Un simulatore parallelo di circuiti elettronici (4)

Circuiti sequenziali

di Giuseppe Cardinale Ciccotti

Il nostro simulatore comincia a prendere corpo e finalmente abbiamo una buona struttura anche se ancora da perfezionare, ci serve tuttavia ancora un processo che ci permetta di monitorare i segnali che ci interessano una specie di registratore del segnale che non interferendo sul segnale stesso lo traferisca sul nostro dispositivo di output. Ricordate comunque che, anche se stiamo cercando di astrarre questo progetto dai sistemi di sviluppo per Transputer abbiamo comunque necessità di riferirci ad un oggetto concreto nei nostri esempi, perciò ci riferiremo al nostro host con cui comunichiamo tramite i canali fs e ts, se qualcuno è così fortunato da poter utilizzare invece delle schede a Transputer che pilotano direttamente i dispositivi di output, si servirà direttamente di questi

Quello che non tratteremo è l'interfacciamento con l'host, supporremo perciò che i dati che ci interessa visualizzare dovranno essere salvati dall'host ed in seguito, eseguita la simulazione, visualizzati. Si potrebbe fare di meglio, ne siamo pienamente coscienti, ma ci interessa focalizzare la nostra attenzione sui meccanismi Occam che sono alla base del nostro simulatore. Se invece c'è qualche lettore di buona volontà che vuole imbarcarsi nell'impresa di progettare un'interfaccia per questo simulatore che permetta di immettere dei parametri e di visualizzare i risultati della simulazione mentre il simulatore gira, si faccia avanti in modo tale che possiamo definire le modalità di interfacciamento, chiaramente su queste pagine verrà dato risalto ad un lavoro siffatto.

Il registratore di segnali

Finora abbiamo visto come programmare il funzionamento dei componenti elementari, tuttavia non esiste nulla che ci permetta di renderci conto dei risultati della simulazione. Supponiamo di trasferire i dati utili all'host e a questo a simulazione terminata affidare il compito di visualizzare come meglio crediamo o tramite un pacchetto i risultati.

Per prima cosa dobbiamo definire come organizzare tali dati: la cosa migliore sembra quella di registrare soltanto i cambiamenti di stato del segnale piuttosto che lo stato per ogni colpo di clock. In questo modo rimaniamo svincolati dalla grandezza dello step del clock scelto per la simulazione; per contro sarà necessario memorizzare anche il tempo in cui si è verificato tale cambiamento; la cosa più semplice è quella di organizzare un vettore di Nx2 Interi a 16 bit. Bisogna però, tenere presente che l'Occam non prevede l'allocazione dinamica e pertanto bisogna stabilire a tempo di compilazione la grandezza di tali vettori cioè fissare il valore della N.

In effetti, non siamo in grado a priori di saper quanto deve essere perché nel caso peggiore potremmo trovarci a

```
PROC ProbeRec (CHAN OF INT16 probe, output, mclock, off)

  [n,2]INT 16 track:
  INT16 signal,status,nsample,ck,counter,any, mclockCycle,i:
  BOOL active :

    SEQ
      mclock ? mclockCycle
      active := TRUE
      counter := 0
      nsample :=0
        WHILE active
          ALT
            probe? signal
              IF
                (NOT(signal=status))
                    SEQ
                      track[nsample,0]:=signal
                      track[nsample,1]:=counter*mclockCycle
                      nsample:=nsample+1
                      status:=signal
                        IF
                          (nsample=n)
                            SEQ i=0 FOR n-1
                              output ! track[i,0]
                              output ! track[i,1]
                              nsample:=0
                          TRUE
                            SKIP
                TRUE
                  SKIP
            mclock ? ck
              counter:=counter+1
            off ? any
              active := FALSE
      SEQ i=0 FOR nsample-1
        output ! track[i,0]
        output ! track[i,1]
:
```

