di ingresso parallelo e di memoria di uscita, per il problema P10.3.

P11.23 Progettare e disegnare lo schema, completo di registri di ingresso parallelo e di memoria di uscita, per il problema P10.4.

P11.24 Progettare e disegnare lo schema, completo di registri di ingresso parallelo e di memoria di uscita, per il problema P10.5.

P11.25 Progettare e disegnare lo schema, completo di registri di ingresso parallelo e di memoria di uscita, per il problema P10.6.

P11.26 Progettare e disegnare lo schema, completo di registri di ingresso parallelo e di memoria di uscita, per il problema P10.7.

P11.27 Progettare e disegnare lo schema, completo di registri di ingresso parallelo e di memoria di uscita, per il problema P10.8.

P11.28 Progettare un circuito contatore Modulo 325.

NOTA: Utilizzare contatori di modulo opportuno (massimo 16) connessi in cascata. Si definiscano le caratteristiche, che si suppone abbiano ogni uno dei contatori.

P11.29 Progettare un circuito rivelatore di transizioni positive e negative di una forma d'onda digitale asincrona rispetto ad un clock locale. L'impulso di uscita di tale circuito deve essere sincrono con il fronte di salita del clock locale e di durata un semiperiodo dello stesso. Vedi Fig. P11.2

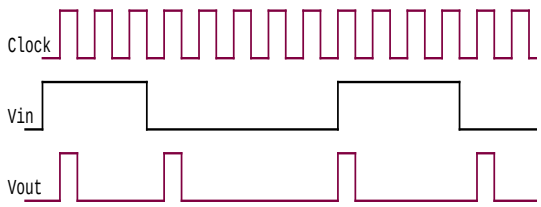


Fig. P11.2

P11.30 Progettare un circuito rivelatore di transizioni positive e negative di una forma d'onda digitale asincrona rispetto ad un clock locale. L'impulso di uscita di tale circuito deve essere sincrono con il fronte di salita del clock locale e di durata 2 periodi dello stesso. Vedi Fig. P11.3

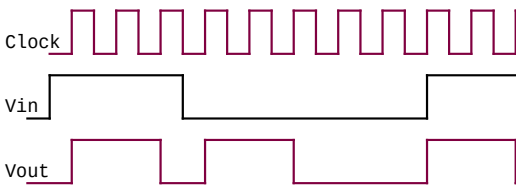


Fig. P11.3

P11.31 Progettare un circuito logico sequenziale che

a) Trasferisca una byte parallelo in una stringa di bit seriale (NRZ-L) (vedi Fig. P11.4, rigo 3, per un esempio)

b) Trasforma la stringa NRZ-L in una Stringa BiFase-L (vedi Fig. P11.4 rigo 4) Essenzialmente essa consiste in una combinazione del dato NRZ-L con il clock. Il risultato deve essere tale che, a metà periodo del clock, la forma d'onda ha sempre una transizione negativa se il dato NRZ-L trasmesso nel periodo è un 1 viceversa una transizione positiva. La trasmissione seriale così trovata si dice auto clocking ed è usata con qualche modifica nelle trasmissione dati TCP/IP.

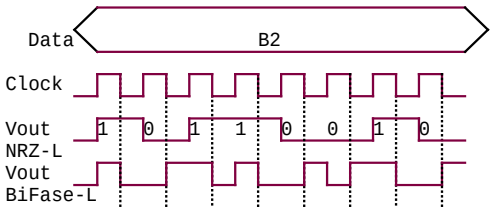


Fig. P11.4

P11.32 Supponendo di conoscere il valore del periodo del clock di Fig. 11.4 progettare un circuito logico sequenziale che

a) Ricostruisce, dalla stringa BiFase-L, la forma d'onda del clock con la corretta relazione di fase.

b) Rivela valori della stringa seriale che ha generato la stringa BiFase-L

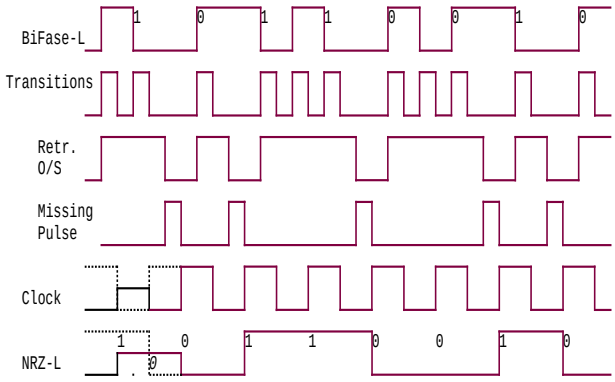


Fig. P11.5

Suggerimento: La Fig. P11.5 mostra il timing che risolve il problema a partire dalla stringa BiFase-L di Fig. P11. 4. Si noti il clock ricostruito e il dato rivelato hanno la probabilità, inizialmente, di avere rispettivamente la fase e valore sbagliato. Questi errori sono superati con la prima variazione del dato trasmesso (da 1 a0 o da 0 ad 1) nella stringa BiFase-L. La stringa NRZ-L è ricostruita campionando, al fronte di salita del clock ricostruito, i livelli BiFase-L.

