Prova Finale
Progetto di Reti Logiche

A.A 2022-2023

Alibeaj Michael (Codice Persona 10750827 - Matricola 959450)
Chiaverini Raffaele (Codice Persona 10711746 - Matricola 955999)

Aprile, 2023

Politecnico di Milano

1 Sommario

2 Introduzione

2.1 Obiettivo

Implementare un algoritmo che permette al sistema di riceve indicazioni circa una locazione di memoria, il cui contenuto deve essere indirizzato verso un canale di uscita fra i quattro disponibili. Le indicazioni circa il canale da utilizzare e l'indirizzo di memoria a cui accedere vengono forniti mediante un ingresso seriale da un bit, mentre le uscite del sistema, ovvero i succitati canali, forniscono tutti i bit della parola di memoria in parallelo.

2.2 Dettagli Implementativi

Il modulo implementato presenta:

- 2x Ingressi Primari ciascuno da 1 bit: *W* e *START*
- 5x Uscite Primarie ciascuna da 8 bit: *Z0, Z1, Z2, Z3*
- 1x Uscita Primaria da 1 bit: *DONE*
- 2x Segnali unici per tutto il sistema: *CLOCK* e *RESET*

Inoltre, valgono le seguenti considerazioni secondo specifica:

- Il modulo deve essere progettato considerando che prima del primo START=1 (e prima di richiedere il corretto funzionamento del modulo) verrà sempre dato il RESET (RESET=1). Una seconda (o successiva) elaborazione con START=1 non dovrà invece attendere il reset del modulo. Ogni qual volta viene dato il segnale di RESET (RESET=1), il modulo viene re-inizializzato.
- La sequenza di ingresso è valida quando il segnale START è alto (=1) e termina quando il segnale START è basso (=0). Il segnale START rimane alto per almeno di 2 cicli di clock e non più di 18 cicli di clock (2 bit del canale e 16 bit per il massimo numero di bit per indirizzare la memoria). Si assuma questa condizione sempre verificata (non è necessario gestire il caso in cui il segnale di START rimanga attivo meno di 2 cicli di clock o più di 18).
- Il segnale START è garantito rimanere a 0 fino a che il segnale DONE non è tornato a 0. Il tempo massimo per produrre il risultato deve essere inferiore a 20 cicli di clock.
- Le uscite Z0, Z1, Z2 e Z3 sono inizialmente 0. I valori rimangano inalterati eccetto il canale sul quale viene mandato il messaggio letto in memoria; i valori sono visibili solo quando il valore di DONE è 1.
- Quando il segnale DONE è 0 tutti i canali Z0, Z1, Z2 e Z3 devono essere a zero. Contemporaneamente alla scrittura del messaggio sul canale, il segnale DONE passa da 0 passa a 1 e rimane attivo per un solo ciclo di clock (dopo 1 ciclo di clock DONE passa da 1 a 0).
- I dati in ingresso, ottenuti come sequenze sull'ingresso primario seriale W lette sul fronte di salita del clock, sono organizzati nel seguente modo:
    - 2x bit di Intestazione (i primi della sequenza), che permettono di identificare il canale di uscita Zi sul quale deve essere indirizzato il messaggio
    - Nx bit di indirizzo della memoria (in cui è memorizzato il messaggio da 8 bit che deve essere indirizzato verso il canale di uscita).
- Gli N bit di indirizzo possono variare da 0 fino ad un massimo di 16 bit. Gli indirizzi di memoria sono tutti di 16 bit, in caso contrario l'indirizzo viene esteso con 0 sui bit più significativi
- Tutti i bit su W devono essere letti sul fronte di salita del clock

3 Architettura

3.1 Strumenti di Sintesi Utilizzati

- Xilinx Vivado ML Edition
- Target Artix-7 FPGA xc7a200tfbg484-1

3.2 Descrizione Implementazione

Facciamo uso di un unico processo (lambda_delta) nel quale gestiamo sia la funzione delta che lambda della Macchina a Stati Finiti di **Moore**.

