Prova finale – Reti Logiche

Martina Giacomelli 10767035 – 961199

Marco Giulio Grilli 10681127 – 961054

A.A. 2022/2023

1 Requisiti del progetto

Il progetto consiste nell'implementazione di un modulo hardware, descritto in linguaggio VHDL, che si interfacci con una memoria e, prelevandone un dato, indirizzi tale dato su un canale di uscita. Le informazioni relative all'indirizzo di memoria e al canale di uscita da selezionare sono ricevute attraverso un ingresso seriale nel seguente modo: i primi due bit letti rappresentano il canale di selezione tra i quattro disponibili, i restanti bit, che possono essere in numero tra 0 e 16, rappresentano l'indirizzo di memoria a cui si vuole accedere. All'interno della memoria verrà letto, all'attivazione del segnale di lettura, un valore di lunghezza 7 bit che verrà poi indirizzato sul canale corrispondente.

Attraverso un segnale di reset tutti i valori sono inizializzati, preparando il sistema a ricevere il primo segnale di start. Durante l'esecuzione i valori letti e indirizzati ai cicli precedenti vengono salvati, così da poter essere mostrati in uscita ogni qualvolta il segnale di "o_done" sia attivato. L'intera esecuzione si basa su un segnale di clock, la lettura dei segnali avviene sul suo fronte di salita. Qualora si leggesse il segnale di reset bisognerebbe prepararsi ad una nuova esecuzione.

1.1 Richieste di progetto

Al componente viene richiesto di:

- leggere almeno due bit e al massimo 18 bit che rappresentino dapprima il canale di selezione, poi l'indirizzo di memoria dal segnale in ingresso i_w
- accedere all'indirizzo di memoria specificato, leggendo il dato contenuto e salvandolo nel segnale "i_mem_data"
- a seconda dei bit di selezione, indirizzare il valore sull'uscita corrispondente che lo mostrerà in parallelo
- ripetere il processo fino alla fine del tempo di clock

2 Implementazione

2.1 Algoritmo e Macchina a stati finiti

L'algoritmo è implementato in modo tale da utilizzare un solo processo, che sequenzialmente esegue le istruzioni di una macchina a stati finiti (Figura 1).

L’idea dell’algoritmo è quella di utilizzare delle variabili interne al processo per leggere i bit di selezione e l’indirizzo di memoria: vengono utilizzati quindi due bit singoli, che rappresentano il primo bit e il secondo bit letti dal flusso di “i_w”, e un registro di 16 bit per l’indirizzo di memoria. A ogni ciclo di clock, se il segnale “i_start” ha valore 1, viene letto un bit, che viene concatenato al registro in cui è salvato l’indirizzo, shiftandolo di una posizione a sinistra; quando “i_start” ritorna ad avere valore 0 si legge il valore dall’indirizzo di memoria corrispondente. Nel caso in cui “i_start” sia immediatamente 0, dopo la lettura dei bit di selezione, si passa direttamente allo stato successivo di lettura dalla memoria all’indirizzo a cui è inizializzato “o_mem_address”, quindi un vettore di 16 bit a 0.

Finita la lettura si entra in uno stato che rappresenta un demultiplexer (Figura 2) a un ingresso e 4 uscite.