Figura 1
Listato processo ProbeRec.

```
PROC MultiTrackRec (CHAN OF [m] INT16 probe, CHAN OF INT 16
mclock, off, CHAN OF SP fs,ts)

  VAL fname IS "Wave.dat"

  [n*2]INT 16 track:
  INT32 mystream:
  INT16 signal,any, mclockCycle,i,j :
  BYTE ret1,ret2:
  BOOL active :

    SEQ
      mclock ? mclockCycle
      active := TRUE
      so.open(fs,ts,fname,spm.text,spm.output,mystream,ret1)
        WHILE (active AND (ret1=spr.ok))
            PAR
              ALT  i:=0 FOR m
                probe [i]? signal
                  SEQ
                    track[0]:=signal
                    SEQ j:=1 FOR n*2-1
                      probe [i]? signal
                      track[j]:=signal
                  so.fwrite.int (fs,ts,mystream,i,0,ret2)
                    SEQ j:=0 FOR n*2-1
                      so.fwrite.int
(fs,ts,mystream,track[j],0,ret2)
              off ? any
                active := FALSE
      so.close(fs,ts,mystream,ret1)
:
```

Figura 2 - Listato del processo MultiTrackRec versione senza supporto del processo multiplexer di libreria.

```
PROC MultiTrackRec (CHAN OF [m] INT16 probe, CHAN OF INT 16
mclock, off, CHAN OF SP fs,ts)

  VAL fname IS "Wave.dat"

  [m]SP myfs,myts:
  [m,n*2]INT 16 track:
  INT32 mystream:
  INT16 signal,any, mclockCycle,i,j :
  BYTE ret1,ret2:
  BOOL active :

    SEQ
      mclock ? mclockCycle
      active := TRUE
      so.open (fs,ts, f,spm.text,spm.output,mystream,ret1)
      so.multiplexor(fs,ts,myfs,myts,stopper)
        WHILE (active AND (ret1=spr.ok))
          PAR
            PAR  i:=0 FOR m
              SEQ
                probe [i]? signal
                track[i,0]:=signal
              SEQ j:=1 FOR n*2-1
                probe [i]? signal
                track[i,j]:=signal
                so.fwrite.int
(myfs[i],myts[i],mystream,i,0,ret2)
              SEQ j:=0 FOR n*2-1
                so.fwrite.int
(myfs[i],myts[i],mystream,track[i,j],0,ret2)
            off ? any
              active := FALSE
      so.close(fs,ts,mystream,ret1)
:
```

Figura 2a - Listato del processo MultiTrackRec con il supporto del processo multiplexer di libreria.

campionare un segnale che varia velocemente quanto il clock (potremmo sempre inventare un tale componente); in tal caso N è pari al numero di cicli previsti per la simulazione. Questo caso è quantomeno assai raro, ciò non toglie che una stima precisa non è possibile.

L'alternativa necessaria consiste nel registrare un numero fisso di stati e poi trasferire il vettore all'host finché non è terminata la simulazione, l'host poi li potrà scaricare in un file ad esempio o tenerli in memoria. Il valore di N deve perciò essere calcolato in base al sovraccarico di comunicazione verso l'host che è notoriamente lento, bisognerà fare in modo che N sia il più grande possibile compatibilmente con le capacità di memoria del Transputer su cui gira questo processo. In tal modo si minimizza il tempo necessario al Transputer per inizializzare una comunicazione con l'host.

C'è poi da puntualizzare che non verrà mantenuto un solo vettore bensì un vettore per ciascun punto del circuito che si vuole monitorare, dovranno perciò essere allocati e trasmessi M vettori di N componenti ognuno. Sempre per il problema dell'allocazione statica dei dati è necessario definire a tempo di compilazione il numero dei punti che si vogliono monitorare e quindi bisognerà architettare un altro processo multiplexer come il processo MasterClock presentato nella parte terza.

In figura 1 potete vedere il listato del processo ProbeRec che non fa niente altro che «acchiappare» tutto ciò che gli viene passato dal dispositivo connesso in ingresso. L'unico controllo che esegue serve a memorizzare effettivamente soltanto la variazione dello stato del segnale ricevuto.

Quando il vettore track è pieno cioè nsample è pari alla lunghezza del vetto-

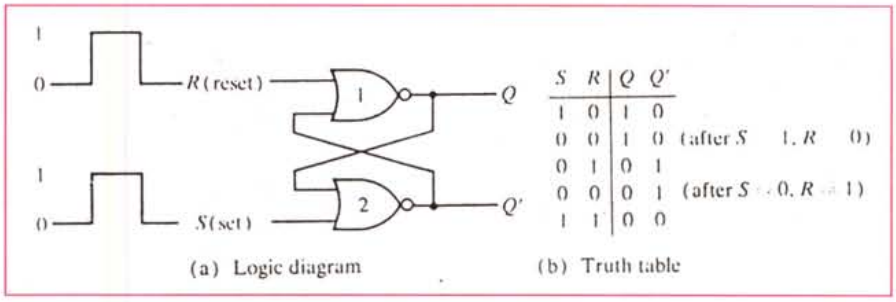

Figura 3 - Flip-Flop tipo RS asincrono con NOR.