La “Sensitivity List” è costituita dai segnali di START, RESET e CLOCK.

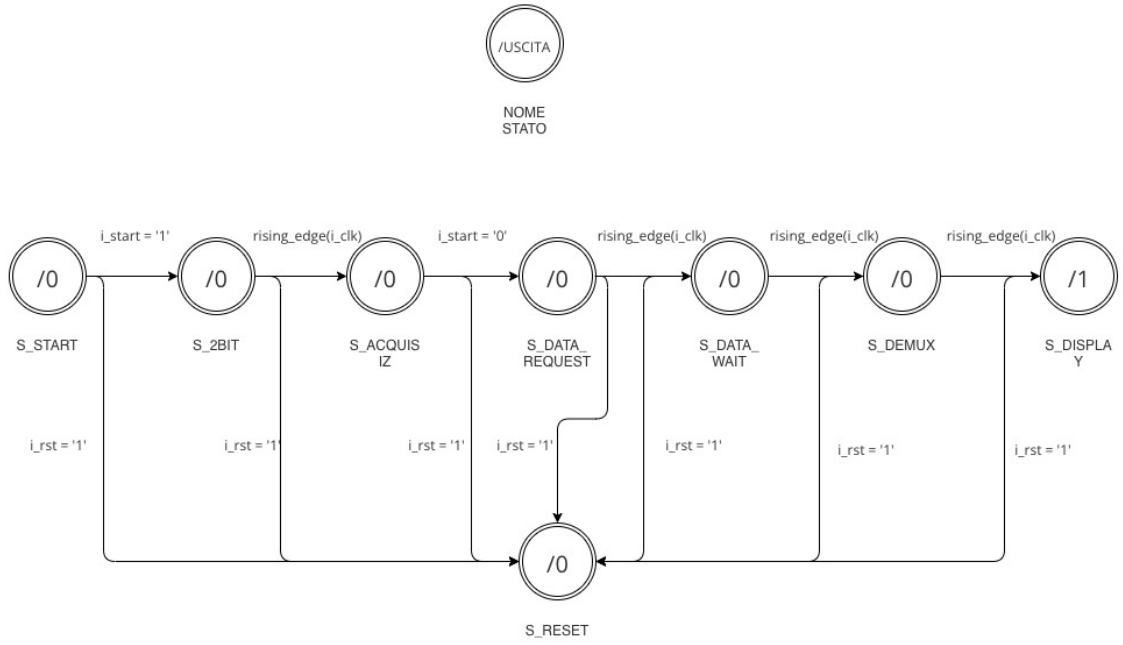

Figura 1: FSM di Moore.

Descrizione dei Segnali

- **i_clk:** segnale di CLOCK in ingresso generato dal Test Bench.
- **i_rst:** segnale di RESET che inizializza la macchina pronta per ricevere il primo segnale di START.
- **i_start:** segnale di START generato dal Test Bench.
- **i_w:** 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:** segnale di uscita che comunica la fine dell’elaborazione.
- **o_mem_addr:** segnale (vettore) di uscita che manda l’indirizzo alla memoria.
- **i_mem_data:** segnale (vettore) che arriva dalla memoria in seguito ad una richiesta di lettura.
- **o_mem_en:** segnale di ENABLE da dover mandare alla memoria per poter comunicare (sia in lettura che in scrittura).
- **o_mem_we:** segnale di WRITE ENABLE da dover mandare alla memoria (=1) per poter scriverci. Per leggere da memoria esso deve essere 0.
- **sel:** segnale (vettore) che contiene i bit di intestazione necessari a pilotare il canale di uscita.
- **address:** segnale(vettore) che contiene i bit che costituiscono l’indirizzo di memoria dal quale bisogna attingere per leggere il dato da mandare in uscita.
- **w0, w1, w2, w3:** segnali (vettori) che conservano i dati recentemente acquisiti, che devono essere trasmessi ai rispettivi canali di uscita z0, z1, z2, z3 per evitare che questi ultimi vengano sovrascritti quando o_done assume un valore logico basso.

