Progetto Reti Logiche

Matteo Bettiati : 10717078
Gionata Brebbia: 10742330

March 2023

Contents

1 Introduzione

Il progetto consiste in un circuito che riceve un dato in input composto da una serie di bit. Tale dato fornirà indicazioni riguardo la scelta di un canale in uscita tra i quattro disponibili. Il dato conterrà anche un messaggio, che dopo una serie di processi, verrà indirizzato verso uno dei quattro canali di uscita.
In particolare, il sistema ha due ingressi primari da 1 bit (W e START) e 5 uscite primarie. Le uscite sono quattro da 8 bit (Z0,Z1,Z2,Z3) le quali conterranno il messaggio in uscita e una da 1 bit (DONE).
All'istante iniziale, ovvero quello del reset, tutti i canali di uscita avranno valore pari a zero.
I dati di ingresso saranno ottenuti come sequenza di bit sull'ingresso W e devono essere letti sul fronte di salita del clock. La sequenza di bit viene letta quando il segnale di START è alto, la lettura termina quando il segnale di START torna uguale a zero. La sequenza letta sarà composta da almeno 2 bit per un massimo di 18 bit. I primi due bit saranno letti da un registro (reg2), il quale li invierà ad un demultiplexer come vettore di 2 bit, il quale li utilizzerà come selettore dell'uscita. I restanti N bit (compresi tra un valore di 0 bit e 16 bit) verranno inviati ad un altro registro (reg3) che farà in modo di completare il dato aggiungendo 16-N zeri a sinistra del dato. Dopodiché il registro invierà un vettore composto dai 16 bit (contenente gli N bit in ingresso e gli 'zeri' aggiunti) in memoria, la quale restituirà un vettore da 8 bit (i_mem_data). Tale vettore sarà l'ingresso del demultiplexer. Il vettore ottenuto dalla memoria, dopo essere passato per il demultiplexer verrà salvato in uno dei quattro registri in uscita. Il canale di uscita sarà determinato dal valore dei bit passati da reg2. I quattro registri in uscita servono per mantenere il dato, il quale verrà stampato nel ciclo di clock in cui il segnale DONE sarà posto =1. Nel ciclo di clock nel quale il segnale di done tornerà =0 il canale in uscita tornerà ad avere un valore uguale a zero.

1.1 Esempio grafico

Figure 1: Esempio di sequenza dei segnali descritti.

2 Architettura

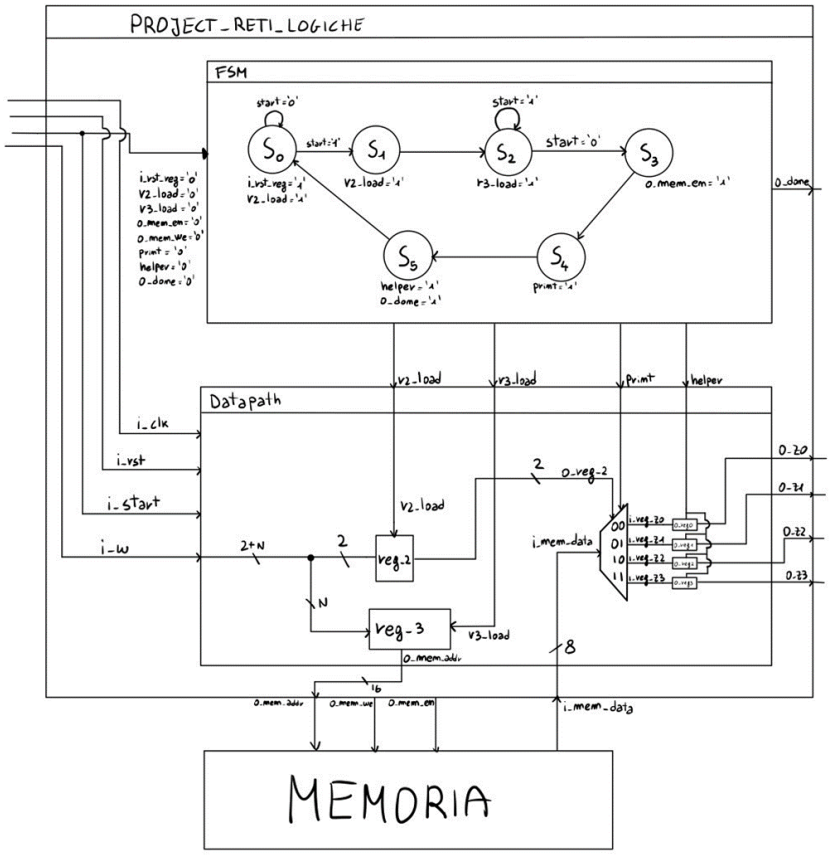

Figure 2: Disegno dei componenti del progetto.