Figura 1

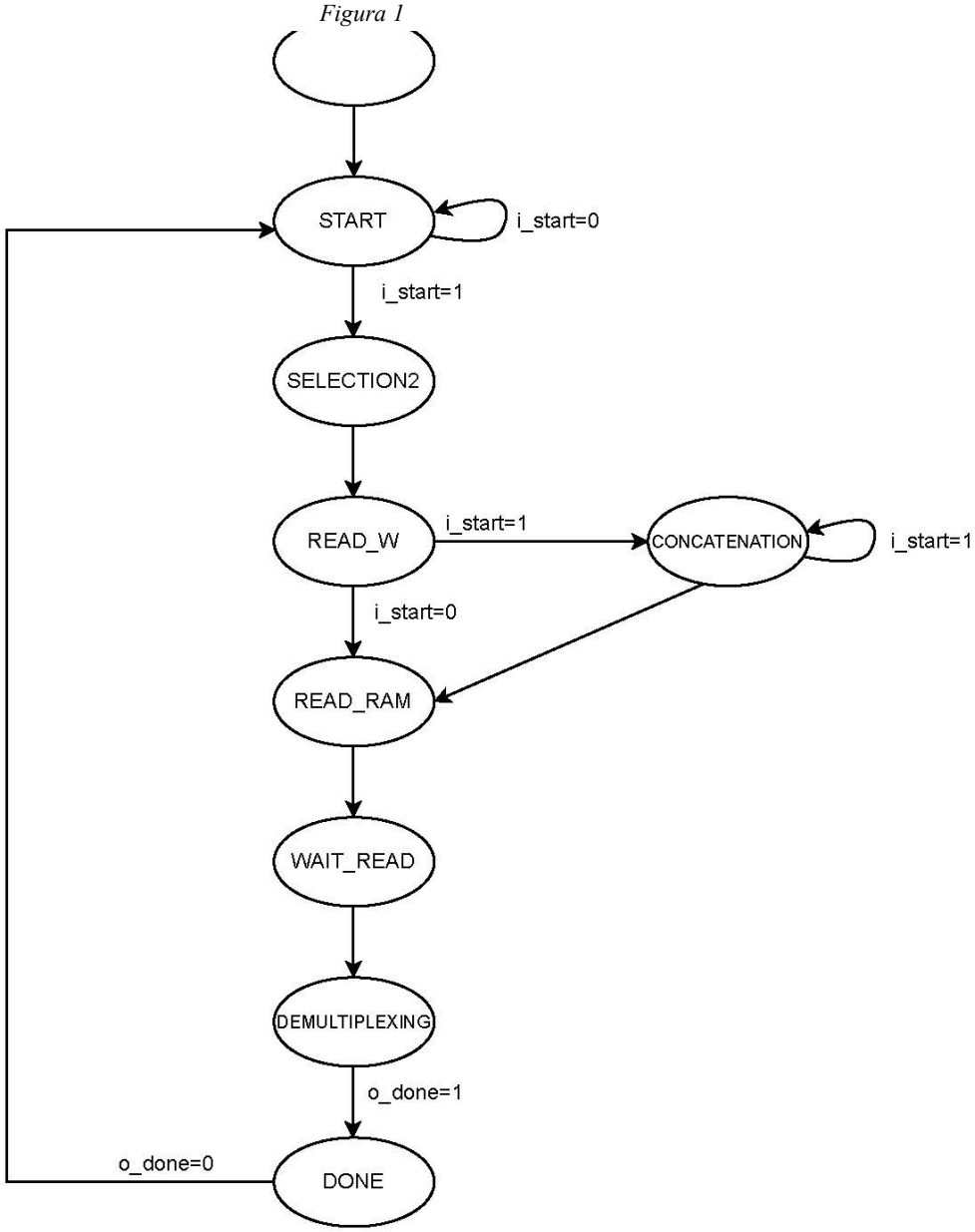
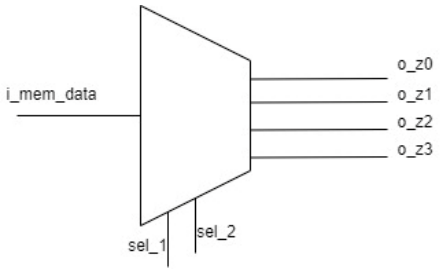


