Latch e Flip-Flop*

Corso di Architetture dei Sistemi di Elaborazione

Prof. Antonino Mazzeo

29 ottobre 2018

Univ. di Napoli Federico II

*Si avvisa che il presente capitolo è in bozza e può contenere errori e imprecisioni

Notazioni utilizzate

I valori vero o falso delle variabili booleane sono indicati dal nome senza alcun segno, se veri e dal nome con apposta una barra superiore ($\overline{X}$) se falsi. Oltre a tale denotazione, in molti testi, spesso per problemi tipografici, per rappresentare le variabili negate sono utilizzate altre notazioni quali: un apice (X') o un punto esclamativo (X!) o un asterisco, (X*) posposto al nome della variabile e, in taluni casi, anche con una barra a esso preposta (/X). Nel presente testo si utilizzeranno o la notazione con barra superiore o quella con apice (X').

Gli operatori AND e OR sono indicati rispettivamente con i simboli $\cdot$ e $+$, ad esempio $(X \cdot Y)$ e $(X + Y)$; gli operatori NAND e NOR rispettivamente con una freccia in alto $(X \uparrow Y)$ e una freccia in basso $(X \downarrow Y)$.

1 Introduzione

Questo capitolo è dedicato alla presentazione dei componenti bistabili (i flip-flop e i latch), reaòizzati con tecnologie elettroniche. I dispositivi ampiamente impiegati nella progettazione dei sistemi digitali e alla base della costruzione dei registri e delle memorie, rappresentando essi una cella elementare di memoria della capacità di un bit. I flip-flop e i latch sono relizzati a partire da un particolare circuito elettronico analogicio detto multivibratore *bistabile*, composto da componenti elettronici attivi (inizialmente costruito a valvole e, successivamente, con dispositivi a semiconduttore di varie tecnologie quali i transistor bipolari e quelli MOS).

I flip-flop e i latch, possono essere studiati con i metodi tradizionali di analisi e progettazione dei circuiti elettronici o come dispositivi logici, utilizzando i metodi di analisi e progettazione tipici delle reti logiche (reti combinatorie e sequenziali sincrone e asincrone) e sintetizzati utilizzando come componenti elementari le porte logiche di tipo inverter, AND, OR, NAND, NOR, etc..

Nel seguito, è presentata, dopo una prima sintetica illustrazione dei circuiti bistabili, la classificazione e l'illustrazione delle caratteristiche funzionali dei latch e flip-flop, il procedimento per la loro analisi e sintesi, anche con riferimento ad alcuni esempi di cui è anche proposta una descrizioni in linguaggio VHDL e mostrati i risultati delle relative simulazioni.

I sistemi digitali si compongono, come visto, di reti logiche che possono essere di tipo combinatorio o sequenziale. Le reti combinatorie sono, come è noto, sistemi "senza memoria", nel senso che esse implementano una funzione booleana (mono, multi valore), di variabili booleane. Le reti sequenziali, siano esse asincrone o sincrone, sono invece reti dotate di memoria che ne determina nel tempo lo *stato*. Tale stato contribuisce, unitamente agli ingressi, alla definizione dei valori delle uscite e dello stato successivo.

Gli elementi di memoria sono stati realizzati nel tempo a partire dalle tecnologie abilitanti di cui si poteva disporre e con cui realizzare l'elemento base che, come detto, è un elemento bistabile. Flip-flop e latch rappresentano, ad oggi, una soluzione dominante per la sintesi di tali tipi di rete e sono diffusamente usati in tutti i progetti di sintesi di apparati digitali.

2 Bistabili: latch e flip-flop

I ciruciti bistabili rappresentano uno dei componenti base per la realizzazione dei latch e dei flip-flop. I bistabili fanno parte della famiglia di circuiti noti come multivibratori. Essi sono circuiti elettronici retroazionati e possono essere di tipo monostabile, astabile e bistabile.

I *multivibratori monostabili* hanno un'uscita che può assumere o un valore di riposo detto "neutro" o un valore "attivo". Il valore attivo dura per una costante di tempo predefinita e caratteristica dello specifico dispositivo. Di fatto il monostabile si comporta da timer, esso mantiene lo stato attivo per un tempo preassegnato e poi ritorna allo stato di riposo dove permane finchè un segnale di controllo in ingresso non lo faccia ricommutare allo stato attivo.

I *multivibratori astabili* sono circuiti elettronici a due stati, ma con una uscita. Essi sono progettati in modo tale da permanere in ciascuno dei due stati per un tempo preassegnato. Tali circuiti sono, pertanto, degli oscillatori il cui periodo e il cui duty cycle sono determinati dalle due costanti di permanenza allo stato 1 e a quello 0. I valori di tali costanti di tempo dipendono dal valore di elementi resistivi e capacitivi del circuito.

I *multivibratori bistabili* sono idealmente circuiti elettronici che possono permanere solo in uno di due stati stabili. Per ciascuno stato le uscite assumono valori (complementari nel senso che se un'uscita assume un potenziale alto, l'altra ne assume quello basso e viceversa) che mantengono indefinitamente fino a che un nuovo segnale di comando ne forza il cambiamento di stato. Uno o più linee di ingresso permettono, quindi, di forzare l'equilibrio raggiunto portando il circuito a operare fra uno stato stabile e l'altro. Tale dispositivo si comporta quindi come una cella di memoria di un bit e per la sua importanza sarà trattato nel dettaglio di seguito.

2.1 Il Multivibratore Bistabile

Il multivibratore bistabile I circuiti bistabili sono realizzati mediante una cella base a due transistori. Il funzionamento di principio del circuito può essere spiegato con riferimento alla fig.2.1. Il dispositivo presenta due ingressi indicati con A e B e due uscite Q e $\overline{Q}$. Un interruttore bipolare, collega l'uno o l'altro degli ingressi collegati alle basi dei transistor a massa. Quando l'interruttore è connesso al punto A, la base del transistor TR1 è connessa a massa e il transistor è in interdizione e la sua uscita Q si porta al valore Vcc (che in logica positiva rappresenta lo stato 1 logico). Di contro il transistor TR2, polarizzato direttamente, si porta in saturazione e presenta all'uscita $\overline{Q}$ un valore di tensione (Vcesat) prossimo a 0 Volt (associato allo stato 0 logico). Viceversa, quando l'interruttore è spostato sulla posizione B, TR2 va in interdizione e TR1 in conduzione, commutando l'uscita Q allo stato 0 e la $\overline{Q}$ a 1. Le uscite Q e $\overline{Q}$, nelle due situazioni di

stabilità, sono l'una negata dell'altra. Durante il transitorio (si veda fig.2.1), a seguito di una commutazione di breve durata τ, le uscite, con continuità, si portano da un valore all'altro, passando per un punto di equilibrio instabile (o metastabile), in cui le relative tensioni sono uguali. Tale punto è critico e se non controllato potrebbe portare a forme di indeterminazione delle uscite con successivi malfunzionamenti di un circuito.

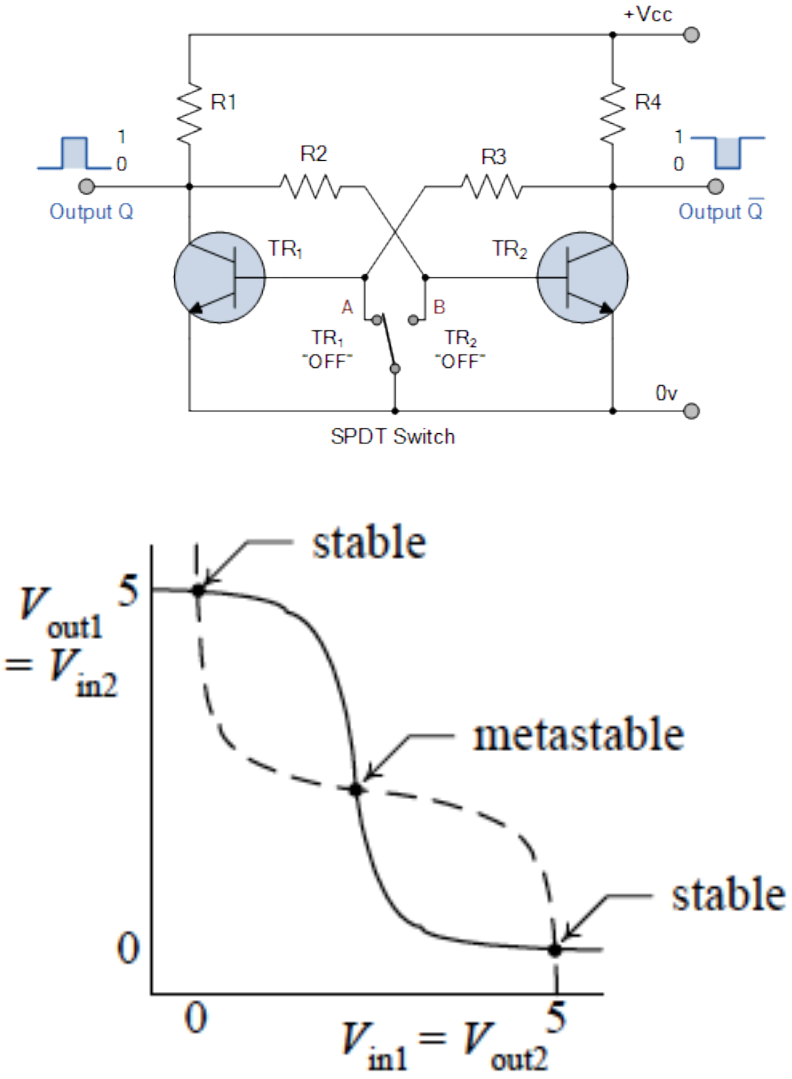


Figura 2.1: Circuito multivibratore bistabile e stati delle uscite

Come si può notare sempre dalla fig.2.1, il bistabile è realizzato controreazionando due invertitori (ciascuno dei due transistor in figura opera da invertitore).

2.2 Bistabili come elementi di memoria

Un elemento di memoria svolge nel tempo la funzione di contenitore di un dato binario (bit). Tale elemento può operare in modo statico o dinamico. Nel modo statico, durante il periodo di funzionamento, la memoria è non volatile (in talune tecnologie, come quella delle memorie magnetiche, la non volatilità resta anche a dispositivo non alimentato e la memoria si dice essere persistente), nel modo dinamico la memoria è volatile nel senso che perde il suo valore nel tempo e si richiede il ricorso a operazioni cicliche di rinfresco del valore (refresh) per mantenerlo.

I primi elementi di memoria sono stati realizzati ricorrendo a tecniche di memorizzazione dei dati dinamiche e basate sull'uso di linee elettriche di ritardo. In esse, la memorizzazione di un bit è effettuata sfruttando l'effetto di persistenza, dovuto alla propagazione del segnale, del valore a esso associato. Una volta immesso un segnale in una linea elettrica di lunghezza L opportuna, questo si propaga nella linea in un tempo di ritardo τ (la propagazione avviene alla velocità della luce nel mezzo e il cui valore dipende dalla lunghezza e dalle proprietà elettriche della linea stessa, ad es. 0,8c). Dopo tale tempo, terminata la propagazione, il segnale raggiunge l'altro estremo della linea posto a una lunghezza L, riflettendosi indietro e subendo effetti di attenuazione per gli effetti resistivi e di degrado per gli effetti capacitivi e induttivi della linea stessa. Occorre pertanto, mediante un apposito meccanismo (realizzato con un circuito elettronico che provvede a rigenerarne forma e livelli elettrici), procedere alla sua reimmissione (una sorta di refresh) nella linea per compiere un nuovo ciclo che, se ripetuto, ne garantisce il mantenimento del valore nel tempo. Tale meccanismo di refresh provvede a ripristinare e mantenere, quindi, nel tempo il valore del segnale memorizzato.

Al posto di una linea di ritardo si può utilizzare, mantenendo lo stesso meccanismo di memorizzazione del dato, un buffer (fig.2.2), avente un ritardo di propagazione (o di trasporto) t_{pd} e in cui l'uscita è collegata in retroazione con l'ingresso. Una volta immesso in esso un valore 0 o 1, questo resterà memorizzato a causa dell'effetto di rigenerazione dinamico che ciclicamente propaga il segnale fra l'uscita e l'ingresso. Lo stato binario è memorizzato, quindi, in cicli di durata t_{pd}, il tempo impiegato dal segnale per propagarsi fra l'ingresso e l'uscita del circuito (si fa osservare che allo stato corrente della tecnoclogia dei componenti, il ritardo t_{pd} è dell'ordine di 1 ns o meno).

Ovviamente, tale soluzione è di principio e di scarsa utilità sia per la difficoltà che si incontra nel forzare un nuovo valore da memorizzare nel buffer sia per i consumi del circuito che, per mantenere memorizzato un bit, deve essere mantenuto sempre alimentato.