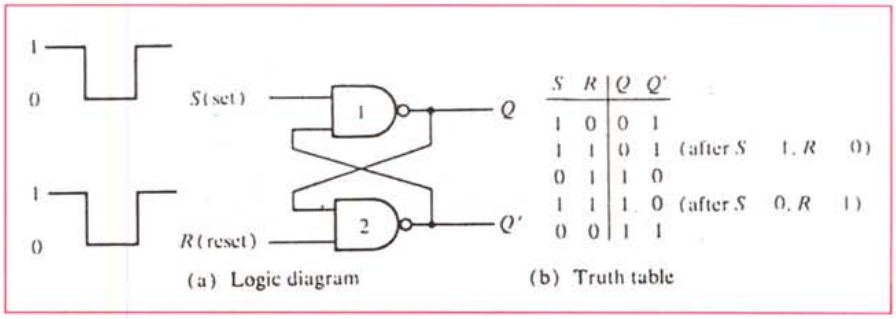

Figura 4 - Flip-Flop tipo RS asincrono realizzato con gate NAND.

re, questo viene scaricato al processo MultiTrackRec che raccoglie tutti i vettori track da tutti i processi ProbeRec disposti nel circuito; si occupa poi di trasferirli all'host passando anche il nome in modo tale che l'host sappia a chi appartengono i dati ricevuti. Ricordate che nei sistemi di sviluppo a Transputer commerciali si dispone di un solo canale fisico di comunicazione con l'host. In figura 2 trovate il listato del processo MultiTrackRec in dettaglio, facciamo uso di qualche chiamata alla libreria «Host.lib» del sistema di sviluppo; in questo caso apriamo un file con so.open vi scriviamo degli interi so.fwrite.int e lo chiudiamo so.close.

Il funzionamento del processo è semplice: aspetta sulla guardia replicata che uno dei processi ProbeRec gli invii i dati, non appena ne riceve da uno «scarica» tutto il vettore in uno locale. In seguito interagisce con l'host scrivendo prima il numero d'ordine del processo che ha trasmesso e poi il vettore dei dati. Si sarebbe potuto fare un unico SEQ ricevendo e trasmettendo senza il vettore di buffer locale ma in tal modo si bloccava il processo ProbeRec alla velocità, molto minore trattandosi oltretutto di accesso a disco, dell'host; con questa soluzione il processo è totalmente svincolato ed una volta scaricato il vettore può continuare a registrare nuovi dati.

In figura 2a possiamo trovare un'altra implementazione del processo MultiTrackRec, utilizziamo un utile processo di libreria so.multiplexor che si occupa di multiplexare il canale di comunicazione con l'host. In definitiva disporremo di un array di canali con l'host piuttosto che uno solo, in tal modo potremo usare il costrutto PAR perché non avremo condivisione di variabili.