Abbiamo scelto di dividere il progetto in due macro entità, una macchina a stati (FSM) e un datapath per dividere i segnali di controllo da i componenti che poi concretamente gestiscono i bit.

2.1 FSM

La macchina a stati ha 8 stati, un segnale di start e uno di clock per gestirne il controllo. L'obiettivo della macchina a stati è di mandare segnali al datapath in modo da controllare il comportamento dei registri e del demultiplexer. Ad ogni ciclo di clock la macchina a stati compie un movimento che può essere sullo stato corrente o su quello successivo.

Il primo (S0) rappresenta lo stato iniziale, nel quale il segnale di start è ancora basso (=0), e fino a quando il segnale rimane invariato la macchina continua a rimanere nello stesso stato. All'interno di S0 la macchina darà un segnale di reset ai registri reg2 e reg3 in modo da resettarli. All'interno del primo stato verrà dato il segnale r2_load al registro reg2 in modo che possa legger il primo dei 2 bit della sequenza. Il secondo stato (S1) verrà raggiunto nel momento in cui il segnare di start sarà passato = 1. La macchina rimarrà in questo stato per un ciclo di clock per fare in modo che reg2 legga il secondo bit degli N bit in ingresso. Lo stato S2 ha il compito di dare il segnale di lettura a reg3 in modo che legga i restanti N bit del dato in ingresso. Nel ciclo di clock successivo al passaggio di start a 0 la macchina passerà allo stato S3.
Il quarto stato S3 rappresenta lo stato in cui i registri hanno finito di leggere il proprio ingresso, viene quindi dato il segnale di ENABLE alla memoria rendendo possibile la comunicazione con essa. La macchina rimane in S5 per un ciclo di clock dopodiché passa allo stato successivo.
Il quinto stato S4 ha il compito di dare un segnale chiamato 'print' al demultiplexer in modo che quest'ultimo selezioni il canale di uscita e ci passi il dato. La macchina rimarrà in questo stato per un ciclo di clock.
Nell'ultimo stato S5 la macchina darà due segnali. Il primo (o_done) è il segnale in uscita che comunica la fine del processo. Il secondo segnale (helper) sarà strettamente legato al primo, poiché nel momento in cui o_done passerà =1 il sistema dovrà stampare il dato.
Il segnale 'helper' comunicherà quindi ai registri in uscita di stampare il dato ottenuto dal demultiplexer.

2.2 Datapath

Il datapath ha 4 segnali di ingresso generati dal test bench (i_clk, i_rst, i_start, i_w), 4 segnali generati dalla FSM. È inoltre composta da 2 registri (reg2, reg3) che si occupano della lettura del dato i_w, un demultiplexer per selezionare il canale in uscita, e 4 registri con il compito di stampare il dato in uscita (o_reg0, o_reg1, o_reg2, o_reg3).
Il registro reg2 ottiene in ingresso 2 bit che corrispondono ai primi due bit del dato nei primi 2 cicli di clock, ha un segnale di lettura (r2_load) generato dalla macchina a stati e ha come uscita un vettore di 2 bit (o_reg 2). Se viene dato il segnale i_rst a reg2 il dato all'interno del registro esso verrà resettato.
Il registro reg3 ha un segnale di lettura (r3_load) generato dalla macchina a stati. Quando viene dato al registro il segnale di load esso inizia a leggere un numero di bit pari al numero di cicli di clock nei quali il segnale rimane attivo, fino ad un massimo di 16 cicli. Il registro è dotato inoltre di un canale in uscita nel quale il dato verrà convertito in un vettore di 16 bit (o_mem_addr) e passato in memoria. Il demultiplexer ha in ingresso un vettore di 8 bit ottenuto dalla memoria (i_mem_data), 4 uscite (z0, z1,z2,z3), e un segnale di selezione. Il segnale è determinato dal vettore di 2 bit in uscita da reg2, e in base al valore di esso verrà determinata l'uscita. Se o_reg_2 = '00' l'uscita sarà z0, o_reg_2 ='01'