Per la realizzazione di una cella di memoria elementare si preferisce, pertanto, ricorrere ai circuiti bistabili, realizzati come visto con circuiti elettronici detti multivibratori bistabili o con circuiti elettronici a riempimento-svuotamento di carica (come ad esempio nelle celle a transistor MOS, in cui lo stato booleano è associato alla presenza o meno di una certa quantità di carica elettrica in un dispositivo).

Un dispositivo bistabile realizza, quindi, fintantochè il suo circuito è alimentato e funzionante, una memoria di un bit, mantenendo stabili nel tempo i valori elettrici associati

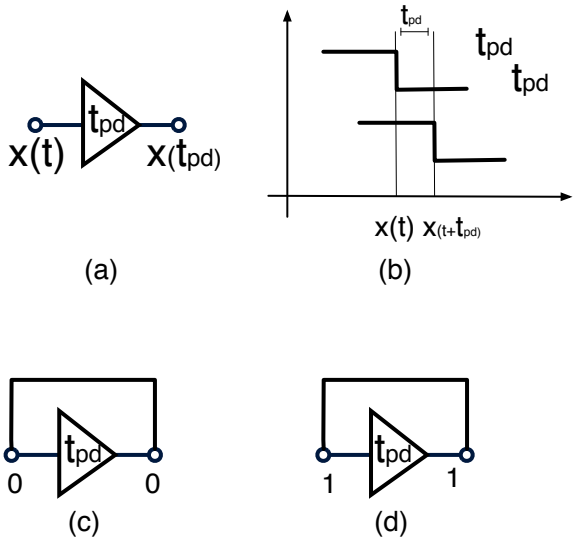


Figura 2.2: Buffer con ritardo per memorizzare valori booleani

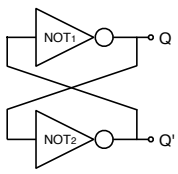


Figura 2.3: Circuito bistabile

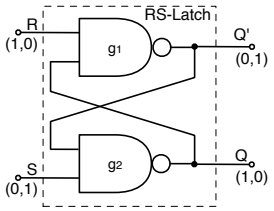


Figura 2.4: Bistabile realizzato con due porte NAND

a ciascuno dei suoi due stati logici.

La fig.2.3, mostra un bistabile di principio realizzato logicamente mediante due inverter (che operano da amplificatori invertitori). Le due uscite in tensione V_H e V_L ne rappresentano rispettivamente gli stati logici 1 e 0.

Il funzionamento del bistabile può essere descritto concettualmente come segue: supponendo che al tempo t_0 sia Q=V_H, essendo Q collegato all'ingresso dell'inverter NOT_2, l'uscita da tale inverter si porterà al valore Q'=$V_L$che, a sua volta, reazionato all'ingresso del NOT1, genererà sulla sua uscita il valore Q=V_H, chiudendo così l'anello di reazione in modo consistente fra i valori degli ingressi e delle uscite e soddisfacendo il vincolo del flip-flop Q=Not Q'.

Il bistabile sopra descritto è dotato sole delle due uscite Q e Q', e non posside ingressi. Esso è solo di principio in quanto in tal caso i valori di Q e Q' manterrebbero per tutto il periodo di funzionamento l'uno o l'altro dei due valori di regime raggiunto in modo non determinabile dopo il transitorio che segue l'alimentazione del circuito. Per rendere concretamente utilizzabile tale bistabile, occorre inserire due ulteriori linee di ingresso tramite cui poter controllare i valori delle uscite e, quindi, la commutazione dello stato.

Ciò può ottenersi sostituendo i due inverter con due porte NAND o con due porte NOR così come illustrato in fig.2.4. Se si usano le porte NAND, i segnali di input, per rispettare le equazioni logiche del flip-flop, devono operare in logica negativa se NOR in logica positiva (per chiarezza si indicheranno con R, S gli ingressi codificati in logica 1-attiva e con R', S', quelli in logica 0-attiva).

Tale tipo di circuito elettronico bistabile può essere studiato anche come una semplice macchina asincrona, operante secondo il modello fondamentale di Huffman (per tale motivo esso è noto come latch RS fondamentale). Per tale macchina, gli ingressi in grado di produrre il cambiamento di stato sono detti di Set (S), utilizzato per forzare lo stato Q a 1, e di Reset (R), per forzarlo a 0. Se essi assumono contemporaneamente il valore neutro (il valore non attivo), lo stato resta invariato, se assumono invece il valore attivo (condizione logica proibita dalla specifica del latch RS), le uscite del bistabile assumono entrambe il medesimo valore V_H, se esso è realizzato con porte NOR, o il valore V_L se con NAND, violando quindi la condizione logica Q = Not Q'. Volendo impedire tale anomalia nel funzionamento fisico del latch rispetto a quello logico, occorre pervenire a una realizzazione in grado di vincolare gli ingressi S e R a non essere simultaneamente

attivi.

Il bistabile, memoria elementare di 1 bit, rappresenta l'elemento base per la sintesi di componenti di memoria più complessi realizzati come macchine sincrone o asincrone.

Come rete sequenziale asincrona, il suo funzionamento può essere descritto dalla:

- *equazione caratteristica,* che esprime lo stato prossimo in funzione dello stato corrente e dello stato degli ingressi;
- *tabella delle transizioni di stato,* che riporta nelle righe lo stato corrente, nelle colonne gli ingressi e negli incroci lo stato prossimo;
- *tabella invertita in forma compatta* che riporta nelle righe gli ingressi e nelle colonne lo stato prossimo;
- *tabella delle eccitazioni* che riporta nelle righe le combinazioni possibili di stato corrente-stato prossimo e nelle colonne i valori degli ingressi che permettono la specifica transizione di stato.

I bistabili possono essere classificati in base al numero e alle caratteristiche dei segnali di ingresso e dal fatto di operare in modo asincrono o sincrono. Si indicano con *latch* i bistabili che se abilitati commutano il loro stato al variare degli ingressi con continuità e restano insensibili alle variazioni degli ingressi quando disabilitati; con *flip-flop* i bistabili che commutano lo stato solo in ben precisi istanti di tempo coincidenti con il fronte di salita o di discesa del segnale di abilitazione e restano inattivi quando l'abilitazione assume i valore a livello 0 o 1.

La principale differenza fra latch e flip-flop sta quindi nel fatto che in un latch abilitato a operare lo stato evolve con l'evoluzione degli ingressi, esso cambia il suo stato al variare dei suoi ingressi e, quando disabilitato, mantiene l'ultimo stato memorizzato. Di contro, un flip-flop cambia stato soltanto sul fronte attivo del suo segnale di abilitazione, e cioé in un preciso istante di tempo e non in una finestra temporale, così come avviene per il latch. Se il fronte è quello caratterizzato dalla variazione 0->1 di un segnale di *Enable*, il flip-flop è detto sensibile (o attivo) sul fronte di salita (rising edge), se quello caratterizzato dalla variazione 1->0, è detto sensibile al fronte di discesa (falling edge). Passato il fronte attivo, il flip-flop diventa insensibile a qualunque variazione dei suoi ingressi e mantiene costante il suo stato.

I bistabili possono anche essere classificati, in base al numero e al tipo degli ingressi, in flip-flop o latch di tipo RS, D, JK e T.

Si fa osservare, infine, che per la sintesi di reti sequenziali sincrone, è preferibile utilizzare, come elementi di memoria nelle linee di retroazione, i flip-flop e non i latch, in quanto essi consentono di sincronizzare il trasferimento di un nuovo stato in un ben preciso istante di tempo, coincidente con uno dei due fronti di un segnale di abilitazione (che se regolare è detto clock).

2.3 Analisi e progettazione del latch RS fondamentale

Il latch RS di seguito descritto è una rete sequenziale asincrona operante in modo fondamentale e cioè una macchina con sequenze di ingresso a livello in cui gli ingressi sono vincolati a cambiare uno alla volta e in cui una sola uscita può cambiare a seguito di un passaggio di stato. La specifica verbale del latch RS fondamentale è la seguente:

> *Una macchina asincrona con due ingressi S e R (Set e Reset) e due uscite Q e Q'. L'ingresso attivo S pone lo stato di Q al valore attivo, l'ingresso attivo R al valore non attivo, l'ingresso neutro (00 per i segnali 1-attivi e 11 per quelli 0-attivi) lascia lo stato del latch invariato. C'è il vincolo sulle uscite di essere* $Q = \overline{Q'}$*, vincolo che si traduce nel vincolo sugli ingressi R ed S di non assumere contemporaneamente il valore attivo. Nel caso di segnali 1-attivi tale vincolo è dato da R·S=0, nel caso di segnali 0-attivi dalla condizione duale R+S=1.*

Da tale descrizione funzionale si può dedurre direttamente, riferendosi ad esempio a segnali 1-attivi, l'equazione di stato logica del latch di seguito riportata (si è indicato con Q_p il valore corrente della variabile di stato e con Q il valore successivo):

$$Q = S + \overline{R}Q_p \text{ con vincolo } R \cdot S = 0$$

Tale equazione asserisce che lo stato Q assume il valore vero quando il segnale di set è vero o quando lo stato precedente è vero e il segnale di reset è falso.

In modo analogo, per segnali 0-attivi, Q assume il valore vero quando il segnale di set è falso o quando lo stato precedente è vero e il segnale di reset è vero, così come indicato dall'equazione di stato seguente:

$$Q = S' + RQ_p \text{ con il vincolo } R + S = 1$$

Il latch RS con ingressi 0-attivi, realizzato come si vedrà di seguito con porte NAND, è del tutto analogo nel funzionamento a quello con ingressi 1-attivi, realizzato con porte NOR. La differenza sta solo nel valore dello stato neutro degli ingressi e nei valori che esprimono il vincolo non ammessi (R=S=1 è la configurazione dello stato degli ingressi non ammesso), il valore attivo R=0 forza l'uscita Q=1 e quello attivo S=1 la forza a Q=0. Tale comportamento è il duale del latch realizzato a porte NOR.

Il latch, rappresentato come macchina di Moore, può essere descritto dalla tabella caratteristica estesa o da quella compatta tab.2.1 e dalla tabella delle transizioni di stato (tabella caratteristica) spesso utilizzata per la sintesi delle reti sequenziali 2.3.

Stante la semplicità del circuito, una prima sintesi diretta può essere effettuata a partire dalla tabella delle transizioni di stato ridotta di tab.2.3 o dalle equazioni logiche del latch, che porta alla realizzazione del cosidetto latch dinamico in cui lo stato è codificato con un'unica variabile. Tale tipo di latch ha il vantaggio di non presentare alee combinatorie, e quindi corse, ma presenta lo svantaggio, avendo un solo segnale di stato-uscita Q, di

R	S	Q_p	Q
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	-
1	1	1	-

R	S	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
0	1	1	0
1	0	0	1
1	1	-	-

Tabella 2.1: Tabella caratteristica in forma estesa e compatta per il latch RS

	Q_p/RS	00	01	11	10	Q (Q')
Stato di reset	s_0	**s_0**	s_1	-	s'_0	1(0)
	s'_0	s_0	s_1	-	**s'_0**	0(1)
Stato di set	s_1	s'_1	**s_1**	-	s'_0	0(1)
	s_1	**s'_1**	s_1	-	s'_0	0(1)

	Q_p/RS	00	01	11	10
Stato di reset	s_0	-	s_1	**s_0**	s'_0
	s'_0	-	s_1	s_0	**s'_0**
Stato di set	s_1	-	**s_1**	s'_1	s'_0
	s_1	-	s_1	**s'_1**	s'_0

Tabella 2.2: Tabella primitiva del latch RS per segnali 1-attivi (realizzazione con porte NOR) e 0-attivi (realizzazione con porte NAND)

dover derivare quello complementare Q' da Q mediante l'uso di un inverter, cosa questa che rende asimmetrica l'architettura del latch e che ritarda la presentazione del nuovo valore di Q' rispetto a Q. Tale soluzione è pertanto solo di principio e raramente utilizzata.

La sintesi del latch RS può essere fatta sistematicamente, utilizzando, a partire dalle tabelle di transizione di stato, il classico procedimento di sintesi delle macchine asincrone di Paull-Hunger che prevede la minimizzazione dello spazio degli stati e un'apportuna codifica dei segnali di ingresso, stato e uscita.

Si fa osservare la natura *asincrona impulsiva* di tale macchina. Essa, infatti, opera solo con sequenze di ingresso intervallate dallo stato neutro. Ad esempio, nel caso del latch a NAND, il cui stato neutro è dato da R=S=1, si possono presentare sequenze del tipo "11-10-11-01...". Come detto non è ammessa la sequenza proibita R=S=0 che, se applicata, porterebbe a cambiamenti di stato non definiti.