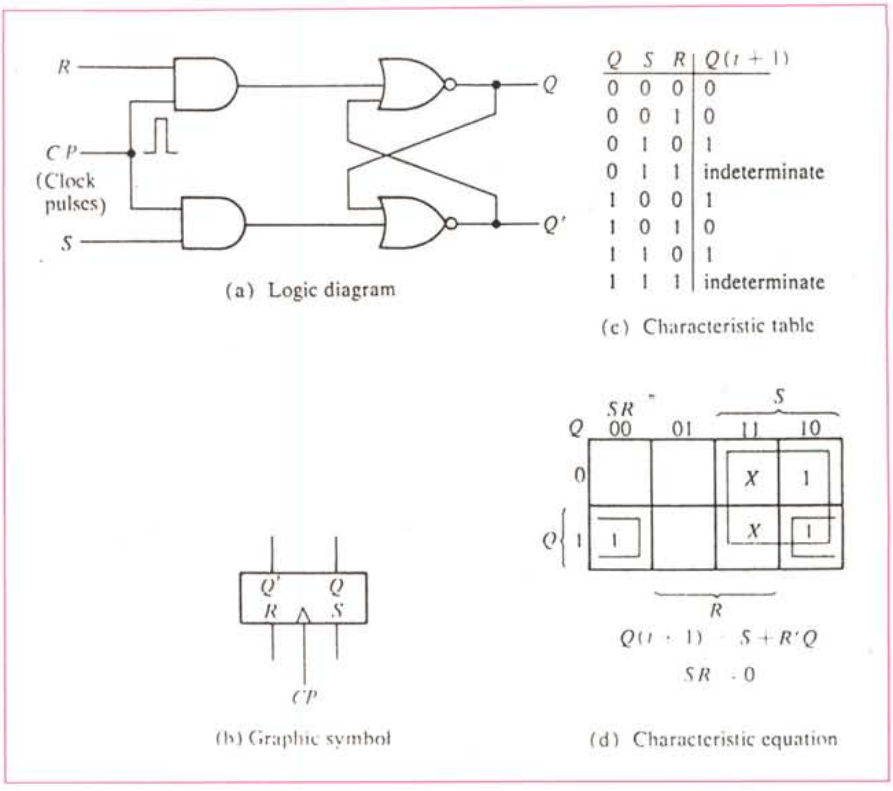

Figura 5 - Flip-Flop tipo RS sincrono.

I Flip-Flop

Finalmente eccoci a parlare del primo componente sequenziale, l'arcinoto Flip-Flop. Rinfreschiamo la memoria: un Flip-Flop è un circuito che può mantenere uno stato binario indefinitamente finché non riceve un segnale che lo forza a cambiare stato. Esistono diversi tipi di Flip-Flop che condividono tale definizione e si distinguono principalmente per il numero di input e per la maniera in cui questi agiscono sullo stato e sull'output.

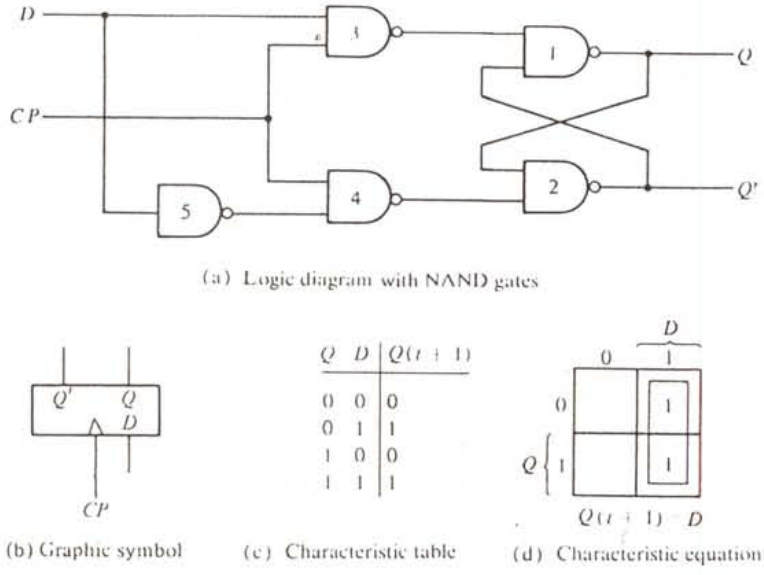

Figura 6 - Flip-Flop tipo D. La porta 5 ha entrambi gli ingressi connessi insieme.

Una breve risposta

Rispondo al sig. Luca Marchese che mi ha indirizzato una richiesta di reperibilità di hardware e documentazione tecnica sull'Occam ed i Transputer.

Purtroppo non esistono in commercio pubblicazioni disgiunte dall'hardware, almeno in Italia.

Tutte le informazioni che noi utilizziamo sono desunte dai manuali forniti con il sistema di sviluppo.

Troverà comunque dei riferimenti nella bibliografia di questa rubrica.

Per i sistemi di sviluppo può rivolgersi alle seguenti ditte:

Lasi Elettronica per i sistemi Inmos
I.T.D. per i prodotti TransTech.

Esistono comunque altre schede più economiche come la CSA kit di cui però non abbiamo un certo riferimento.

Non appena avremo notizie di novità in tal senso le comunicheremo su queste pagine.