Figura 2



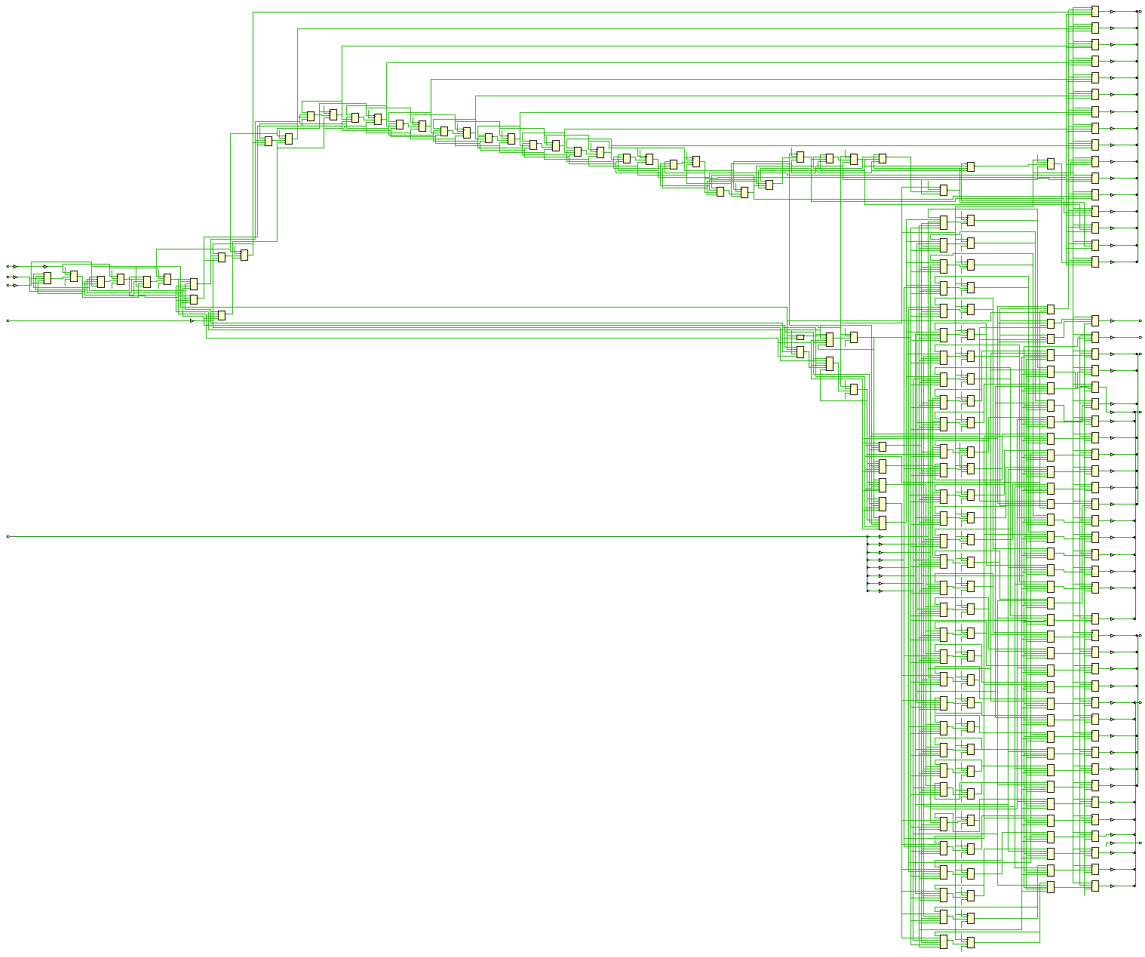
Gli stati della macchina in figura 1 sono formalmente descritti come segue:

- START: la macchina inizializza tutti i valori e attende che “i_start” diventi 1, quando ciò accade legge il primo bit di “i_w” e lo assegna a “sel_1”
- SELECTION2: viene letto il secondo bit di “i_w”, al ciclo di clock successivo, e viene assegnato a “sel_2”
- READ_W: se “i_start” è 0 si passa direttamente alla lettura da memoria dall’indirizzo 0000000000000000, altrimenti leggo il primo bit dell’indirizzo di memoria e passo allo stato CONCATENATION
- CONCATENATION: se “i_start” è 1 si itera sullo stato corrente, leggendo i bit di “i_w” finchè “i_start” non diventa 0. A quel punto si assegna a “o_mem_addr” il valore del registro data_reg che stavo utilizzando per costruire l’indirizzo di memoria e si passa allo stato successivo
- READ_RAM: “o_mem_en” è settato a 1, così da poter leggere dalla memoria allo stato successivo
- WAIT_READ: “o_mem_en” è risettato a 0, il valore dalla memoria viene letto
- DEMULTIPLEXING: il valore letto al ciclo precedente è ora assegnato a “i_mem_data” per essere indirizzato su uno dei canali di uscita. Sugli altri canali è mostrato il valore delle letture precedenti, salvato in degli appositi vettori. “o_done” viene settato a 1
- DONE: i bit di selezione sono inizializzati a 0, “o_done” è settato di nuovo a 0 e così anche le uscite del sistema.

2.2 Schema dell'implementazione

Segue lo schema circuitale, realizzato dal tool di sintesi Vivado (Figura 3)

Figura 3



3 Risultati sperimentali

3.1 Testing

Il componente è stato testato numerose volte, grazie ai test bench forniti e grazie all'utilizzo di uno script python per generare test casuali, sia in pre-sintesi che in post-sintesi. Inoltre, è stato testato il componente sia in presenza del segnale di reset, che in assenza di tale segnale. Tutti i test sono stati passati con successo.

I test sono stati utilizzati per coprire corner case e verificare la funzionalità del componente. È stato infatti fondamentale nel corso dello sviluppo del progetto analizzare con attenzione i singoli test forniti e i test generati, per individuare il flusso dei dati che il componente dovesse rispettare. La macchina a stati inziali, comprendente solo di sei stati, è stata ampliata grazie alle batterie di test,

che, evidenziando comportamenti specifici per particolari data set in ingresso, ci hanno permesso di introdurre stati di attesa o di modifica di eventuali segnali. Dopo la realizzazione di uno schema iniziale, grazie all'attenta lettura degli esempi forniti, è stato possibile stendere un codice che testato intensamente ha permesso di individuare limiti e punti di forza del componente realizzato.

3.2 Report di sintesi

La sintesi del componente riporta il seguente utilizzo di componenti:

- LUT: 89
- FF: 103

In Figura 4 si può vedere come non siano presenti latch, dal report di sintesi del tool Vivado.

Figura 4

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	89	0	134600	0.07
LUT as Logic	89	0	134600	0.07
LUT as Memory	0	0	46200	0.00
Slice Registers	103	0	269200	0.04
Register as Flip Flop	103	0	269200	0.04
Register as Latch	0	0	269200	0.00
F7 Muxes	0	0	67300	0.00
F8 Muxes	0	0	33650	0.00

Analizzando l'aspetto temporale si può verificare che il constraint imposto dal clock è rispettato, quindi i vincoli sono soddisfatti.

4 Conclusioni

Il componente è stato realizzato secondo le specifiche progettuali, per poi essere testato sia casualmente che mettendolo alla prova tramite la sottomissione di corner case più specifici. Per rispettare l'intrinseca caratteristica dei segnali, che quindi vengono letti in un ciclo di clock e mostrati al successivo, è stato necessario aggiungere nella FSM degli stati che rappresentassero la lettura effettiva dei dati e permettessero il corretto aggiornamento dei segnali. Possiamo quindi affermare che il progetto sia stato portato avanti con costanza e continuità, per permetterci di stressarlo nei punti più critici e testarne il funzionamento.

La comprensione completa del linguaggio e delle caratteristiche di simulazione del tool di sintesi sono state infatti fondamentali per il completamento del nostro progetto e l'analisi attenta dei test bench forniti ci hanno permesso di comprendere come il linguaggio VHDL in cui è stato sviluppato

il componente sia un linguaggio di basso livello, e come fosse quindi necessario ragionare in un'ottica più architetturale e meno di alto livello.