Q_p/RS	00	01	11	10	Q (Q')
S_0	**S_0**	S_1	-	**S_0**	1(0)
S_1	**S_1**	**S_1**	-	S_0	0(1)

Q_p/RS	00	01	11	10	Q,Q'
S_0	-	S_1	**S_0**	**S_0**	0,1
S_1	-	**S_1**	**S_1**	S_0	1,0

Tabella 2.3: Tabella delle transizioni di stato del latch RS fondamentale per segnali 1-attivi (realizzazione con porte NOR) e 0-attivi (realizzazione con porte NAND)

$\mathbf{Q_p/RS}$	00	01	11	10	Q,Q'
S_0	$\mathbf{S_0}$	S	-	$\mathbf{S_0}$	10
S	-	S_1	-	S_0	00
S_1	$\mathbf{S_1}$	$\mathbf{S_1}$	-	S	01

=>

$\mathbf{y_1y_2/RS}$	00	01	11	10
10	10	00	-	10
00	-	01	-	10
01	01	01	-	00

Figura 2.5: Tabella delle transizioni e tabella in codice con aggiunta dello stato instabile S per eliminare la corsa (caso di un latch a NOR)

$\mathbf{y_1}$=>

$\mathbf{y_1y_2/RS}$	00	01	11	10
00	**-**		-	**1**
01			-	
11	-	-	-	-
10	**1**		-	**1**

y2=>

$\mathbf{y_1y_2/RS}$	00	01	11	10
00	**-**	**1**	-	
01	**1**	**1**	-	
11	-	-	-	-
10			-	

Figura 2.6: Mappe di Karnaugh per y_1 e y_2(sintesi con NOR)

Onde pervenire alla realizzazione del latch simmetrico con le due uscite Q e Q', occorre utilizzare una codifica con due variabili di stato, y_1 e y_2, cosa che porta, come si vedrà, alla realizzazione del latch con due porte di tipo NOR (se gli ingressi R e S sono 1-attivi) o di tipo NAND (se gli ingressi sono 0-attivi).

In particolare, la codifica dello stato S_0 con y_1= 1 e y_2= 0 e dello stato S_1 con y_1= 0 e y_2= 1, se garantisce che le variabili di stato y_1 e y_2 coincidano con quelle delle uscite Q e Q', nel contempo genera una corsa nella transizione da uno stato all'altro, stante la presenza dell'alea 01<–>10 evidenziata in tabella e dal cambiamento simultaneo dei codici delle due variabili di stato da 10 a 01 o viceversa.

Per eliminare tale alea, e l'eventuale corsa che ne deriverebbe, si introduce un nuovo stato *instabile*, denotato con S, che può essere codificato con una delle due configurazioni libere y_1= 0 e y_2= 0 oppure y_1= 1 e y_2= 1. Se si codifica con y_1=0 e y_2=0 si perviene alla sintesi di un latch a NOR, se con y_1=1 e y_2=1, di un latch a NAND, così come di seguito mostrato:

Dalla tabella in codice di fig.2.5 si derivano le due mappe di Karnaugh per y_1 e y_2 di cui alla fig.2.6. La sintesi della rete combinatoria è effettuata a partire da tali mappe, selezionando per y_1 e per y_2 i primi implicanti marcati in grassetto e da cui deriva, come detto, la sintesi del latch 1-attivo con porte NOR (si veda fig.2.9).

$Y_1 = \overline{S} \cdot \overline{y}_2 = S \downarrow y_2$

$Y_2 = \overline{R} \cdot \overline{y}_1 = R \downarrow y_1$

Se si codifica, invece, lo stato instabile con y_1=1 e y_2=1, si perviene, in modo analogo, alle seguenti equazioni che portano alla sintesi del latch 0-attivo con porte NAND (2.7 e 2.8):

$\mathbf{Q_p/RS}$	00	01	11	10	Q,Q'
S_0	-	S	$\mathbf{S_0}$	$\mathbf{S_0}$	10
S	-	S_1	-	S_0	00
S_1	-	$\mathbf{S_1}$	$\mathbf{S_1}$	S	01

=>

$\mathbf{y_1y_2/RS}$	00	01	11	10
10	-	11	10	10
11	-	01	-	10
01	-	01	01	11

Figura 2.7: Tabella delle transizioni e tabella in codice con incluso lo stato instabile per eliminare la corsa (latch a NAND)

$\mathbf{y_1}$=>

$\mathbf{y_1y_2/RS}$	00	01	11	10
00	-	-	-	**-**
01	-			**1**
11	-		-	**1**
10	**-**	**1**	**1**	**1**

y2=>

$\mathbf{y_1y_2/RS}$	00	01	11	10
00	-	-	-	-
01	-	**1**	**1**	**1**
11	-	**1**	-	
10	-	**1**		

Figura 2.8: Mappe di Karnaugh per y_1 e y_2(sintesi con NAND)

$Y_1 = \overline{y}_2 + \overline{S} = y_2 \uparrow S$
$Y_2 = \overline{y}_1 + \overline{R} = y_1 \uparrow R$

Stante la semplicità di tale latch, se ne può effettuare la sintesi per tramite delle sue *equazioni caratteristiche* (sia per ingressi 1-attivi sia 0-attivi) ottenute minimizzando le mappe di Karnaugh di tab.2.4 derivate dalla tabella in codice riportate in fig.2.7:

$Q = S + \overline{R}Q_p$ con vincolo $R \cdot S = 0$ per la 1-attiva e
$Q = \overline{S} + RQ_p$ con il vincolo $R + S = 1$ per la 0-attiva

Da tali espressioni, dopo una semplice manipolazione algebrica, si ottengono le seguenti due forme in soli NOR e in soli NAND corrispondenti ai circuiti del latch di fig.2.9:

$$Q = S + \overline{R}Q_p = \overline{\overline{S + \overline{R}Q_p}} = \overline{\overline{S} \cdot R\overline{Q}_p} = S \downarrow (R \downarrow \overline{Q}_p) \text{ con vincolo } R \cdot S = 0$$

$$Q = \overline{S} + RQ_p = \overline{\overline{S + RQ_p}} = \overline{S \cdot \overline{RQ_p}} = S \uparrow (R \uparrow Q_p) \text{ con il vincolo } R + S = 1$$

Come si può osservare, le espressioni finali sono, ovviamente, identiche a quelle derivate precedentemente con la procedura di sintesi e a partire dalla descrizione verbale del funzionamento del due latch.

$Q_p \backslash SR$	00	01	11	10	$Q_p \backslash SR$	00	01	11	10
0	0	0	-	1	0	-	1	0	0
1	1	0	-	1	1	-	1	1	0

Tabella 2.4: Mappa di Karnaugh per il calcolo dell'equazione caratteristiche del latch RS con ingressi 1-attivi (a sinistra) e 0-attivi (a destra)

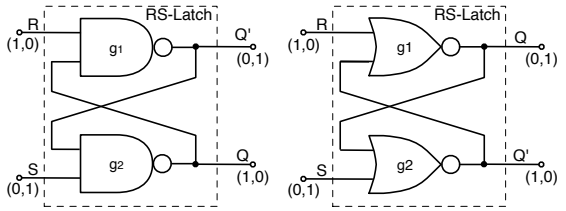


Figura 2.9: Latch RS simmetrico

Si fa osservare che il vincolo R=S=1 per i latch 1-attivi e R=S=0 per quelli 0-attivi presente nella specifica del latch RS, deve essere posto e rispettato nel contesto operativo in cui il latch si trova a operare. Ciò in quanto se gli ingressi S e R assumessero un tale stato proibito, il circuito, operando in modo improprio, forzerebbe lo stato delle uscite a uno stesso valore (0 o 1). In particolare, con riferimento a un latch a NAND (0-attivo) con R=S=1 come stato neutro, ponendo gli ingressi R=S=0, lo stato delle uscite si porterebbe a Q=Q'=1, mentre per una realizzazione a NOR, con R=S=0 come stato neutro, si avrebbe, a fronte di R=S=1 (1-attivo), uno stato delle uscite forzato a Q=Q'=0.

Tale condizione, non ammessa, genererebbe in un latch reale una situazione di non determinismo sulle uscite. Ciò è dovuto all'effetto dei ritardi nelle linee di retroazione che per un latch reale sono certamente fra loro differenti In un latch ideale, dove tali ritardi possono supporsi essere uguali, tale condizione proibita genererebbe un fenomeno oscillatorio delle uscite del latch all'atto del riposizionamento degli ingressi allo stato neutro, con sequenza di transizione 00->11 o 11->00. Tale comportamento del circuito ideale è facilmente verificabile anche nella tabella delle transizioni di stato.

Per verificare concretamente tale comportamento, sono riportati di seguito i risultati della simulazione di un latch RS . Il latch è stato disegnato mediante lo schematic disponibile nell'ambiente ISE di Xilinx e poi simulato in modo behavioral e nei modi non-behavioral post-translation, post-mapping, post-place-and-route.

I dettagli di tali modi di simulazione sono riportati in appendice B. Il test bench per verificare il latch fornisce agli ingressi q e nq uno specifico pattern per ciascuno di essi. Tale pattern è generato mediante il seguente processo in VHDL:

```
tb : PROCESS
  BEGIN
       s <= '0', '1' after 20 ns, '0' after 40 ns,'1' after 65 ns,
                '0' after 80 ns, '1' after 100 ns;
       r <= '0', '1' after 45 ns, '0' after 80 ns,'1' after 90 ns;
   WAIT; -- attesa infinita

  END PROCESS;
```

In fig.2.10 è riportato lo schematico del latch disegnato con l'apposito editor dell'ambiente ISE.

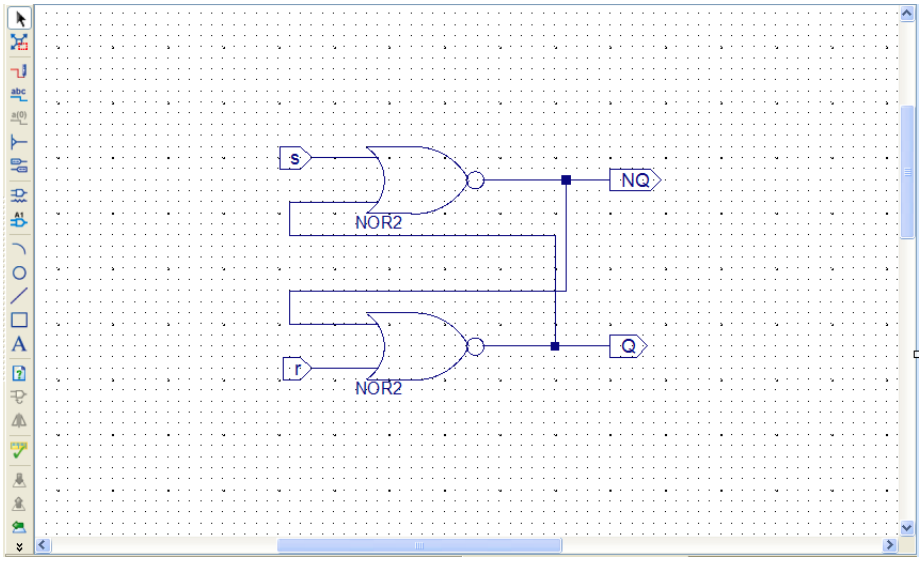


Figura 2.10: Schematico del latch RS

Nella simulazione behavioral, fig.2.11, le porte logiche sono considerate ideali e istantanee nel loro funzionamento per cui passando dalla condizione R=S=1 a quella R=S=0 si generano eventi oscillatori tutti posti all'istante di commutazione dei due segnali che producono equivalenti cicli di delta cycle. Tali cicli non sono rappresentabili dal simulatore in quanto il tempo di simulazione per essi non avanza. Ciò, quindi, genera una segnalazione di errore del simulatore per raggiunto limite di iterazioni[1]

La simulazione post-translate, introduce nella descrizione del circuito da simulare componenti logici di una libreria Xilinx ma sempre ideali nei tempi di commutazione considerati istantanei. I diagramma di fig.2.12 è pertanto uguale a quello precedente e si manifesta, ovviamente, lo stesso errore di raggiungimento del limite di iterazioni delta cicle.

La simulazione post-map, di cui alla fig.2.13, avviene dopo che nel processo di sintesi si sono inseriti componenti logici con gli associati ritardi ma senza tenere conto dei ritardi delle connessioni fra essi considerate istantanee. Tale situazione in cui è mantenuta la simmetria delle porte è l'unica che permette di evidenziare nel diagramma temporale prodotto dal simulatore il fenomeno delle oscillazioni sulle uscite all'atto della transizione degli ingressi R e S a 0 logico. Si fa notare che tale caso è ideale e si manifesta in quanto la rete di connessione fra i componenti opera a ritardo nullo.