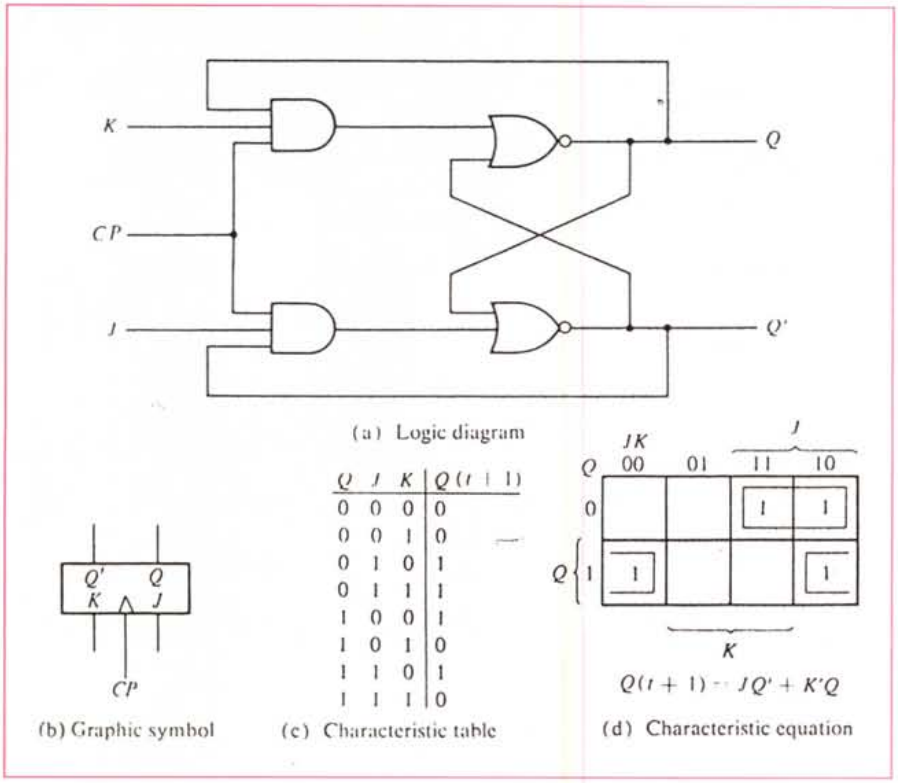

Figura 7 - Flip-Flop tipo JK.

Discuteremo dei tipi più noti di Flip-Flop.

Il circuito base del Flip-Flop può essere costruito con due porte NOR come in figura 3 o NAND come in figura 4. Questi semplici circuiti possono essere usati per costruire tipi di Flip-Flop più complessi, come vedremo.

È importante notare come l'uscita di una porta è connessa ad un ingresso dell'altra, formando un feedback che rende il circuito intrinsecamente asincrono.

Ciascun Flip-Flop ha due uscite, Q e Q', e due ingressi Set e Resest, per questo motivo questo circuito è chiamato Flip-Flop RS. La tavola della verità esplicita abbastanza bene il funzionamento del circuito, tuttavia è necessario precisare meglio la modalità di funzionamento.

Ricordiamo che l'uscita della porta NOR è 0 quando uno qualsiasi degli ingressi è 1 e che l'uscita vale 0 solo se tutti gli ingressi sono 0. Assumiamo dunque che l'ingresso S sia 1 e che quello R sia 0, visto che la porta 2 ha l'uscita a 0 significa che la porta 1 avendo gli ingressi a 0 darà 1 in uscita su Q. Quando S ritorna a 0, le uscite rimangono invariate perché l'uscita Q rimane ad 1, lasciando un ingresso della porta 2 ad 1. Ciò provoca a sua volta che Q' sia 0, il che lascia entrambi gli input della porta 1 a 0 in modo che Q sia 1.

Con lo stesso ragionamento è possibile dimostrare che un 1 sull'ingresso R cambia le uscite Q a 0 e Q' ad 1. Questi stati sono dunque stabili.

Se invece entrambi gli ingressi R e S sono pari ad 1, tutte e due le uscite Q e Q' vanno a 0. Questo risultato viola la condizione che Q' sia sempre il complemento di Q e viceversa. È necessario pertanto prendere cura che tale condizione non si verifichi. Infatti lo stato stabile si ha quando gli ingressi S e R rimangono a 0, allora il Flip-Flop mantiene il suo stato. Se invece R e S vengono posti ad 1 e poi ritornano a 0, lo stato del Flip-Flop è indeterminato e dipende da quale input rimane a 1 più a lungo prima di ritornare a 0.

