Prova Finale (Progetto di Reti Logiche)

Professor Palermo Gianluca

Cognome e nome	*Matricola*	*Codice persona*
Rossi Christian	955333	10736464
Rampone Matteo	935349	10702030

Sommario

Introduzione

Scopo del progetto

Lo scopo del progetto è creare una macchina in grado di leggere una sequenza di bit in ingresso, interpretarla come indice di un canale assieme a un indirizzo di memoria e esporre il dato salvato in memoria sul canale specificato.

Specifiche

Quando viene attivato il segnale di start: il componente dovrà leggere due bit che indicano il canale su cui mandare il dato in memoria. Dopodiché leggerà un numero di bit variabile (massimo sedici) da interpretare come indirizzo di memoria da cui recuperare il dato in lettura. Successivamente alla lettura del dato dalla memoria esso verrà esposto sul canale di uscita selezionato in precedenza rivelando eventuali valori passati precedentemente esposti sui canali rimanenti.

Interfaccia

L'interfaccia del componente dovrà essere la seguente:

```
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
);
END project_reti_logiche;
```

Nello specifico:

i_clk	è il segnale di CLOCK in ingresso generato dal Test Bench.
i_rst	è il segnale di RESET che inizializza la macchina pronta per ricevere il primo segnale di START.
i_start	è il segnale di START generato dal Test Bench.

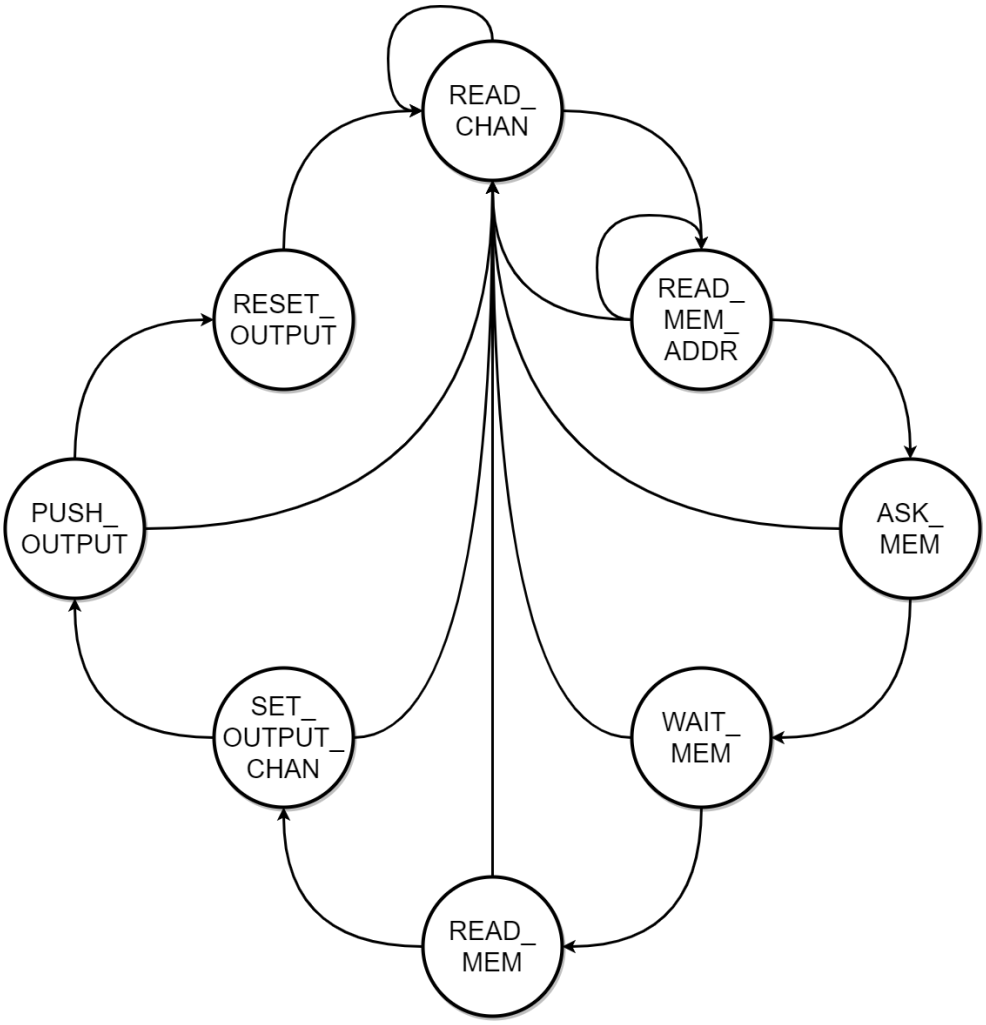
i_w	è il segnale W precedentemente descritto e generato dal Test Bench.
o_z0, o_z1, o_z2, o_z3	sono i quattro canali di uscita.
o_done	è il segnale di uscita che comunica la fine dell'elaborazione.
o_mem_addr	è il segnale (vettore) di uscita che manda l'indirizzo alla memoria.
i_mem_data	è il segnale (vettore) che arriva dalla memoria in seguito ad una richiesta di lettura.
*o_mem_*en	è il segnale di ENABLE da dover mandare alla memoria per poter comunicare (sia in lettura che in scrittura).
o_mem_we	è il segnale di WRITE ENABLE da dover mandare alla memoria (=1) per poter scriverci. Per leggere da memoria esso deve essere 0.