[1]ERROR: at 80 ns(10000): Iteration limit 10000 is reached. Possible zero delay oscillation detected where simulation can not advance in time because signals can not resolve to a stable value in File "E:/Ambienti-ASE/flip-flop/Latch_SR/latch_rs_nor/latch_rs_nor/RS_nor.vhf" Line 48. Please correct this code in order to advance past the current simulation time.

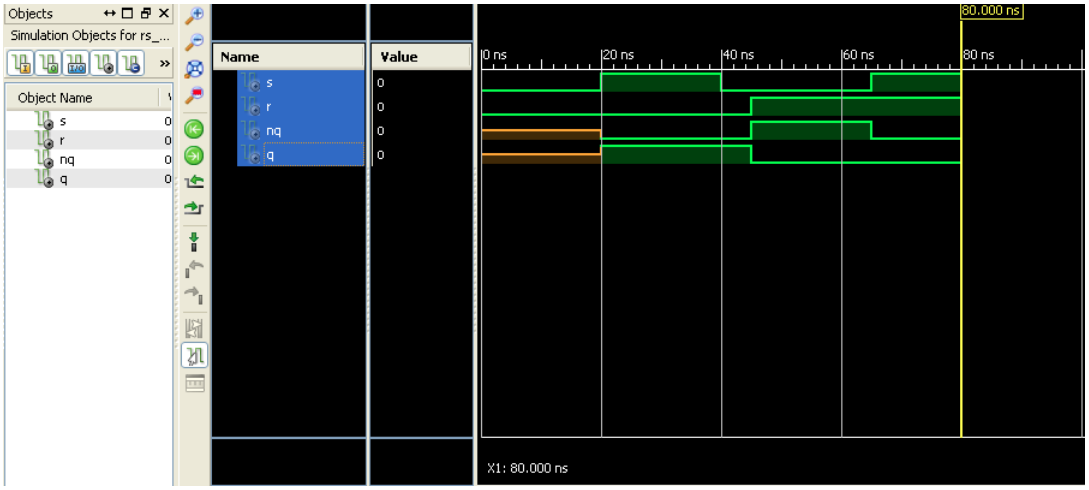


Figura 2.11: Simulazione del latch RS Behavioural

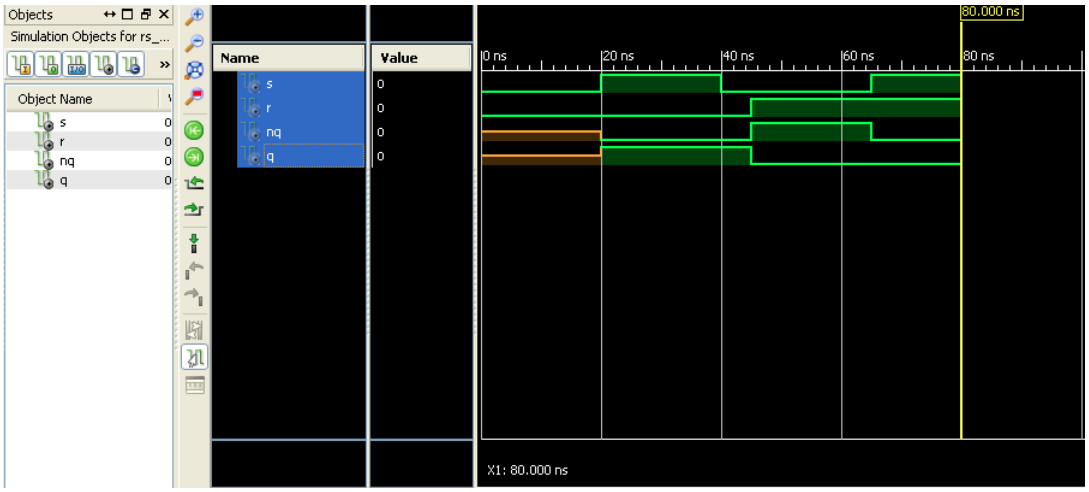


Figura 2.12: Simulazione del latch RS Post-Translate model

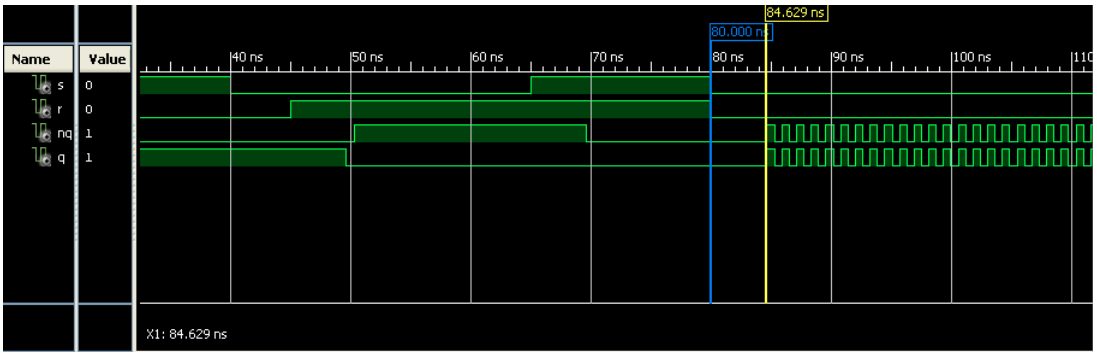


Figura 2.13: Simulazione del latch RS Post-Map

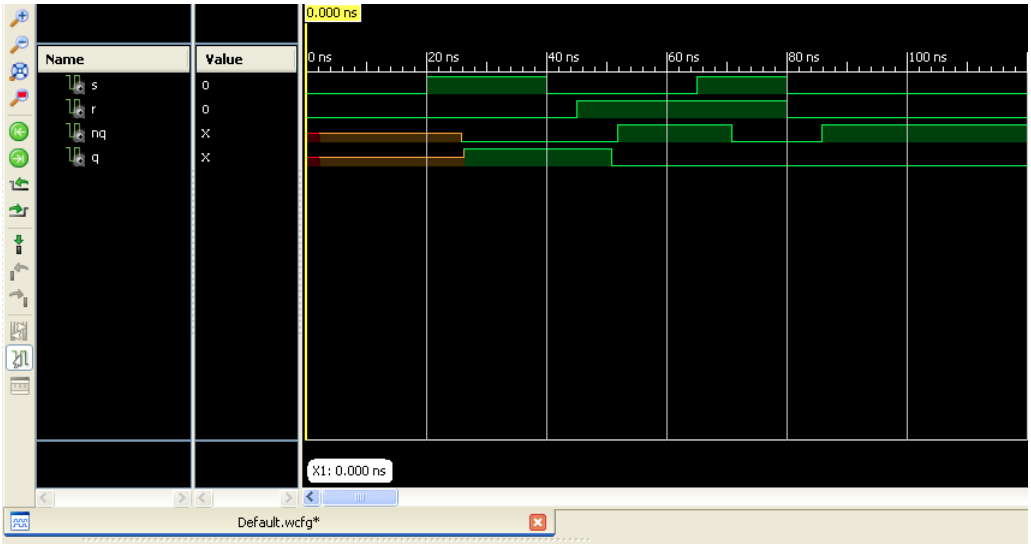


Figura 2.14: Simulazione del latch RS Post-Route

La simulazione post-route, è la più dettagliata e vicina a quello che è il comportamento reale del circuito sintetizzato sul dispositivo FPGA selezionato. In essa si tiene anche conto dei ritardi delle connessioni dipendenti dallo specifico routing effettuato nella fase di mapping tecnologico. Ciò inserisce certamente delle asimmetrie nel circuito che determinano lo stato delle uscite al raggiungimento dello stato R=S=0. In fig.2.14 si può, infatti, notare che lo stato delle uscite si pone a q=0 e nq=1.

2.4 Latch RS abilitato

Il latch RS abilitato è derivato dal latch RS di cui ne eredita il comportamento quando il segnale di abilitazione E (Enable) è attivo mentre conserva l'ultimo stato memorizzato, e quindi è insensibile alle variazioni degli ingressi R e S, quando E è neutro. In fig.2.15 il valore attivo di E è quello alto, mentre il valore basso, forza le uscite delle due porte And a 0-logico, valore neutro questo per il latch RS a valle del circuito. La progettazione di tale latch può essere effettuata considerando tale sistema composto da un latch RS fondamentale e da una rete di abilitazione composta dalle due porte And e dal segnale abilitante E.

2.5 Latch trigger T

Il latch T (abbreviazione di Toggle che in inglese significa "cambio di stato in un funzionamento", a parte gli eventuali ingressi di Preset e Set asincroni utilizzati per forzare l'uscita Q rispettivamente allo stato falso o vero, ha un unico segnale di ingresso T (se-

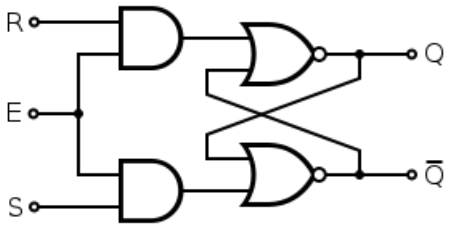


Figura 2.15: Latch RS abilitato

gnale di trigger) che se attivo fa commutare stato del latch da falso a vero e viceversa. Il latch T (e, in particolar modo la versione flip-flop T) è comunemente usato come circuito divisore di frequenza e, quindi, impiegato nei contatori. In tal senso la versione correntemente utilizzata introduce un secondo ingresso detto di clock (clk) in AND con l'ingresso T. In tale configurazione, se l'ingresso T=0, il latch trigger mantiene lo stato precedente, se T=1, il latch segue il comando proveniente dall'ingresso clk che lo fa commutare, per cui l'eventuale segnale di clk in ingresso è riportato in uscita a frequenza dimezzata.

Come macchina asincrona operante in modo fondamentale, il latch T può essere sintetizzato a partire da un latch base e da una rete logica combinatoria posta a monte. Il valore neutro di T lascia invariato lo stato del latch (nel caso riportato in figura di latch a NOR, lo stato neutro è lo zero logico per cui il valore T=0 genera la condizione R=S=0 neutro per il latch RS posto valle), mentre quello attivo, abilitando la retroazione dell'uscita Q sull'ingresso R e dell'uscita negata Q' sull'ingresso K, fa commutare il latch, sempre che si rispettino i vincoli della macchina fondamentale asincrona che opera in tal casa come macchina a impulsi (0-1-0). La durata dell'impulso (i tempo di permanenza nello stato attivo di T) deve essere compresa fra un Tmin e un Tmax. Tmin deve essere sufficientemente grande per consentire al circuito di commutare, Tmax sufficientemente piccolo per evitare effetti di ulteriori commutazioni che, per tale tipo di reti, portano a un'instabilita che si manifesta con un comportamento oscillatorio delle due uscite Q e Qn.

L'equazione caratteristica del latch T è data da: Q = T'Qp + TQp' = T $\oplus$ Qp, avendo indicato con Q lo stato prossimo e con Qp lo stato precedente.