P11.33 Nelle trasmissioni seriale asincrone di dati, è necessario distinguere un dato trasmesso dal successivo (sincronizzazione). Per ottenere ciò, alla stringa di dati (che secondo lo standard RS232 può essere formato di 5, 6, 7, 8 bit) è preceduta da uno o più bit di **start** a **zero logico** ed è seguita da almeno un bit di **stop** a **uno**. I bit di dati sono quindi codificati in NRZ-L. La definizione di una trasmissione seriale asincrona è quindi fissata:

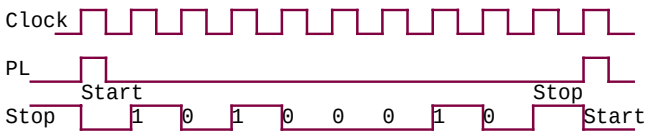


Fig. P11.6

1) Dal **Baud Rate** (la frequenza con cui vengono trasmessi i singoli bit). Il periodo è noto come "*bit-time*" ossia il tempo impiegato per la trasmissione di un singolo bit,
2) Dal **numero di bit di start**,
3) Dal **numero di bit di stop**,
4) Il **numero di bit di cui è formato il dato**,

5) Se in testa al dato, è posto un bit di **parità pari o dispari**.

La trasmissione del dato avviene sempre a partire dal bit meno significativo. La Fig. P11.6 rappresenta il timing di una trasmissione asincrona (continua) con un bit di start, un bit di stop ed 8 bit di dati, $(45)_{16}$.

Progettare e disegnare un circuito logico sequenziale che, a partire da un byte di dati parallelo (bit di parità pari o dispari compreso), ne faccia la trasmissione seriale aggiungendo un solo bit di start ed un solo bit di stop. Si supponga di avere un generatore di onda quadra da cui scegliere una delle frequenze standard (110, 300, 600, 1200. 2400, 4800, 9600, 19200, 38400, 115200).

P11.34 Le UART (Universal Asynchronous Receiver-Transmitter) oltre a trasmettere le informazioni seriali, con un circuito funzionalmente come descritto in P11.28 e realizzato con la soluzione del citato problema, ricevono le informazioni inviate da analoghe unità remote. Le UART per ricevere le informazioni seriali asincrone remote, in generale, utilizzano l'informazione della Baud Rate (nominale) e generano un clock locale che ha una frequenza **16 volte maggiore**. Usano quindi questo clock locale (asincrono rispetto al quello della Baud Rate della stringa di dati in ricezione) per ricostruire un clock pari a quella della Baud Rate nominale e a questa (quasi) sincrona.

Progettare e disegnare lo schema di un circuito digitale sequenziale che genera il clock con le caratteristiche sopra descritte. Si supponga di possedere un generatore di onda quadra alla frequenza richiesta.

Suggerimenti e Osservazioni: la sincronizzazione è fatta utilizzando le transizioni presenti nella stringa NRZ-L. Piccole variazione dovute al fatto che i due clock di partenza sono tra di loro asincroni e con, possibilmente, valori di frequenza che si scostano di pochi percento da quella nominale non provocheranno errori di ricezione.

P11.35 Utilizzando il clock pari alla Baud Rate nominale, ricostruito con la soluzione del problema P11.29 e il clock per 16, progettare un circuito sequenziale che campioni la stringa NRZ-L. Il campionamento per rivelare il valore del bit va fatto (circa) metà di ogni bit time.

P11.36 Si completi il progetto del circuito, realizzato con la soluzione del problema P11.30 che, conoscendo le caratteristiche di trasmissione (baud rate, numero di bit di start (zero), numero di bit di stop (uno) e numero di bit di dati (otto))

a) Esegue la sincronizzazione al livello di dato (byte) verificando che nella stringa seriale ricevuta tra un bit di start ad uno ed il bit di stop a zero logico insistono esattamente il numero di bit stabilito (otto).
b) Elimini il bit di start ed il bit di stop e trasferisca il dato in parallelo ad una memoria.