Architettura

Descrizione della macchina a stati finiti

Come da specifica, il componente riceve un segnale di RESET tutte le volte che viene inizializzato.
Nel momento in cui il segnale di START viene settato a 1 vengono letti i 2 bit del canale di uscita e i successivi bit di memoria.
Appena START torna a 0 viene conclusa la lettura dell'indirizzo di memoria e si passa alla lettura del dato in memoria.
Una volta che il dato è stato salvato nel canale corretto viene settato a 1 il segnale di DONE e vengono mostrati i valori effettivi dei canali di uscita.
Al ciclo di clock successivo il segnale di DONE torna a 0 ed è possibile leggere un nuovo dato oppure effettuare un RESET.
N.B.: Il componente ammette segnali di reset asincroni.

Descrizione degli stati della FSM



READ_CHAN

È lo stato iniziale e di reset del componente. In questo stato il componente attende il segnale *i_start* ad alto e, successivamente, immagazzina in un registro la sequenza di 2 bit che caratterizza il canale di uscita della macchina.
Una volta letti i 2 bit di canale il componente passa allo stato di READ_MEM_ADDR.

READ_MEM_ADDR

In questo stato il componente legge una stringa di bit da *i_w*, che rappresenta l'indirizzo di memoria da cui recuperare il dato, fino a che il segnale *i_start* è alto.
Quando il segnale di *i_start* torna a basso, la macchina passa allo stato ASK_MEM.

ASK_MEM

In questo stato il componente si interfaccia con il modulo di memoria settando *o_mem_en* ad alto. Il segnale *o_mem_addr* riporta l'indirizzo memorizzato allo stato precedente.
A questo punto il componente passa allo stato di WAIT_MEM.

WAIT_MEM

In questo stato viene attesa per un ciclo di clock l'elaborazione del dato in uscita dalla memoria. Viene inoltre resettato a basso il segnale *o_mem_en.*
Allo scadere del ciclo di clock il componente passa allo stato di READ_MEM.

READ_MEM

In questo stato il componente memorizza il valore in uscita dal modulo di memoria e effettua una transizione allo stato di SET_OUTPUT_CHAN.

SET_OUTPUT_CHAN

In questo stato il componente aggiorna il valore effettivo del canale scelto nello stato READ_CHAN. Dopodiché' viene effettuata una transizione allo stato PUSH_OUTPUT.

PUSH_OUTPUT

In questo stato vengono propagati i valori effettivi dei segnali sui canali in uscita *o_zX* e viene impostato ad alto il segnale *o_done*.
Viene poi effettuata una transizione a RESET_OUTPUT.

RESET_OUTPUT

In questo stato vengono resettati i segnali dei canali in uscita *o_zX* e viene impostato a basso il segnale *o_done*.
Viene poi effettuata una transizione a READ_CHAN.

Risultati ottenuti

Analisi dei Test Bench

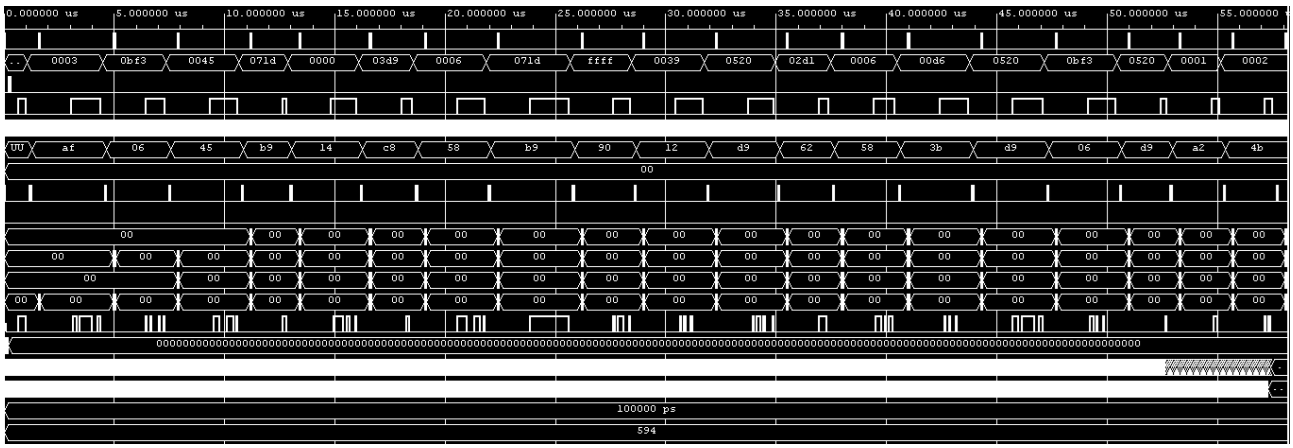
Tutti i test bench forniti dal docente passano lo step di simulazione. Abbiamo convenuto che, tra i vari test forniti, i più sofisticati e meritevoli di una analisi approfondita fossero il test bench due ed il test bench cinque.