2.6 Latch JK

Il latch JK fondamentale è simile nel funzionamento a quello SR ove J corrisponde a S e K a R. Esso, se si pone J=K=1, si comporta però da latch trigger (si ricorda che per il latch RS a Nor, R=S=1 è la configurazione non ammessa. Il latch JK è molto simile come architettura al latch T. Così come mostrato in fig.2.17 esso, al posto di un unico segnale T di ingresso collegato alle due porte AND, presenta due ingressi separati di J e K ciascuno diretto a una sola porta AND. Analogamente al latch T, sulle due porte AND sono reazionate le due uscite Q e Q' (Q verso l'ingresso di reset K e l'uscita Q'

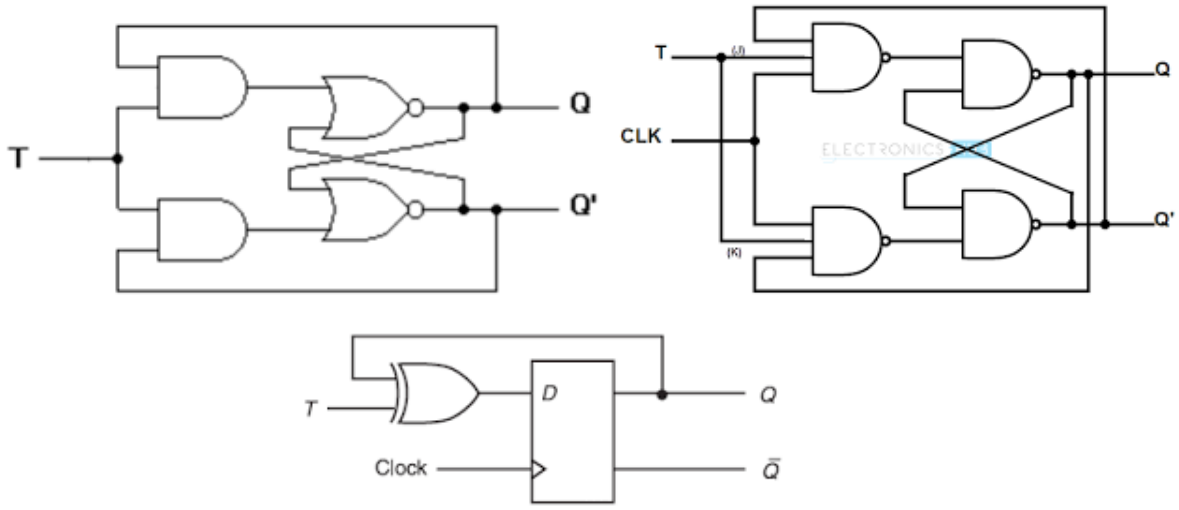


Figura 2.16: Latch T fondamentale, T abilitato e T derivato da latch D

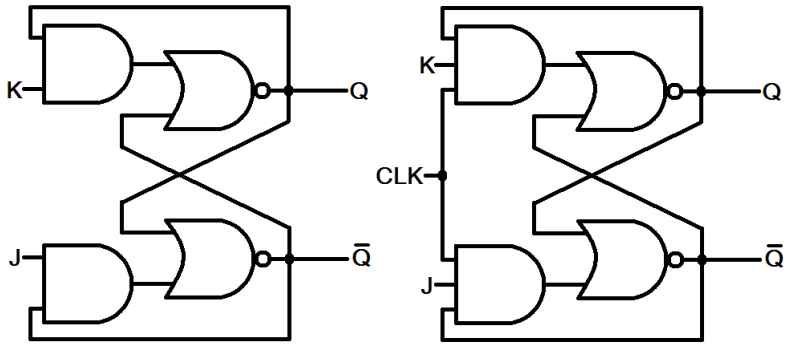


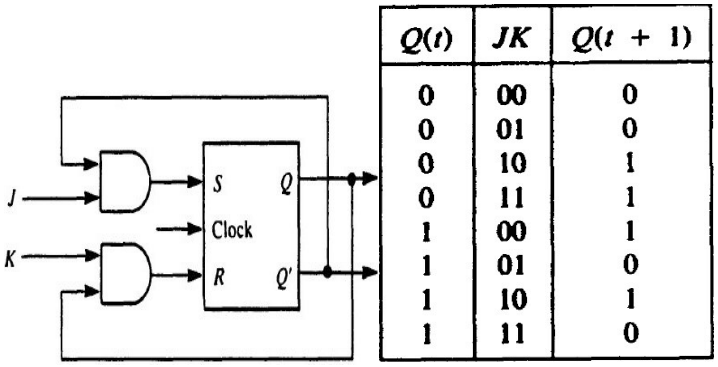
Figura 2.17: Latch JK fondamentale e JK abilitato

verso quello di set J) del latch RS. A destra della fig.2.17 è anche riportata la versione con clock clk. Il segnale clk, quando attivo (e cioè allo stato alto per la and) abilita la commutazione del latch, quando neutro, lo disabilita mantenendo memorizzato lo stato precedente.

La tabella di fig.2.18 descrive il funzionamento del JK con clock.

I bistabili JK commerciali, a differenza dello schema precedente hanno anche due ingressi detti di Preset e Clear, come mostrato in fig.2.19, utilizzati per inizializzare il dispositivo forzandolo in uno dei due stati possibili.

2.6.1 Simulazione del latch J-K



Q(t)	JK	Q(t + 1)
0	00	0
0	01	0
0	10	1
0	11	1
1	00	1
1	01	0
1	10	1
1	11	0

Figura 2.18: Schema di un bistabile JK

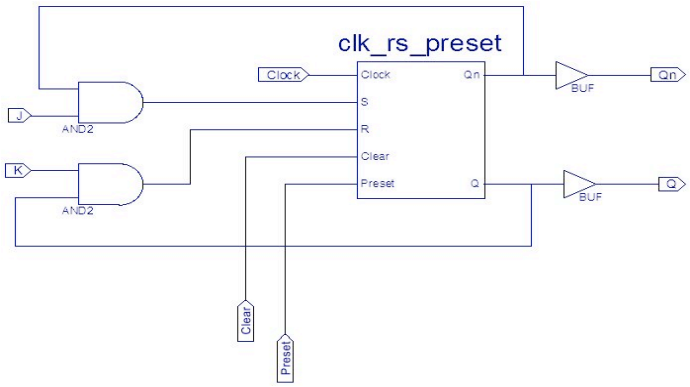


Figura 2.19: Schema di un flip-flop JK commerciale derivato da un RS

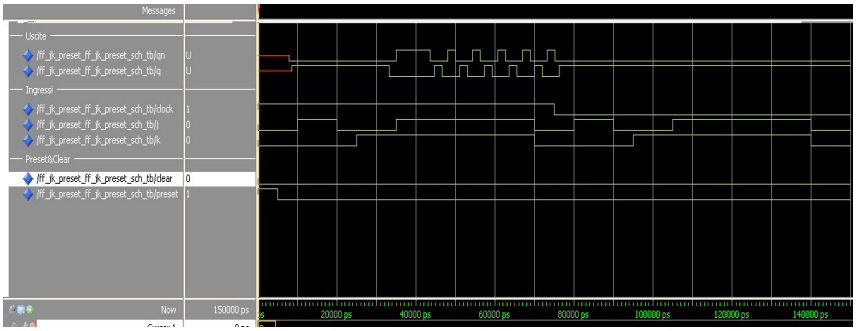


Figura 2.20:

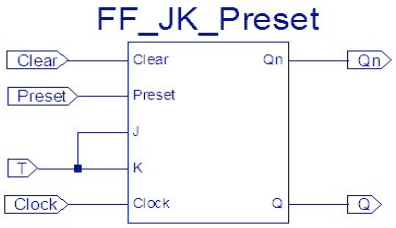


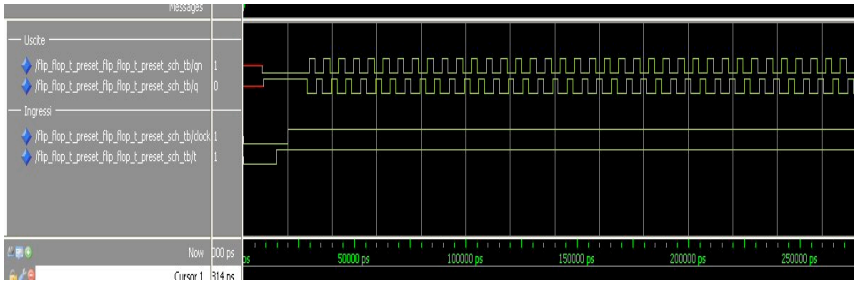
Figura 2.21: Latch JK usato da Trigger

```
Preset <= '1','0' after 5 ns;
Clear <= '0';
Clock <= '1','0' after 75 ns;
J <= '0', '1' after 10 ns, '0' after 20 ns,'1' after 35 ns,'0' after 70 ns,'1'
K <= '0', '1' after 25 ns, '0' after 70 ns,'1' after 95 ns, '0' after 140 ns;
```

Ottenendo:

Dalla simulazione si può notare come il comportamento sia del tutto analogo a quello del latch SR, tranne che nel caso in J = K = 1, in questo caso infatti le uscite del JK commutano continuamente per un numero di volte che dipende dalla durata per cui i segnali J e K permangono alti. È estremamente difficile riuscire ad ottenere un'unica commutazione, questo avverrebbe infatti, solo se si riuscisse a far permanere i segnali J e K pari ad 1 per un tempo abbastanza grande affinché si possa avere un cambiamento nel circuito, ma abbastanza piccolo da non comportare più di una commutazione, quest'intervallo è di pochi nanosecondi, il che rende difficile, se non impossibile, utilizzarlo in questo modo.

Dal JK si può derivare il latch T semplicemente collegando i due ingressi J e K in un

Figura 2.22: 

unico ingresso T (fig.2.21).

In questo caso la tabella di verità diventa:

T	**Q**	**Q'**
0	Qp	Qp'
1	Qp'	Qp

Dove per Qp si indica il valore precedente dell'uscita.

Il circuito così ottenuto può essere simulato post-route con le seguenti waveform

T <= '0', '1' after 15 ns; Clock <= '0','1' after 20 ns; Preset <= '1', '0' after 5 ns; Clear <= '0';

per ottenere:

Che mette ancora più in evidenza il problema prima citato, versioni funzionanti di questo flip flop si ottengono a partire da JK Edge Triggered o Master-Slave

2.7 Analisi e sintesi di un flip flop D a variazione sul fronte (edge triggered)

Il flip flop *D edge triggered* è un dispositivo bistabile con due ingressi, di cui uno D (dato) e uno clk (clock), e due uscite Q e Q'. Esso effettua la transizione di stato all'occorrenza di una variazione sul fronte di salita (o su quello di discesa) del segnale di clock e dello stato degli ingressi in quell'istante. Tale dispositivo può essere analizzato e progettato o con metodi elettronici, come un circuito elettronico digitale con retroazione, o con i tradizionali metodi di sintesi usati per le macchine sequenziali asincrone o come un sistema di reti sequenziali composto da tre sottosistemi latch di tipo RS.

Di seguito sono roportate l'analisi e la sintesi del flip-flop D edge triggered composto da tre latch RS fondamentali (realizzati con porte NAND, se il flip flop è attivo sul fronte di salita del clock, con porte NOR, se su quello di discesa).

2.7.1 Analisi del funzionamento

Per l'analisi del funzionamento del flip-flop D al variare dei due segnali di ingresso clk e D ci si riferirà allo schema a porte NOR, sensibile ai fronti di discesa del clock, è riportato in fig.2.23. Esso impiega solo 6 porte logiche per un totale di 26 transistor, a differenza

dalle 11 porte (con 38 transistor) richieste per la sintesi di un analogo flip-flop operante secondo il modello master-slave. Le sei porte NOR realizzano tre latch RS fondamentali, indicati come *latch1*, *latch2* e *latch3* e fra loro interconnessi.

Si ricorda che una porta NOR è 0-attiva, essa presenta un'uscita alta solo quando i suoi ingressi sono entrambi bassi, negli altri casi la porta fornisce l'uscita bassa (la NOR riconosce, quindi, lo stato in cui tutti i suoi ingressi sono bassi), per cui se si pone un valore logico '0' a uno dei suoi due ingressi (un valore che consideriamo neutro), il valore logico dell'uscita dipenderà soltanto da quello dell'altro ingresso.

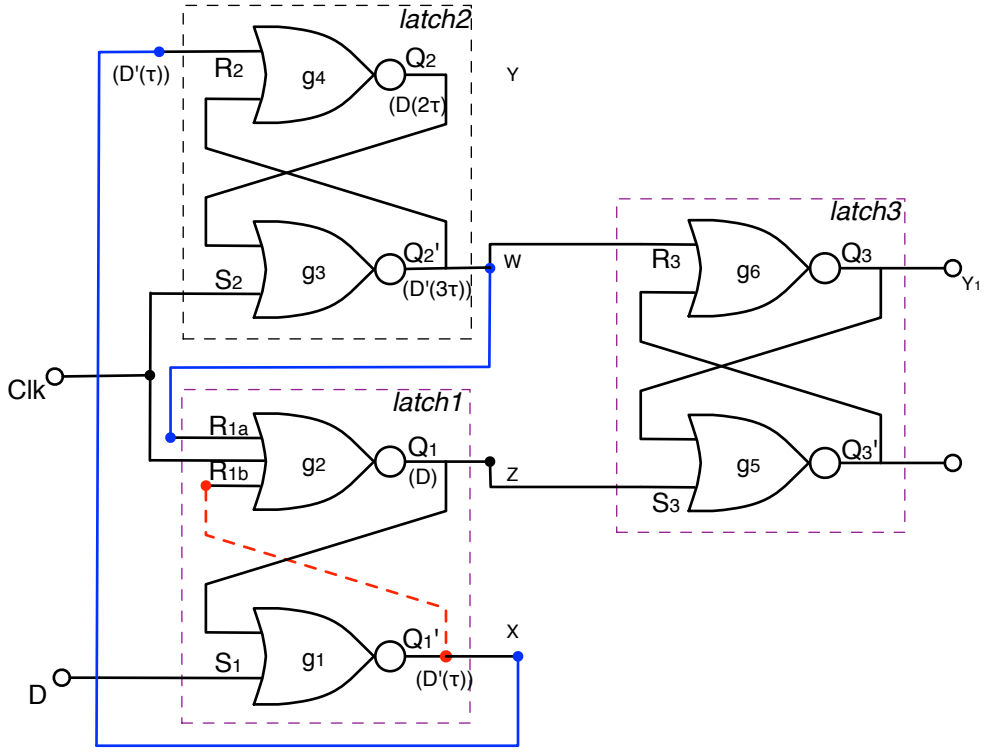


Figura 2.23: flip-flop D edge triggered

I latch *latch1* e *latch2* di cui alla fig.2.23, sono fra loro interconnessi in anello chiuso. La reazione avviene per tramite delle linee evidenziate in blu che collegano le due uscite negate Q_1' e Q_2' rispettivamente con gli ingressi R_2 e R_{1a} [2].