Il circuito con le porte NAND si comporta nello stesso modo con la considerazione che gli 0 siano sostituiti agli 1.

Non implementeremo tale Flip-Flop asincrono perché nella realtà non è mai usato e poi è talmente semplice che ognuno dei lettori che ci segue è in grado di costruire il programma Occam sfruttando i processi gNAND o gNOR che abbiamo illustrato negli articoli precedenti.

Flip-Flop RS clockato

Come abbiamo appena visto il circuito base del Flip-Flop è intrinsecamente asincrono, tuttavia aggiungendo un paio di porte è possibile rendere il Flip-Flop sensibile agli ingressi soltanto in presenza del clock.

Questo tipo di Flip-Flop è in figura 5 e consiste del circuito NOR base e di due porte AND. Le uscite delle due porte AND rimangono a 0 finquando l'ingres-

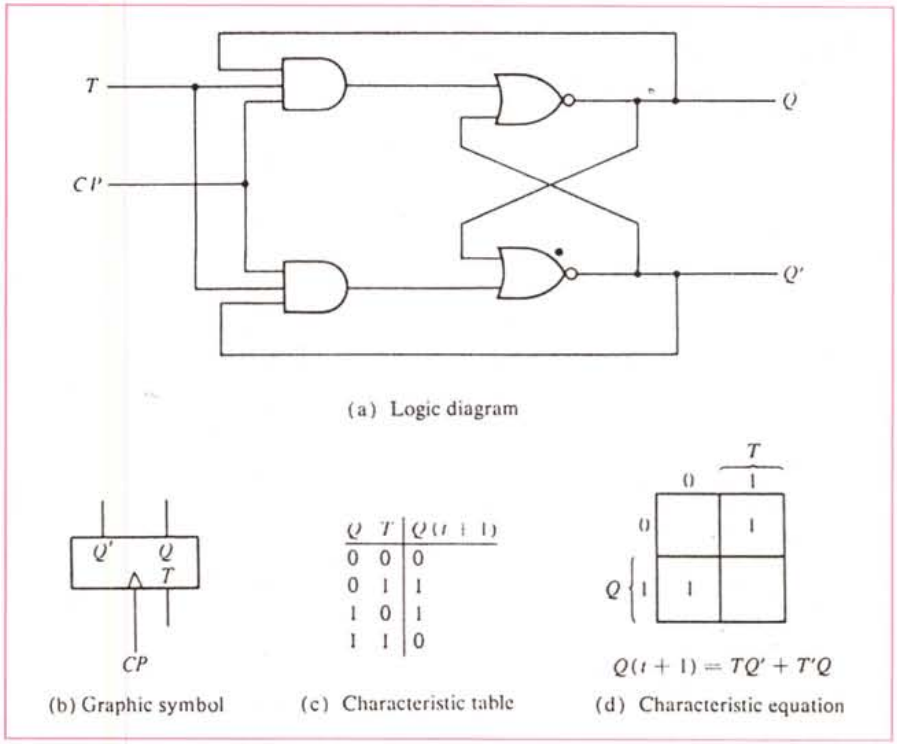

Figura 8 - Flip-Flop tipo T.

so CP (Clock Pulse) permane a 0, e gli ingressi R e S non hanno nessuna influenza. Quando CP è 1, i valori presenti su R e S sono abilitati verso le porte NOR.

Notate un fatto importante: la tavola della verità è significativamente diversa da quelle finora incontrate perché a destra compare anche Q; ciò sta ad indicare che lo stato in cui va il Flip-Flop dipende anche dallo stato presente in cui si trova. Perciò la colonna sinistra della tavola della verità è nominata Q(t+1), proprio ad indicare lo stato successivo.

Bibliografia

M.M. Mano, ***«Digital Design»***, Prentice-Hall, 1984
D. Pountain, D. May, ***«A tutorial introduction to Occam programming»***, BSP, 1988.