z1, o_reg_2 = ‘10’ z2 ed infine o_reg_2 = ’10’ z3.
I ‘registri in uscita’ (i_reg_z0, i_reg_z1, i_reg_z2, i_reg_z3) ottengono come valore in ingresso il vettore di 8 bit in uscita dal demultiplexer. Essi hanno un segnale ‘helper’ generato dalla macchina a stati, il quale ha il compito di comunicare al registro se stampare il dato ottenuto in ingresso o stampare un vettore con valore nullo. I registri avranno quindi in uscita un vettore di 8 bit (o_z0/o_z1/o_z2/o_z3) che può essere ‘i_mem_data’ oppure ‘00000000’.

3 Risultati Sprimentali

Utilizzando Vivado abbiamo sintetizzato e testato il progetto con svariate simulazioni, questi sono i risultati principali.

3.1 Sintesi:

Il progetto viene sintetizzato da vivado con successo.
Tra i vari dati risultano in particolare:

- Non sono stati utilizzati latch per la realizzazione e il funzionamento del progetto.
- Guardando il report sommario dei tempi dopo la sintesi si ottiene uno SLACK(MET) pari a 97 ns.

3.2 Simulazioni:

In seguito vengono forniti, tra i vari testbench testati, quelli che in particolare cercano di coprire situazioni non standard.
Tutti i testbench sono stati eseguiti sia in post-sintesi functional, sia in post-sintesi timing dando esito positivo.

Figure 3: Wave form del testbench fornito dal professore.

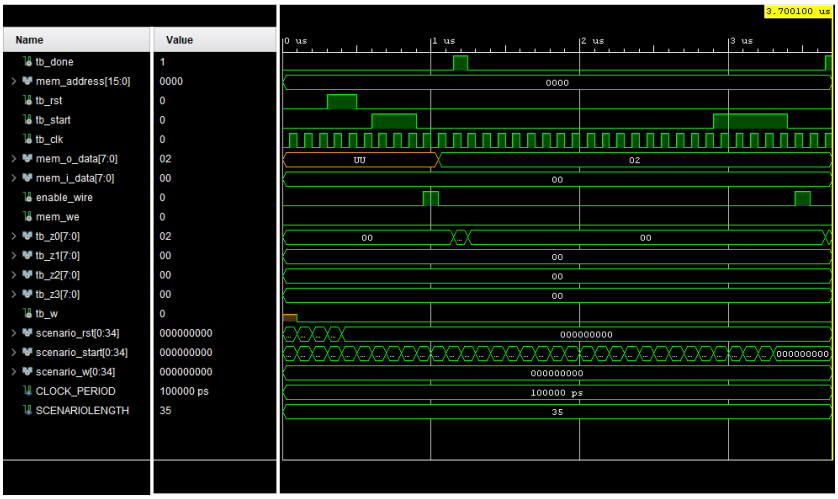

Figure 4: Testbench creato per osservare il comportamento del circuito nel caso, come dato in ingresso, arrivasse una stringa, per due volte consecutive, esclusivamente di 0.

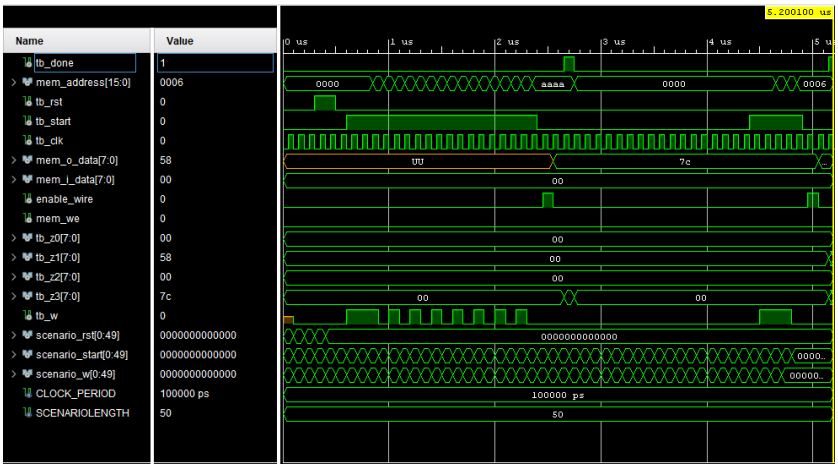

Figure 5: Testbench creato per testare il comportmento del circuito nel momento in cui esso riceva due stringhe da 18 bit casuali.