Se il segnale di clock è nello stato alto (clk=1), il nodo Z della porta g_2 e quello W della g_3 sono forzati al livello basso, indipendentemente dai valori presenti negli altri ingressi delle due porte. Tale stato (Z = W=0) rappresenta un valore neutro per gli ingressi R_3 e S_3del *latch3* che, pertanto, continua a mantenere immutato lo stato delle sue uscite Q_3 e Q_3'. Le due porte g_1 e g_4 sono, quindi, libere di commutare e di seguire lo stato

[2]Il motivo per cui il *latch1* ha due ingressi di reset e non uno, introducendo una asimmetria nel circuito, sarà spiegato nel seguito

dell'ingresso D. In particolare, l'uscita di g_1 assumerà il valore D' con il ritardo τ proprio della porta, mentre l'uscita di g_4, invertendolo, il valore D, ma con un ritardo 2τ dovuto ai tempi di propagazione del segnale attraverso le due porte g_1 e g_4.

Quando il segnale di clock è allo stato basso (clk=0), le due porte g_2e g_3, operano da semplici inverter, essendo il valore 0 un valore neutro per esse. In particolare, g_3 inverte il valore di D dopo 2τ, in quanto proveniente da g_4 attraverso g_1, mentre g_2 inverte con un ritardo di 3τ il valore D' proveniente da g_1 per tramite di g_3 .

Quindi, quando clk=0, dopo 3τ, il punto W, collegato al reset R_3 , assumerà il valore D' e il punto Z, collegato al set S_3 il valore D. Tali valori forzeranno, quindi, le uscite del *latch3* ad assumere un valore coerente con il valore di D, in particolare, se si verifica la condizione D=Z=1, si forzerà l'uscita ad assumere il valore Q_3=1, se la condizione D'=W=1 il valore Q_3=0.

Da quanto illustrato si deduce che quando clk=1, il *latch1* segue il valore di D e il *latch2* quello di D'; all'atto della transizione 1->0 di clk, lo stato dei valori di D e D' in quell'istante è trasferito rispettivamente a R_3 e S_3 definendo, in tal modo, lo stato delle uscite del *latch3* coerentemente con il comando D.

Il flip-flop descritto, supponendo, come detto, uguale a τ il ritardo delle sue porte, impiega 3τ per la commutazione di *latch1* e *latch2* e 2τ per la commutazione del *latch3*, per un totale di 5τ che rappresenta il tempo di setup t_{setup}[3].

Sostituendo le sei porte NOR con altrettante porte NAND si perviene a un flip-flop D sensibile sul fronte di salita del clock. La NAND, dualmente alla NOR, è 1-attiva in quanto presenta l'uscita bassa solo se i suoi ingressi sono entrambi alti mentre negli altri casi fornisce l'uscita alta.

L'analisi del D-latch può essere fatta anche considerando il flip-flop D realizzato come un sistema di reti sequenziali composto da tre latch RS (ovviamente tale soluzione è del tutto equivalente circuitalmente a quella a 6 NOR precedentemente descritta). La fig.2.24 mostra tale soluzione (non si riporta per semplicità nel disegno il terzo latch RS pilotato dalle due linee di comando S_3 e R_3, inessenziale per la spiegazione del funzionamento del circuito).

La tabella 2.5 riporta gli stati degli ingressi e delle uscite dei due latch al variare degli ingressi D e Clk del flip-flop D. Come già visto nell'analisi del circuito a NOR sopra riportata, quando Clk=1, il *latch2*, se D=0, ha gli ingressi R_2=S_2=1 e, di conseguenza, le sue uscite assumeranno entrambe il valore Q_2= Q_2'= 0, analoga cosa avviene per il *latch1* quando D=1. In tali condizioni i primi due stati seguono le variazioni di D (sono trasparenti a D), in particolare nel primo caso il flip-flop si predispone per memorizzare l'ingresso D=0, nell'altro il D=1. Tale valore di D sarà trasferito in uscita al verificarsi del fronte attivo 1-0. Quando si perviene alla condizione Clk=0, è attivo il circuito colorato in blu nella figura, quindi il *latch2* è pilotato solo dall'ingresso R_2che assume il valore neutro 0 per il *latch2* se D=1 e il valore 1 se D=0. Riepilogando, il *latch1* presenterà lo

[3]Si ricorda che l'uso di un tale tipo di flip-flop come elemento di memoria in circuiti digitali reazionati richiede che sia soddisfatta la condizione che il tempo di ritardo t_d nell'anello di reazione soddisfi la condizione $t_d > t_{setup}$

stato Q1=S3=1 se D= 1 e il *latch2* lo stato Q2'=R3=1 se D=0, pilotando così in modo coerente il *latch3*.

Clk	D	R_1	S_1	$Q_1 = S_3$	Q_1'	R_2	S_2	Q_2	Q_2'=R_3
1	0	1	0	0	1	1	1	0	0
1	1	1	1	0	0	0	1	1	0
0	0	0	0	0	1	1	0	0	1
0	1	0	1	1	0	0	0	1	0

Tabella 2.5: Tabella analisi del flip-flop D visto come sistema di tre latch RS

2.7.2 Sintesi del flip-flop D sensibile al fronte di salita del clock

La sintesi del flip-flop D sensibile al fronte di salita (e quindi a porte NAND) è effettuata seguendo il tradizionale procedimento di sintesi delle macchine sequenziali asincrone. Tale procedimento si avvia con la costruzione della tabella primitiva (Tab.2.6) che riporta, a partire da uno stato iniziale stabile, le evoluzioni di stato del sistema al variare dello stato dei suoi ingressi (D e clk) e il valore assunto dalle uscite (Q). La tabella primitiva può essere dedotta a partire da un diagramma temporale che associa differenti stati q_i a tutte le combinazioni differenti dei segnali di ingresso e di quelli delle uscite.

Dovendo rappresentare per ciascuna configurazione degli ingressi l'evoluzione della rete al variare delle possibili configurazioni delle uscite, per una rete con n ingressi e m uscite si potranno avere al massimo 2^{n+m} configurazioni possibili che corrisponderanno ad altrettante righe della tabella di flusso primitiva, contenente per ogni riga un solo stato stabile. Tale tabella, per come è costruita, risulterà ridondante nel numero di stati, per cui essa è da ridurre nelle fasi successive della sintesi per pervenire a una tabella a stati ridotti da codificare negli stati, ingressi e uscite e, quindi, da sintetizzare.

Nel caso specifico, avendo la rete sequenziale due ingressi e due uscite, la tabella primitiva potrà avere al massimo 2^{2+2}=16 righe, ciascuna con uno stato stabile. Essendo però le uscite Q e Q' l'una inversa dell'altra, si può semplificare la costruzione della tabella considerando una sola delle due uscite e operare, quindi, con una tabella primitiva ancora ridondante, ma con la metà delle righe (2^{2+1}=8) rispetto alla precedente. Una volta ridotta applicando la minimizzazione degli stati, la tabella primitiva si trasforma in una tabella equivalente di sole quattro righe. Si procede, quindi, alla codifica degli stati equivalenti per pervenire alla tabella in codice di fig.2.6.

Dalla tabella in codice si derivano le due tabelle di verità per la sintesi delle variabili secondarie y1 e y2 riportate in fig.2.7:

Dopo aver effettuato la minimizzazione è possibile derivare le equazioni per Y_1 e Y_2 (si selezionano per Y1 gli implicanti primi {5,7,9,11} {8,9,10,11} e {8,10,12,15} per Y2 {5,7,9,11} {2,6,10,14}):

	D, clk				**Q**
Stato	*00*	*01*	*11*	*10*	
S1	**S1**	S2	-	S4	0
S2	S1	**S2**	S3	-	0
S3	-	S2	**S3**	S4	0
S4	S1	-	S7	**S4**	0
S5	**S5**	S2	S3	S8	1
S6	S5	**S6**	S7	-	1
S7	-	**S6**	**S7**	S8	1
S8	S5	-	S7	**S8**	1

Codifica degli stati	**y1y2**
Q_0(S1, S2, S3)	00
Q_1(S4)	01
Q_2(S5)	10
Q_3(S6, S7, S8)	11

	D, clk↑				**Q**
	00	*01*	*11*	*10*	
Q_0	**Q_0**	**Q_0**	**Q_0**	Q_1	0
Q_1	Q_0	**-**	Q_3	**Q_1**	0
Q_3	Q_2	**Q_3**	**Q_3**	**Q_3**	1
Q_2	**Q_2**	Q_0	-	Q_3	1

	D, clk↑				**Q**
y1y2	*00*	*01*	*11*	*10*	
00	**00**	**00**	**00**	01	0
01	00	**-**	11	**01**	0
11	10	**11**	**11**	**11**	1
10	**10**	00	-	11	1

Tabella 2.6: Tabelle di flusso primitiva del flip-flop D-Edge Triggered sul fronte di salita, ridotta con fusione righe e codificata

Tabella per Y1	**D,clk**				**Q**	Tabella per Y2	**D,clk**				**Q**
y1y2	*00*	*01*	*11*	*10*		**y1y2**	*00*	*01*	*11*	*10*	
00	0	0	0	0	0	*00*	0	0	0	1	0
01	0	**-**	1	0	0	*01*	0	**-**	1	1	0
11	1	1	1	1	1	*11*	0	1	1	1	1
10	1	0	-	1	1	*10*	0	0	-	1	1

Tabella 2.7: Tabelle relative alle variabili di stato secondarie Y1 e Y2

$Y_1 = y_1 \cdot y_2 + y_2 \cdot clk + y_1.\overline{clk} = y_1 \cdot (y_2 + \overline{clk}) + y_2. \cdot clk = y_2 \cdot clk + y_1\overline{clk}$

$Y_2 = y_2 \cdot clk + y_2 \cdot D + D \cdot \overline{clk} = D(y_2 + \overline{clk}) + y_2 \cdot clk$

che espresse in forma di soli NAND divengono:

$Y_1 = (y_1 \uparrow (\overline{y}_2 \uparrow clk)) \uparrow (y_2 \uparrow clk)$

$Y_2 = (D \uparrow (\overline{y}_2 \uparrow clk)) \uparrow (y_2 \uparrow clk)$

Da esse, dopo aver fatto le assegnazioni di seguito riportate, si perviene al circuito di fig.2.25, senza la connessione indicata con la linea tratteggiata in rosso che vedremo sarà inserita per risolvere la corsa tra xxxxx:
$g_4 : (g_1 \uparrow g_3)$
$g_3 : (y_2 \uparrow clk)$
$g_2 : (\overline{y}_2 \uparrow clk)$
$g_1 : (D \uparrow g_2)$
$g_6 : (g_5 \uparrow g_3)$
$g_5 : (y_1 \uparrow g_2)$

Il circuito generato risulta costituito da tre latch di tipo RS fondamentale, di cui i primi due interconnessi in ciclo chiuso. Il *latch1* è però incompleto in quanto privo della retroazione indicata con la linea tratteggiata in rosso. Per le considerazioni fatte precedentemente nell'analisi del flip-flop a porte NOR, l'ingresso R_{1a} assume, quando clk=0, il valore D' ritardato di 3τ. Essendo presente all'uscita della g_1 lo stesso segnale D' ritardato di τ, lo si può mettere in or con R_{1a} collegando l'uscita dell'OR all'ingresso del *latch1*, eliminando in tal modo la condizione di hazard (che sarà discussa nel paragrafo seguente) e, nel contempo, generando nel *latch1* il collegamento mancante presente nei latch RS. <<<<ATTENZIONE inserire il ragionamento dell'alea in tabella!!>>>>

2.7.3 Analisi degli Hazard mediante simulazione

Con riferimento al D-latch implementato a NAND di fig.2.25, si riscontra un'alea quando l'uscita Z della porta g_2 è alta, forzata da $clk = 0$, mentre $R1a = 1$ (tale valore neutro determinato dal valore $D = 1$ che si propaga tramite g1, g4 e g3 sul punto W, con $W = 1, Y = 1, X = 0$) e la linea di *clk* effettua una transizione da 0->1.