Individuazione delle criticità

Le criticità fondamentali che su cui ci siamo concentrati durante lo sviluppo di questo progetto sono state:

- La lettura e interpretazione dell'indirizzo di memoria da cui recuperare i dati in RAM.
- La gestione della funzione di reset nel modulo.
- La permanenza dei dati letti su canali non selezionati.

Test Bench #3

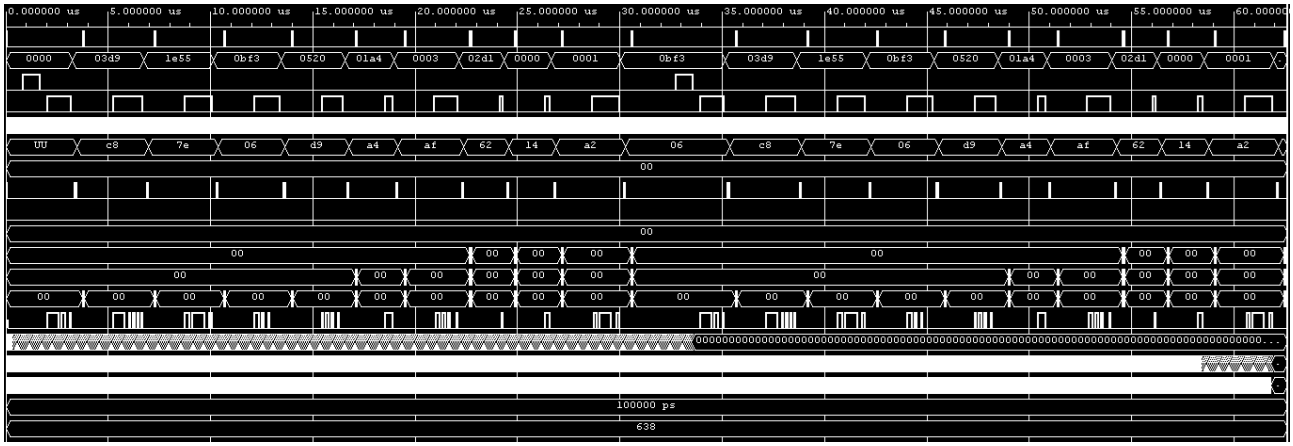


Time: 58.203.737 ps

Questo test valuta le performance del modulo sotto due punti di vista:

- Usando lunghezze "utili" di *i_w* diverse viene verificato che il modulo sintetizzato costruisce e interpreta correttamente l'indirizzo di memoria da cui leggere i dati.
- Vengono usati vari canali di uscita, con alcune ripetizioni. Si verifica così che sia la permanenza dei dati precedentemente letti che la sovra-scrizione di un canale precedentemente usato funzionano correttamente.

Test Bench #5



Time: 62.603.737 ps

Questo test valuta le performance del modulo sotto un punto di vista aggiuntivo rispetto a quelli visti in precedenza:

- In questo test bench viene inviato un segnale di reset al modulo. Viene così appurato che la funzione di reset sintetizzata riporta il modulo a uno stato “pulito” senza problemi.

Conclusioni

Risoluzione delle criticità

Rispetto alle criticità evidenziate nella sezione precedente, le soluzioni applicate sono state:

- L'aggiunta di uno stato di "wait" per mitigare il tempo di accesso ai dati nella RAM, stimabile a 2 ns.
- Un segnale di reset processato in modo asincrono che resetti tutti i segnali interni al modulo
- L'utilizzo di segnali intermedi e un meccanismo di accoppiamento e disaccoppiamento tra il dato letto e il valore riportato in uscita.

Ottimizzazione

A nostro avviso il modulo presenta un sufficiente grado di ottimizzazione. Date le strutture dati usate non sembra possibile una ulteriore crasi degli stati della macchina. Notiamo tuttavia che sarebbe possibile condensare lo stato READ_CHAN con READ_MEM_ADDR a patto di usare una struttura dati diversa, ma per idiomaticità e chiarezza degli stati abbiamo preferito tenerli separati.

Risultati della sintesi

Su tutti i test bench forniti, il componente sintetizzato supera correttamente le tre tipologie di simulazione possibili:

- Behavioural.
- Post-Synthesis Functional.
- Post-Synthesis Timing.