RELAZIONE PROVA FINALE RETI LOGICHE A.A. 2022/2023

FIORINI ARMANDO (Cod. Persona 10709856)

March 2023

1 INTRODUZIONE

L'obiettivo della prova è la progettazione di un componente hardware che, interfacciandosi con una memoria, fornisca in output, su un uscita da 8 bit specificata in input tra le 4 disponibili, il dato contenuto in memoria all'indirizzo anch'esso fornito in input (i_w).

Il componente deve inoltre essere dotato di un segnale di reset, che può essere attivato (reset = 1) in qualsiasi momento e lo riporta allo stato iniziale, e due segnali di start e done: il primo viene messo a 1 quando inizia una nuova sequenza di bit da leggere in input e indica perciò una nuova richiesta a memoria: il segnale di start rimane alto per almeno 2 cicli di clock e per non più di 18 cicli; nella lettura, che deve avvenire sempre durante il fronte di salita del clock, i primi due bit ricevuti codificano l'uscita a cui il dato dovrà essere inviato (00 è la prima uscita, 01 la seconda e così via) mentre gli altri (da 0 a 16) a cui vanno aggiunti gli opportuni 0 di estensione a sinistra, identificano un indirizzo di memoria a 16 bit da cui prendere il dato nella memoria; tutti i 18 bit sono forniti sempre a partire da quello più significativo.

Il secondo segnale (o_done) assume il valore 1 per un solo ciclo di clock quando la richiesta è completata e il dato richiesto è disponibile e visualizzabile in corrispondenza dell'uscita specificata: in corrispondenza di ciò su ognuna delle altre tre uscite deve essere visualizzabile l'ultimo valore che quell' uscita ha visualizzato in seguito a precedenti letture da memoria, se ci sono state; al ciclo di clock successivo il segnale o_done torna a 0 e le uscite tornano ad assumere tutte valore 0;

Il componente deve avere la seguente interfaccia:

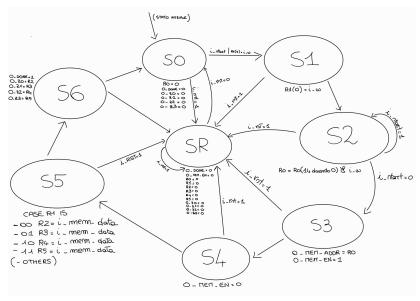
entity project_reti_logiche is
Port (
 i_clk : in std_logic;

```
i_rst: in std_logic;
i_start: in std_logic;
i_w: in std_logic;
o_z0: out std_logic_vector(7 downto 0);
o_z1: out std_logic_vector(7 downto 0);
o_z2: out std_logic_vector(7 downto 0);
o_z3: out std_logic_vector(7 downto 0);
o_done: out std_logic;
o_mem_addr: out std_logic_vector(15 downto 0);
i_mem_data: in std_logic_vector(7 downto 0);
o_mem_we: out std_logic;
o_mem_en: out std_logic;
o_mem_en: out std_logic
```

La memoria, il cui funzionamento è descritto all'interno dei test bench e non fa parte del componente, contiene dati da 8 bit, contenuti in celle identificate da indirizzi da 16 bit, e possiede 4 segnali che il componente deve controllare: uno di enable (o_mem_en), che permette al componente di entrare in comunicazione dalla memoria, uno di scrittura (o_mem_en), che se posto a 0 abilita la lettura da memoria mentre se a 1 la scrittura, un segnale o_mem_address, in cui viene inserito l'indirizzo di memoria da cui vogliamo leggere (o da cui vogliamo scrivere, cosa che il componente non fa), e il segnale i_mem_data, in cui la memoria carica il dato letto.

2 ARCHITETTURA

La realizzazione del progetto si basa sull'idea di una FSM completamente specificata a 8 stati, di cui 1 di reset, il cui funzionamento è specificato attraverso 2 processi: uno che rappresenta lo state register, che gestisce la transizione di stato della macchina, e uno che realizza contemporaneamente la funzione di uscita e quella di stato prossimo: il cambiamento di stato della macchina avviene sempre durante il fronte di discesa del clock, mentre le operazioni relative al secondo processo vengono effettuate sempre durante il fronte di salita. La FSM specificata è così rappresentabile:



- Rappresentazione della FSM (I valori sono espressi, tranne nel case di S5, in codifica decimale, la loro codifica in binario sara assunta bit a bit dal segnale)

2.1 Descrizione dei segnali

Per la realizzazione della FSM ho utilizzato i seguenti segnali:

signal r0: std_logic_vector(15 downto 0); - memorizza i bit di indirizzo da passare alla memoria;

signal r1: std_logic_vector(1 downto 0); - memorizza i 2 bit che identificano l'uscita a cui inivare il dato letto da memoria;

signal r2: std_logic_vector(7 downto 0); - memorizza l ultimo valore assunto da o_z0;

signal r3: std_logic_vector(7 downto 0); - memorizza l ultimo valore assunto da o_z1;

signal r4: std_logic_vector(7 downto 0); - memorizza l ultimo valore assunto da o_z2;

signal r5: std_logic_vector(7 downto 0); - memorizza l ultimo valore assunto da o_z3;

type S is (SR,S0,S1,S2,S3,S4,S5,S6); - tipo che rappresenta gli stati della FSM;

signal curr_st,next_st: S; - rappresentano rispettivamente lo stato corrente e lo stato prossimo della FSM;

2.2 Descrizione degli stati

2.2.1 SR

Stato di reset della macchina, porta a 0 tutti i valori delle uscite, dei segnali R1,..;R5, del segnale di done e di quello relativo all'indirizzo di memoria da cui leggere: si entra in questo stato quando il segnale i_rst si alza a 1 e si esce quando ritorna a 0, entrando nello stato S0.