Tale transizione che dovrebbe lasciare inalterata l'uscita Z a causa della simultanea commutazione di $R1a$ al valore basso. Purtroppo, a causa del ritardo della porta g_3, $R1a$ assume solo dopo τ il valore basso, generando la sequenza tipica 1->0->1 dell'alea che genera un glitch all'uscita di g_2 (evidenziato nel diagramma di fig.2.26, causa di un possibile malfunzionamento.

Aggiungendo alla porta g_3 l'ulteriore ingresso R_{1b} collegato all'uscita di g_1 , che assume sempre un valore inverso a quello di Y, si elimina l'alea garantendo durante la transizione di clk la copertura del valore basso agli ingressi di g_1 .

L'alea può essere evidenziata osservando i risultati della simulazione behavioral del D-latch riportata in fig.2.26 in assenza del collegamento marcato in rosso. In fig.2.27

è, invece, riportata la simulazione effettuata in presenza di tale connessione che elimina l'alea.

2.8 Analisi e progettazione di un flip-flop Master Slave

Il funzionamento dei flip-flop visti precedentemente, in particolare del Flip-Flop JK dipende in maniera critica dalla durata dell'impulso di clock. È possibile rendere i flip-flop insensibili alla durata o al valore dell'impulso di clock, attivando le transizioni di stato solo quando il clock passa da basso ad alto o viceversa. Un circuito che verifica questo funzionamento è il Flip-Flop Master Slave, che è in effetti realizzato come la cascata di due Flip Flop controllati dallo stesso segnale di clock in maniera opportuna. Un possibile schema realizzativo è quello di figura, che può essere implementato a partire dall'entità RS_Clocked definita in precedenza:

Per verificarne le funzionalità si può effettuare una semplice simulazione utilizzando i seguenti stimoli in ingresso:

```
Clock <= '0', '1' after 20 ns,'0' after 40 ns, '1' after 60 ns, '0' after 80 n
Sin <= '0', '1' after 30 ns, '0' after 50 ns;
Rin <= '0', '1' after 70 ns,'0' after 90 ns;
```

La cui simulazione post-route dimostra la funzionalità del circuito.