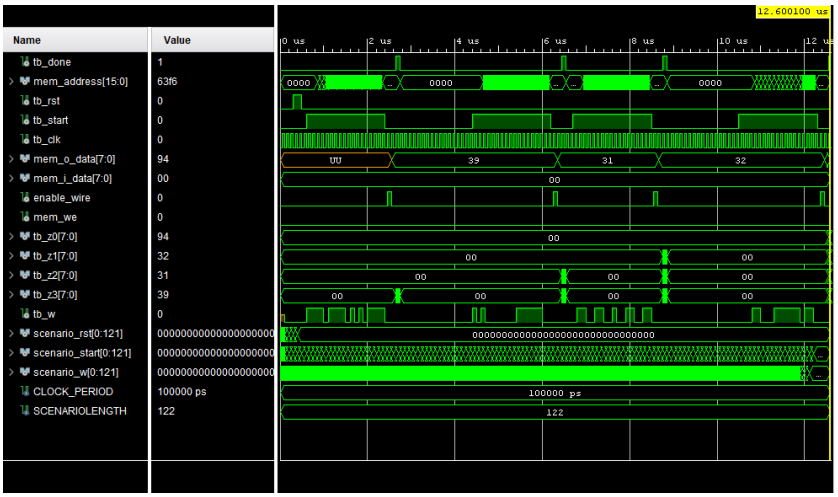

Figure 6: Testbench che per quattro volte di fila pone in ingresso una stringa da 18 bit. (Il progetto è stato testato con esito positivo, anche con più di quattro sequenze).

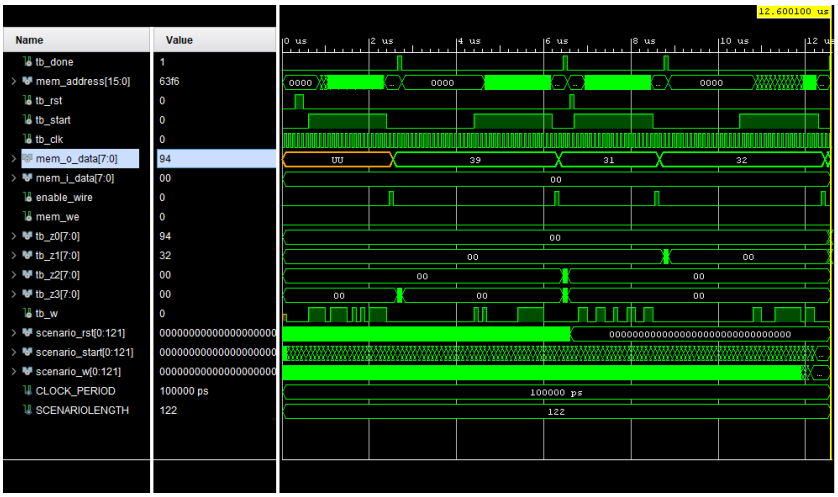

Figure 7: In questo test viene imposto volontariamente un segnale di reset nel mezzo dell'esecuzione del testbench precedente.

4 Conlcusioni

Il progetto rispetta tutte le specifiche richieste.
Inoltre è stato testato a livello di prestazioni e in ogni test di timing post-sintesi ha dato esito positivo.
Tuttavia ci sarebbe un'ultima ottimizzazione da poter implementare.

4.1 Future ottimizzazioni

Future ottimizzazioni porteranno a eseguire tutte le operazioni in un solo ciclo di clock.
In questo momento, da quando il segnale di START torna a '0', tutte le operazioni necessarie vengono svolte in due cicli di clock.
Si può ottimizzare ulteriormente affinchè queste operazioni vengano svolte in un solo ciclo di clock.