2.2.2 S0

Stato iniziale della macchina appena avviata e prima di ogni lettura e raggiunto dalla macchina al termine di ogni richiesta, riporta il Done e le uscite al valore 0 insieme a R0, segnale che contiene l'indirizzo d inviare in input alla memoria: quando il segnale di start sale a 1 da S0 si raggiunge S1, e, associata a questa transizione, abbiamo la lettura del primo bit dei due che indicano l'uscita dove mostrare il dato che sarà letto dalla memoria, che viene inserito nella posizione più significativa del segnale apposito R1.

2.2.3 S1

Lo stato S1 viene utilizzato per leggere il secondo bit di codifica dell'uscita, che viene memorizzato nella posizione meno significativa del segnale R1.

2.2.4 S2

Lo stato S2 viene utilizzato per leggere, fintanto che il segnale i_start rimane sul valore alto, un bit da i_w, che viene memorizzato in R0 dopo uno shift verso sinistra: in questo modo il primo bit letto sarà sempre in posizione più significativa (fatta eccezione per gli 0 di estensione nel caso i bit letti fossero meno di 16), il secondo nella seconda e così via.

2.2.5 S3

In questo stato l'indirizzo in R0 viene passato alla memoria e viene posto a 1 il segnale di enable, che, abbinato al segnale di write enable lasciato fisso a 0, consente di leggere il dato dalla memoria.

2.2.6 S4

Stato in cui viene disabilitato il segnale di enable della memoria, il salvataggio del dato in uscita avviene in uno stato successivo per questione di sincronismo.

2.2.7 S5

Nello stato S5 abbiamo l'aggiornamento del segnale associato all'uscita designata per il dato, calcolata in base al valore memorizzato in R1, il segnale che viene aggiornato non è l'uscita bensì un elemento di memoria (Latch in post sintesi) che mantiene l'ultimo valore arrivato all'uscita che gli è associata, valore che deve essere mostrato per ogni uscita al termine della richiesta di lettura.

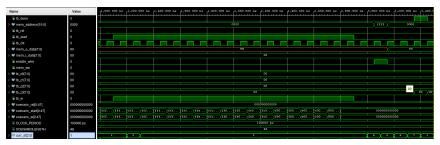
2.2.8 S6

Stato in cui le 4 uscite vengono modificate e da 0, per un ciclo di clock, passano a mostrare i valori presenti nei latch associati; o_done viene messo a 1 durante questo ciclo, dopodichè la macchina torna nuovamente in S0.

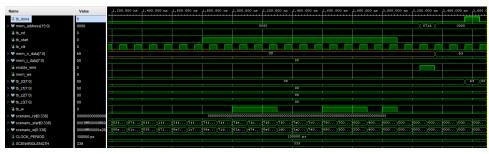
3 RISULTATI SPERIMENTALI

3.1 Simulazioni

Il componente risulta superare tutti i test necessari: Behavioral, Functional post sintesi e Timing. Per verificarne il funzionamento il componente è stato sottoposto ai test bench forniti dal docente, che sono risultati tutti superati. Di seguito alcuni risultati di esempio, riguardanti tipi differenti di casi possibili



1. Esempio di lettura generica, numero di bit massimo, successful



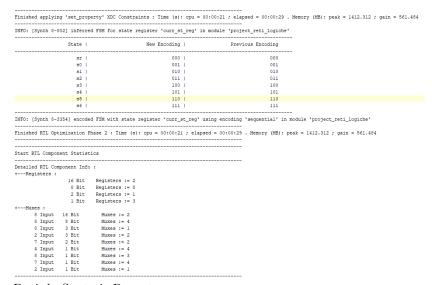
2. Lettura con numero di bit di indirizzo;16 (estensione), successful



3. Lettura con numero di bit minimo, solo i due bit di uscita

3.2 Sintesi

Il componente, correttamente sintetizzabile, viene realizzato utilizzando i componenti descritti dalla parte del report in figura (da notare l'assenza di Latch) I test sopra riportati, in post sintesi, generano lo stessa risposta



Dati da Syntesis Report

1. Slice Logic											
Site Type	i	Used	i	Fixed	i	Prohibited	i	Available	i	Util%	i
Slice LUTs*	ï	16		0		0		134600		0.01	
LUT as Logic	r	16	ī	0	i	0	ī	134600	1	0.01	i
LUT as Memory	Ĺ	0	1	0	i	0	ī	46200	i	0.00	i
Slice Registers	ī	106	ī	0	i	0	ī	269200	ī	0.04	i
Register as Flip Flop	r	106	1	0	i	0	ī	269200	1	0.04	i
Register as Latch	Ĺ	0	1	0	i	0	ī	269200	i	0.00	i
F7 Muxes	L	0	1	0	ī	0	Ī	67300	1	0.00	i
F8 Muxes	r	0	1	0	i	0	ī	33650	1	0.00	i
· ·											ı,

Dati da Utilization Report

3.3 CONCLUSIONI

In conclusione il componente risulta soddisfare la specifica assegnata in maniera efficiente ed ottimizzata, sia dal punto di vista dell' analisi comportamentale, sia nell'analisi post-sintesi, eseguendo quanto richiesto in tempo anche molto breve rispetto alla soglia massima consentita, di fronte a test-bench che ne hanno garantito la sottoposizione a tutti i possibili casi limite ma anche a quelli più generici.

La realizzazione del progetto attraverso una macchina a stati ne ha consentito una facile scalabilità e in un approccio proettato fin dal principio al livello hardware e alla sintesi logica, evitando che quest'ultima creasse particolari problemi e consentendo al componente di risultare anche correttamente implementabile (Ho testato la macchina anche in post-implementation pur non essendo richiesto e risulta superare i test).