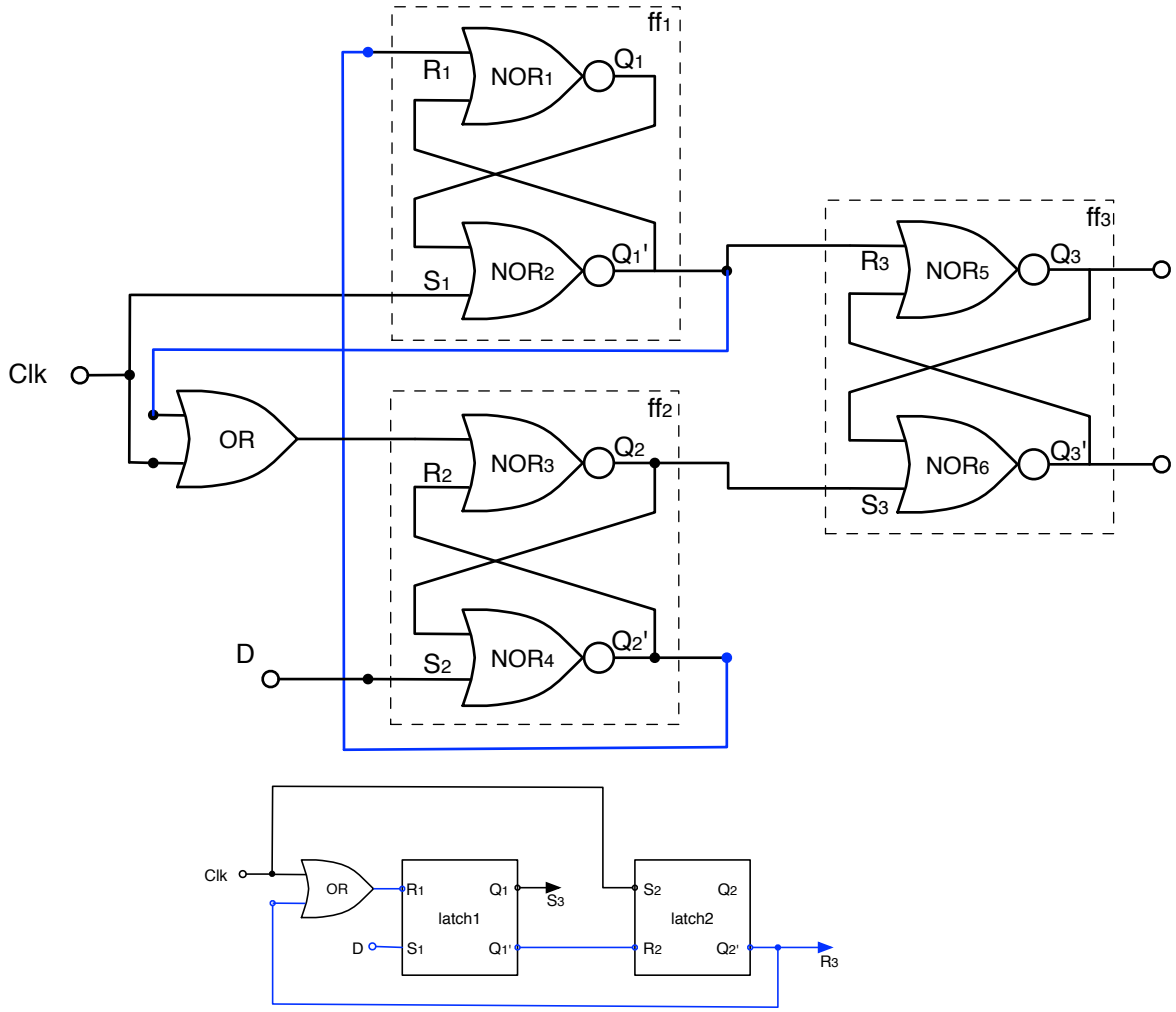


Figura 2.24: Flip-flop D realizzato con latch RS

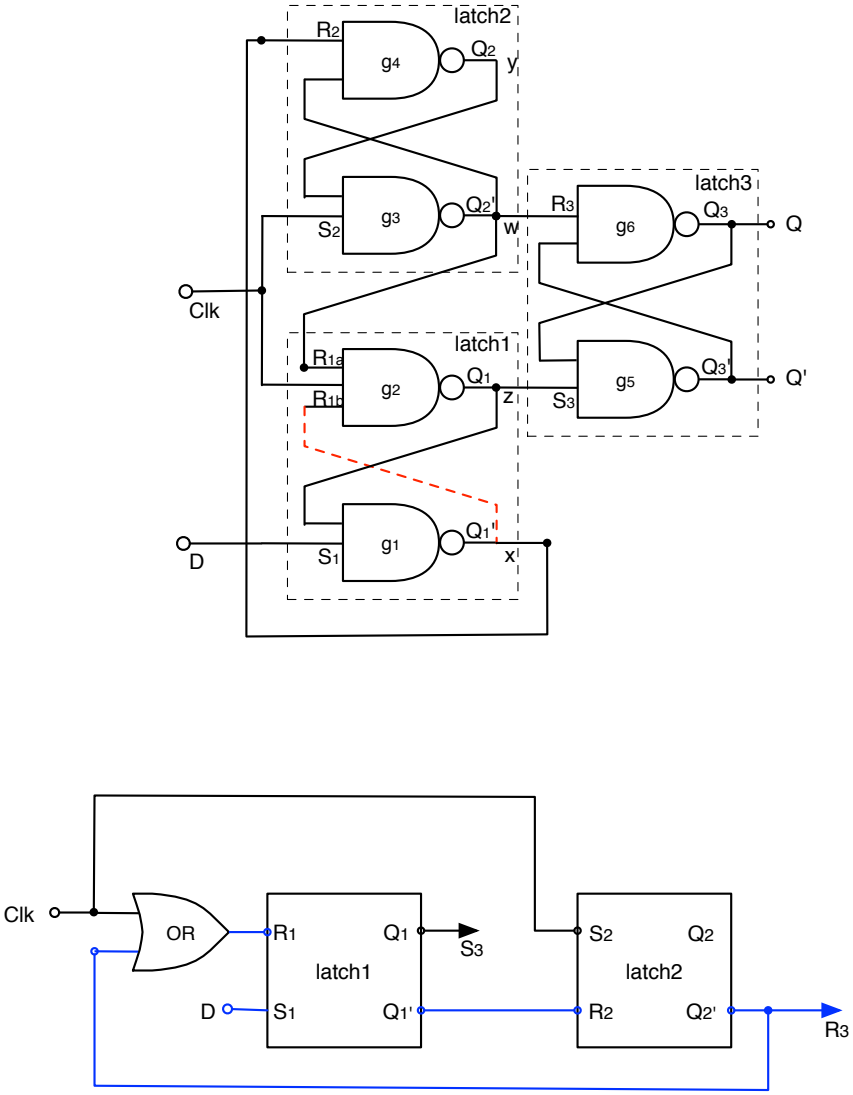


Figura 2.25: flip-flop D edge triggered

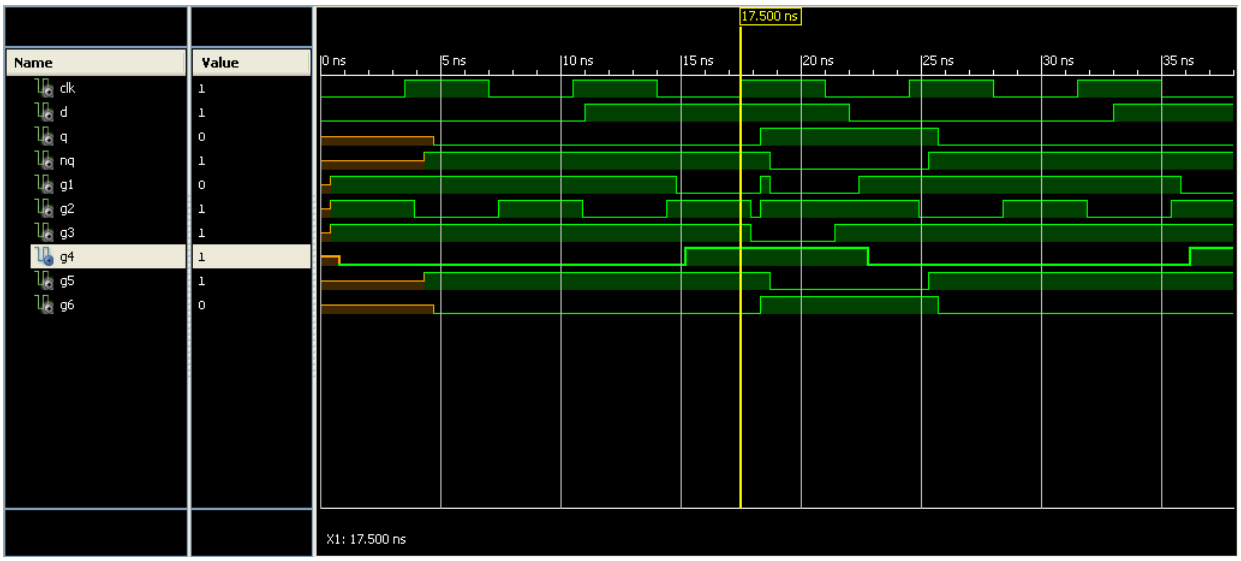


Figura 2.26: Latch RS edge triggered a NAND con alea

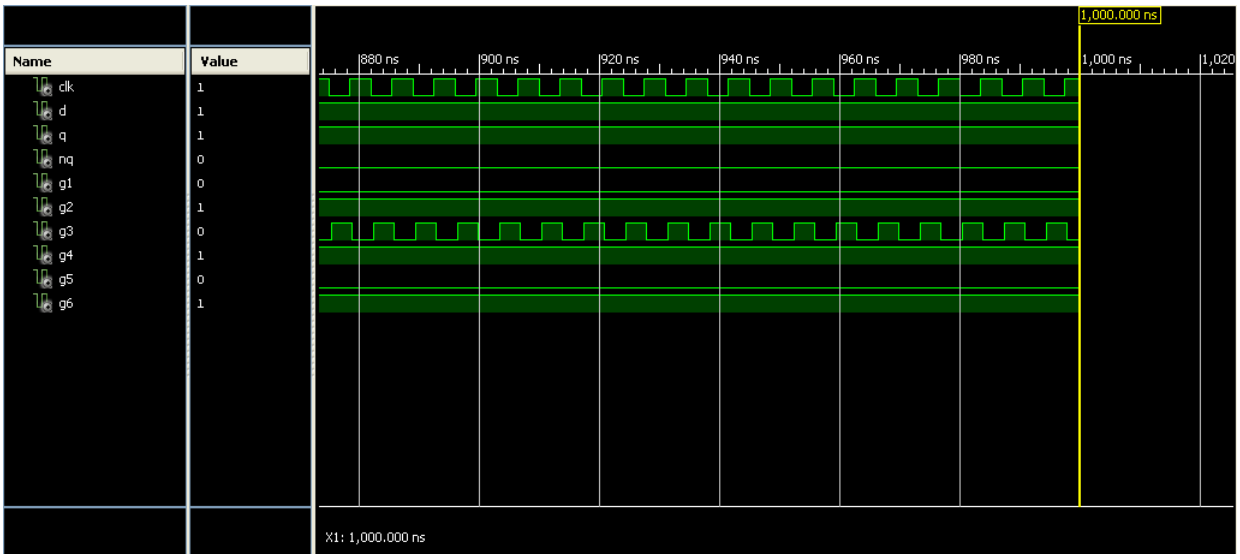


Figura 2.27: Latch RS edge triggered a NAND senza alea

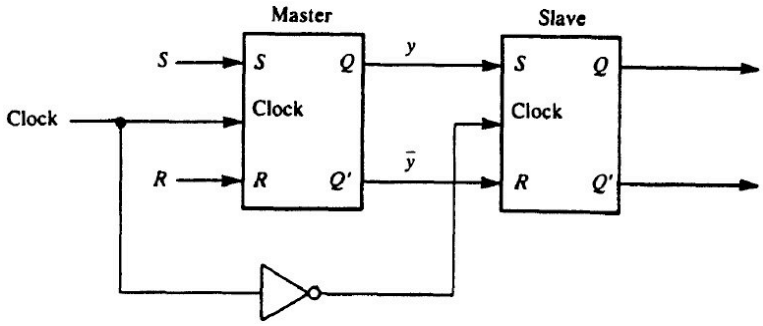


Figura 2.28: Flip-Flop Master-Slave

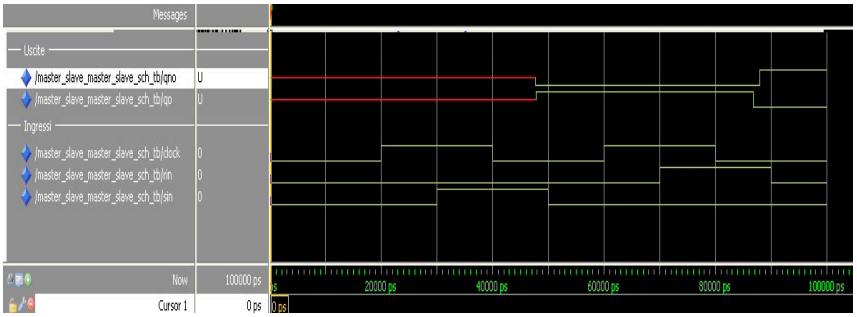


Figura 2.29:

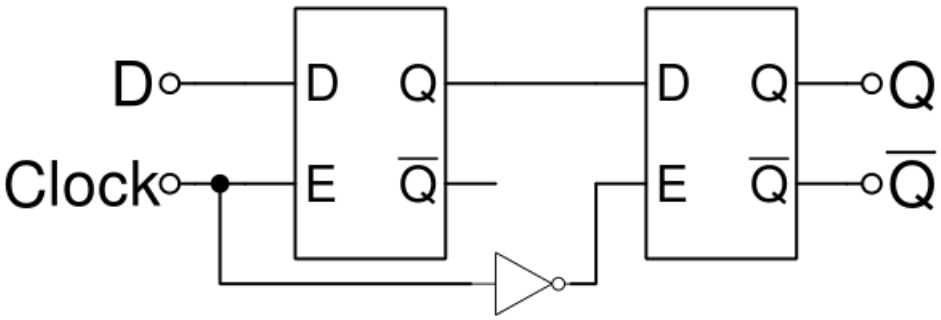


Figura 2.30: Flip-flop D master-slave

3 Esempi in VHDL dei flip flop e latch illustrati

Si riportano di seguito i sorgenti in VHDL di tutti gli esempi discussi precedentemente che possono essere eseguiti per la simulazione o la sintesi su FPGA nell'ambiente ISE di Xilinx utilizzato. Ovviamente, con qualche piccolo riadattamento, gli stessi sorgenti possono essere utilizzati anche in altri IDE di sviluppo.

Si lascia allo studente l'esercizio di effettuare il porting degli stessi esempi nell'ambiente SystemC scaricabile dal sito Accelera.

3.1 D Latch Edge Triggered

3.1.1 Listato del Latch in VHDL

```
--------------------------------------------------------------------------------
--
-- Title       : Flip-flop D_edge-triggered con NOR
-- Design      : D_EDGE
-- Author      : Antonino Mazzeo
-- Company     : DIS
--
--------------------------------------------------------------------------------
--
-- File        : d_edge_nor.vhd
-- Generated   : Friday oct 21, 22:10:44 2011
-- From        : interface description file
-- By          : Itf2Vhdl ver. 1.20
--
--------------------------------------------------------------------------------
--
-- Description : Il flip flop d implementato è di tipo edge triggered sul
    fronte
-- 1->0 di discesa (trailing edge) del clock C
--
--------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity d_edge is
  generic (td: time:=0.0 ns);
end d_edge;
```

```
architecture d_edge of d_edge is

signal  D, C : STD_LOGIC:='0';
signal   Q    : STD_LOGIC;
signal   NQ   : STD_LOGIC;
signal   G1   : STD_LOGIC;
signal   G2   : STD_LOGIC;
signal   G3   : STD_LOGIC;
signal   G4   : STD_LOGIC;
signal   G5   : STD_LOGIC;
signal   G6   : STD_LOGIC;
signal  G2b  : STD_LOGIC;


begin
   G1<=D nor G2 after td;
   G2<=not(G1 or G3 or C) after td; --nor(G1,G3,C)
-- G2<=(G1 nor C) after td;
   G3<=C nor G4 after td;
   G4<=G3 nor G1 after td;

   G5<=G2 nor G6 after td;
   G6<=G5 nor G3 after td;
   Q<=G6; NQ<=G5;
   D <= '1' after 5 ns, '0' after 14 ns, '0' after 18 ns,'1' after 21 ns,'0'
       after 23 ns,'1' after 25 ns,'0' after 40 ns, '1' after 43 ns, '0' after
       48 ns, '0' after 52 ns,'1' after 57 ns,'0' after 61 ns,'1' after 65 ns
       ,'0' after 69 ns;
  -- D<=not D after 6 ns;
  C <= '1' after 5 ns, '0' after 10 ns,'1' after 15 ns, '0' after 20 ns, '1'
      after 25 ns, '0' after 30 ns,'1' after 35 ns, '0' after 40 ns;
   --C<=not C after 1 ns;


end d_edge;
```

4 Sintesi su scheda Nexys2 con FPGA Spartan 3E 1200

4.1 ISE Suite Xilinx

Le varie versioni dei latch e flip-flop descritti, dopo la simulazione behavioral, dovranno essere sintetizzate sull'FPGA Spartan3E (250 o 1200) e ivi testati, mostrando anche gli effetti sulle uscite in presenza dell'alea o meno.

L'ambiente IDE da utilizzare è Ise suite di Xilinx e che comprende al suo interno un completo insieme di strumenti per la modellazione e la sintesi dei progetti da realizzare.

Appendice A

Segnali 0-attivi e 1-attivi

Solitamente, uno dei due valori di una variabile logica è associato all'attivazione di una qualche funzione di un sistema digitale. Un tale valore è detto valore *attivo* mentre l'altro detto valore *neutro*, è un valore di riposo che non crea alcun evento significativo per l'evoluzione del sistema digitale (non attiva alcuna funzione). La scelta di quali dei due valori deve essere attivo dipende dal tipo si sistema considerato e dalla scelta di progetto. In particolare, una variabile è di tipo 1-attiva se lo stato 1 è scelto come stato attivo, 0-attiva nel caso contrario (le variabili 1-attive sono indicate con X e quelle 0-attive con X').

Rappresentazione di una variabile booleana in logica positiva o negativa

Nella realizzazione di un circuito digitale, i due valori di una variabile logica 0 e 1 (indicati anche come L o H) sono associati ai valori di una grandezza di un segnale che, nel caso di tensione, può assumere due valori V_1 e V_0. Se $V_1 > V_0$ la logica utilizzata per rappresentare le variabili è detta *positiva*, se $V_1 < V_0$ è detta negativa (ad esempio, i valori di tensione +5 V associato al livello H e 0 V a quello L, individuano una logica positiva. Occorre fare l'assegnazione inversa se si vuole operare in logica negativa). Una funzione combinatoria f utilizzata in logica positiva, se utilizzata in logica negativa, opera in modo duale [a] e viceversa. Ad esempio, in logica negativa, l'operatore AND ($\overline{X} \cdot \overline{Y}$) opera, se usato in logica positiva, come un operatore OR ($\overline{X} \cdot \overline{Y} \Longleftrightarrow X + Y$), una NAND fra due variabili in logica positiva $\overline{X \cdot Y}$ è equivalente alla sua duale NOR ($\overline{X \cdot Y} \Longleftrightarrow \overline{\overline{X} + \overline{Y}}$) fra variabili che operano in logica negativa. Ove occorra trasformare una variabile operante in logica positiva in una operante in logica negativa, e viceversa, occorre ricorrere all'uso di un inverter.

X	Y	**AND**	**OR**	**NAND**	**NOR**	
L	L	L	L	H	H	
L	H	L	H	H	L	
H	L	L	H	H	L	
H	H	H	H	L	L	
		OR	**AND**	**NOR**	**NAND**	

Figura 4.1: Operatori logici e loro duali

[a] Da una qualsiasi identità boolena se ne può trarre un'altra per dualità, sostituendo a ogni operatore e agli elementi 0 e 1 il rispettivo duale: il duale di + è *, il duale di 0 è 1. Il duale di una semplice variabile x è $\overline{x}$. Data una funzione $f(x_1, x_2, \ldots, x_n)$ in qualsiasi forma, si definisce funzione duale di f, e si indica con δf, una funzione che ha per forma la forma duale di f, ad esempio: $y = \overline{a} + b(c + 0) \qquad \delta f = a * (\overline{b} + \overline{c} * 1) = \overline{a} * (b + c * 1)$.

Si fa osservare che la funzione non cambia qualora non si effettui la negazione delle variabili. Devono essere legate solo le costanti 0 e 1.

Appendice B

Behavioral Simulation A behavioral simulation uses the Verilog source code you wrote in order to model the behavior of the module under test. Neither gate delays nor interconnect delays are modeled. Furthermore, functionality of the behavioral model may not match that of the synthesized logic, if you make use of Verilog language features that the synthesis engine cannot handle. (In particular, be careful about incomplete sensitivity lists.)

While behavioral simulation gives the least accurate prediction of how the final hardware implementation will perform, it is the most useful form of simulation during the initial debugging of a design. There is little point in running more realistic simulations until the behavioral model works correctly.

Non-Behavioral Simulations

Non-behavioral simulations are performed on some form of synthesized netlist. To simulate synthesized netlists of the fsm module, you need to fully synthesize the fsm module. You will need to set fsm as the top module first by right clicking on fsm1.

Post-Translation Simulation

A post-translation simulation uses the synthesized gate-level netlist to model the module under test. The functionality of the gates is modeled using a generic Xilinx library, but propagation delay is not modeled. The simulation should match the behavior of the actual hardware, but will assume the hardware is infinitely fast. (If you are worried about whether your design has synthesized correctly, start by checking the synthesis report file, as all common synthesis mismatch issues will generate warnings in the report file.)

Post-Mapping Simulation In a post-mapping simulation, the gates have been mapped to a library specific to the FPGA device being targeted. This library includes accurate gate delay information. However, interconnect delay is not modeled, because the design at this stage has not yet been placed and routed.

Post-Place-and-Route Simulation A post-place-and-route simulation models interconnect delay, as well as gate delay. This type of simulation will most accurately match the behavior of the actual hardware. However, for large designs, it can take a significant amount of time to extract the interconnect delay values from the place-and-route information, and a significant amount of time to run the actual simulation.

It really only makes sense to perform post-place-and-route simulations at the top level of a design. If you perform a post-place-and-route simulation on a submodule, the place-and-route process is rerun, using the submodule as the top-level of the design. The interconnect delays for the submodule simulation will therefor not match the interconnect delays for that submodule when it is laid out as part of the complete project.