Descrizione degli Stati

- **S_START:** Stato in cui si inizializzano i valori dei vari registri & segnali presenti. In particolare, si portano al valore logico basso i segnali di *o_mem_en*, o *mem_wen*, *o_done* o*_mem_address* e si portano a una configurazione di soli 0 bit i restanti segnali.
- **S_2BIT:** Stato in cui si acquisiscono i 2 bit di intestazione necessari a pilotare il canale di uscita sul quale verrà trasmesso il messaggio.
- **S_ ACQUISITION:** Stato in cui si acquisiscono i restanti bit che vanno a costituire l'indirizzo di memoria e dal quale si leggerà il dato da trasmettere in uscita. I bit vengono memorizzati nel segnale *address* e vengono elaborati con operazioni di shift & estensione in segno.
- **S_DATA_REQUEST:** Stato in cui si effettua una richiesta di lettura alla memoria all'indirizzo specificato dal segnale *o_mem_address*. In particolare, in questo stato portiamo a livello logico alto il segnale *o_mem_en*.
- **S_DATA_WAIT:** Stato in cui si attende un ciclo di clock per avere in uscita dalla memoria il dato da trasmettere in uscita.
- **S_DEMUX:** Stato in cui si pilota il canale di uscita in base ai bit di intestazione acquisiti in precedenza. In base al segnale sel andiamo a selezionare il canale di uscita nel quale verrà mostrato il messaggio.
- **S_DISPLAY:** Stato in cui si visualizzano i valori aggiornati delle uscite nei rispettivi canali. In questo stato portiamo a livello logico alto il segnale *o_done*.
- **S_RESET:** Stato in cui si re-inizializzano i valori dei vari registri & segnali presenti ai valori di default.

Abbiamo fissato il valore del segnale o_mem_wen al valore logico basso in quanto la nostra macchina, nel suo complesso, non va mai a scrivere un dato in memoria.

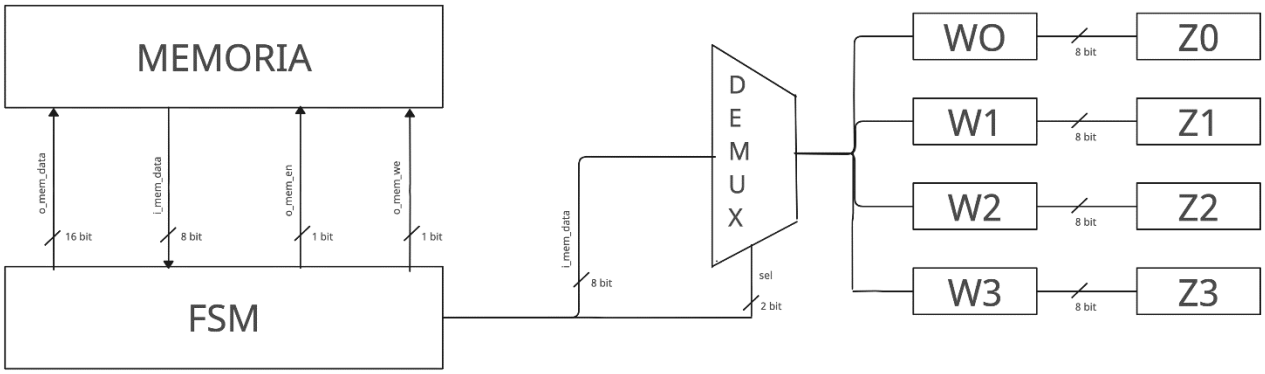

Figura 2: DataPath.

4 Risultati Sperimentali

4.1 Sintesi

La sintesi e l'implementazione sono state fatte usando il software a disposizione Xilinx Vivado Web pack e la FPGA target usata è stata la Artix-7, xc7a200tfbg484-1. Per la realizzazione non sono stati forniti vincoli stringenti sull'area di utilizzo e sul tempo di esecuzione; dunque, non si è badato ad ottimizzare molto questi componenti. Di seguito sono riportati i componenti della FPGA utilizzati.