Flip-Flop tipo D

Dal Flip-Flop RS precedente si può originare un altro tipo di circuito, aggiungendo una sola porta, in figura 6 c'è lo schema del Flip-Flop D realizzato con sole porte NAND. Le ultime due porte a sinistra sono il circuito base di figura 4 mentre i due NAND a monte sostituiscono gli AND della figura 5 (il segnale è attivo basso). L'ingresso D va direttamente all'ingresso S, ed il suo complemento, tramite la porta 5 (il NAND è usato come NOT, i due ingressi sono connessi insieme), è applicato a R. Finquando il clock è 0, le porte 3 e 4 hanno 1 sulle loro uscite e gli altri ingressi non contano. Il segnale su D viene campionato quando CP va ad 1, se è 1 l'uscita della porta 3 va a 0, forzando lo stato

```
PROC gAND (CHAN OF INT16 x, y, output, mclock, off)

    VAL ritardo IS 50 :
    VAL UnknowState IS #AAAA:

    INT16 dx,dy,ck,do,counter,any, mclockCycle :
    BOOL active :

    SEQ
      mclock ? mclockCycle
      active := TRUE
      counter := 0
      WHILE active
        SEQ
          ALT
            x ? dx
              WHILE (((counter*mclockCycle) < ritardo) AND
active)
                ALT
                  mclock ? ck
                    counter := counter + 1
                  x ? d
                    counter := 0
                  y ? dy
                    counter := 0.
                  off ? any
                    active := FALSE
            y ? dy
              WHILE (((counter*mclockCycle) < ritardo) AND
active)
                ALT
                  mclock ? ck
                    counter := counter + 1
                  x ? dx
                    counter := 0
                  y ? dy
                    counter := 0
                  off ? any
                    active := FALSE
          off ? any
            active := FALSE

          IF
            (NOT(dx = Uknowstate)) AND (NOT(dy =
Uknowstate))
              do := (dx BITAND dy)
              output ! do
              counter := 0
            TRUE
              SKIP
:

PROC gNOR (CHAN OF INT16 x, y, output, mclock, off)

    VAL ritardo IS 50 :
    VAL UnknowState IS #AAAA:

    INT16 dx,dy,ck,do,counter,any, mclockCycle :

    BOOL active :

    SEQ
      mclock ? mclockCycle
      active := TRUE
      counter := 0
      WHILE active
        SEQ
          ALT
            x ? dx
              WHILE (((counter*mclockCycle) < ritardo) AND
active)
                ALT
                  mclock ? ck
                    counter := counter + 1
                  x ? d
                    counter := 0
                  y ? dy
                    counter := 0
                  off ? any
                    active := FALSE
            y ? dy
              WHILE (((counter*mclockCycle) < ritardo) AND
active)
                ALT
                  mclock ? ck
                    counter := counter + 1
                  x ? dx
                    counter := 0
                  y ? dy
                    counter := 0
                  off ? any
                    active := FALSE
          off ? any
            active := FALSE

          IF
            (NOT(dx = Uknowstate)) AND (NOT(dy =
Uknowstate))
              do := (NOT(dx BITOR dy))
              output ! do
              counter := 0
            TRUE
              SKIP
:

PROC connect (CHAN OF INT16 x, y, z)

  INT16 a :

    SEQ
      x ? a
      y  ! a
      z  ! a
:

PROC FFRS(CHAN OF INT16 r,s,cp,q,q')

  CHAN OF INT16 e1,e2,e3,e4,e5,e6,e7,e8:
  CHAN OF  4 INT16 mc,off:

    PAR
      gAND(r,e1,e3,mc 1 ,off 1 )
      gAND(s,e2,e4,mc 2 ,off 2 )
      gOR(e3,e5,e7,mc 3 ,off 3 )
      gOR(e4,e6,e8,mc 4 ,off 4 )
      connect(cp,e1,e2)
      connect(e7,e6,q)
      connect(e8,e5,q')
      MasterClock(mc,off)
:
```

Figura 9 - Listato del processo per il Flip-Flop RS sincrono.

del Flip-Flop ad 1, se invece è 0 lo stato del Flip-Flop va a 0. Praticamente il clock abilita il trasferimento del dato su D nel Flip-Flop, da qui il nome D.

Flip-Flop JK

Il Flip-Flop JK non è altro che una modifca di un RS affinché non si verifichino stati indeterminati. In figura 7 potete vedere lo schema di tale circuito nel quale J sta per S e K per R. Quando un 1 è applicato simultaneamente a J e a K allora il Flip-Flop «switcha» il suo stato nel complemento, quindi se Q=1 avremo Q'=0 e viceversa. Il funzionamento nei restanti casi è esattamente come il Flip-Flop tipo RS.

C'è un unico vincolo da rispettare utilizzando tale Flip-Flop: come si evince dalla tavola della verità e dall'analisi del circuito, se il clock rimane attivo per un tempo maggiore del tempo di propagazione del Flip-Flop quando J e K sono pari ad 1, si avrà un fastidioso ping-pong sulle uscite che continueranno a commutare da uno stato all'altro. È necessario pertanto che il duty cycle del clock sia strettamente inferiore al tempo di propagazione del Flip-Flop JK. Questo è il motivo per cui in pratica il Flip-Flop JK non è costruito in questo modo, ma questo ulteriore raffinamento lo vedremo in un prossimo appuntamento.

```
PROC FFD(CHAN OF INT16 d,cp,q,q')
  CHAN OF INT16 e1,e2,e3,e4,e5,e6,e7,e8:
  CHAN OF [5] INT16 mc,off:
    PAR
      gNAND(e1,e3,e5,mc[1],off[1])
      gNAND(e13,e4,e6,mc[2],off[2])
      gNAND(e11,e12,e13,mc[3],off[3])
      gNAND(e5,e7,e9,mc[4],off[4])
      gNAND(e6,e8,e10,mc[5],off[5])
      connect(cp,e3,e24)
      connect(d,e1,e2)
      connect(e9,e8,q)
      connect(e10,e7,q')
      connect(e2,e11,e12)
      MasterClock(mc,off)
:
```

Figura 10 - Listato per il Flip-Flop D. È necessaria una connessione sull'ingresso del gate 5.

Flip-Flop T

Per ultimo presentiamo il Flip-Flop tipo T che è la versione a singolo input del Flip-Flop JK, infatti come si vede in figura 8 è ottenuto dal JK connettendo gli ingressi insieme. Il nome T gli deriva dalla abilità di «togghelare» o meglio commutare lo stato. Semplicemente esso complementa lo stato Q quando il clock è 1 e se T vale 1.

I processi per i Flip-Flop

Nelle figure 9 e10 sono listati i processi per i Flip-Flop RS clockato, e D. Utilizzano come abbiamo già visto i processi per le porte che ormai fanno parte stabile della nostra libreria di processi.

Notate che i processi connect che simulano le connessioni non hanno necessità di essere agganciate al MasterClock se non per il canale off, infatti non devono «esistere» dal punto dei vista della propagazione dei risultati.

Conclusioni

Abbiamo finalmente costruito i primi ed importanti circuiti sequenziali. Ora possiamo finalmente andare avanti ad implementare circuiti un po' più complessi e verificare il funzionamento di strutture hardware più evolute.

MC

ComTree si prende cura delle vostre connessioni

On The Road... PalmLAN

II PRIMO adattatore Pocket LAN al mondo a batteria

- Connessione alla porta parallela per tutti i portatili e desktop.
- Richiede solo un terzo della potenza di altri pocket LAN, il che aiuta ad allungare la vita delle batterie di un portatile.
- Supporta sia il conntore BNC che l'RJ-45 con configurazione automatica. (Disponibile anche per solo cavo coassiale o UTP)
- Supporta i driver per Novell Netware, Microsoft LAN Manager, FTP TCP/IP.

O, in ufficio... VinyLAN

- Selezione automatica di differenti tipi di interfaccia - AUI, BNC o UTP.
- Alta integrazione grazie ad ASIC proprietario per alte performance e affidabilità.
- Cinque indicatori a LED provvedono a monitorare il funzionamento, i problemi ed una più semplice installazione.
- Supporta i driver per Novell Netware, Microsoft LAN Manager, FTP TCP/IP.

Service/Support

1. 2 Anni di garanzia (opzionale 5 anni di garanzia)
2. Sostituzione prodotti difettosi.
3. Risposta entro le 48 ore a problemi tecnici.

We Care About Your Connection

ComTree

ComTree Technology Corporation

5F-7, No. 1, Fu-Hsing North Rd., Taipei, Taiwan, R.O.C. Tel: 886-2-752-9075, Fax: 886-2-752-2449