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs*	51	0	0	134600	0.04
LUT as Logic	51	0	0	134600	0.04
LUT as Memory	0	0	0	46200	0.00
Slice Registers	108	0	0	269200	0.04
Register as Flip Flop	108	0	0	269200	0.04
Register as Latch	0	0	0	269200	0.00
F7 Muxes	0	0	0	67300	0.00
F8 Muxes	0	0	0	33650	0.00

Figura 3: Report di sintesi.

Per quanto riguarda il design timing summary, si sono rilevati un Worst Negative Slack di 94.936 ns ed un Worst Pulse Width Slack di 4.5 ns. Complessivamente il risultato rientra nei limiti del funzionamento.

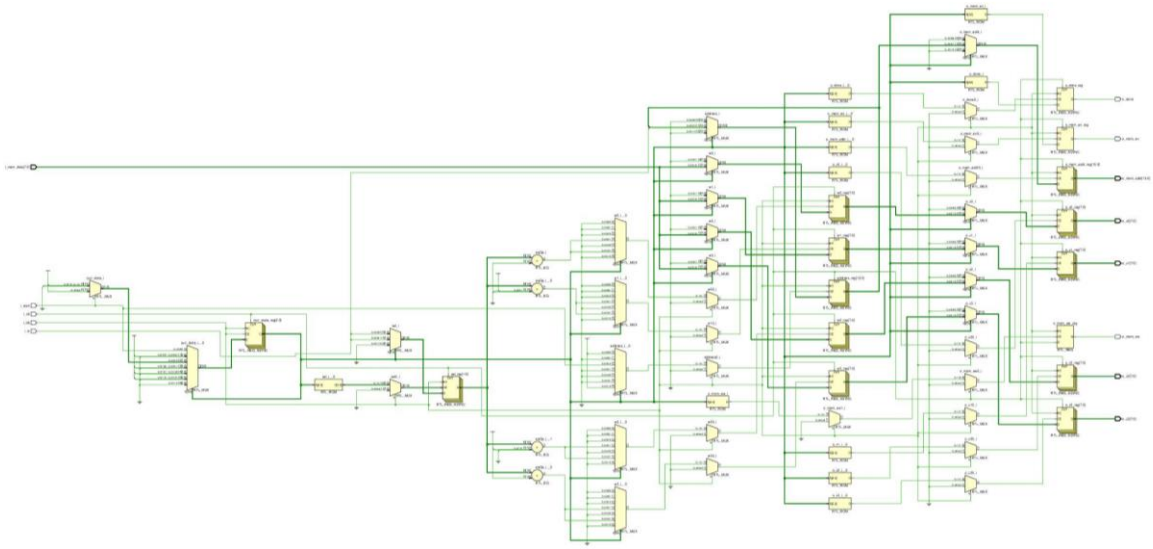

Figura 4: Schema dell'implementazione: 260 cells, 38 I/O ports, 274 nets.

Altre caratteristiche, come l'on-chip power (potenza), sono comunque piccole in confronto con le risorse a disposizione, vista la ridotta complessità del progetto. Il totale on-chip power è 0.132 W.

4.2 Simulazioni

Abbiamo verificato il corretto funzionamento del componente utilizzando sia i test benches forniti che altri da noi pensati per stressare particolari casi della computazione. Si è cercato di coprire tutti i possibili cammini che il componente potesse intraprendere durante il funzionamento: osservando la forma d'onda dei vari segnali, ci siamo accertati che i segnali ricevuti e trasmessi avessero un andamento conforme alla specifica.

- **Test Reset:** Si tratta di un test che va a verificare la corretta gestione del segnale di **i_rst**. Il segnale comporta la re-inizializzazione della computazione andando quindi a portare i valori dei vari segnali al valore di default.

- **Test Bench 1:**

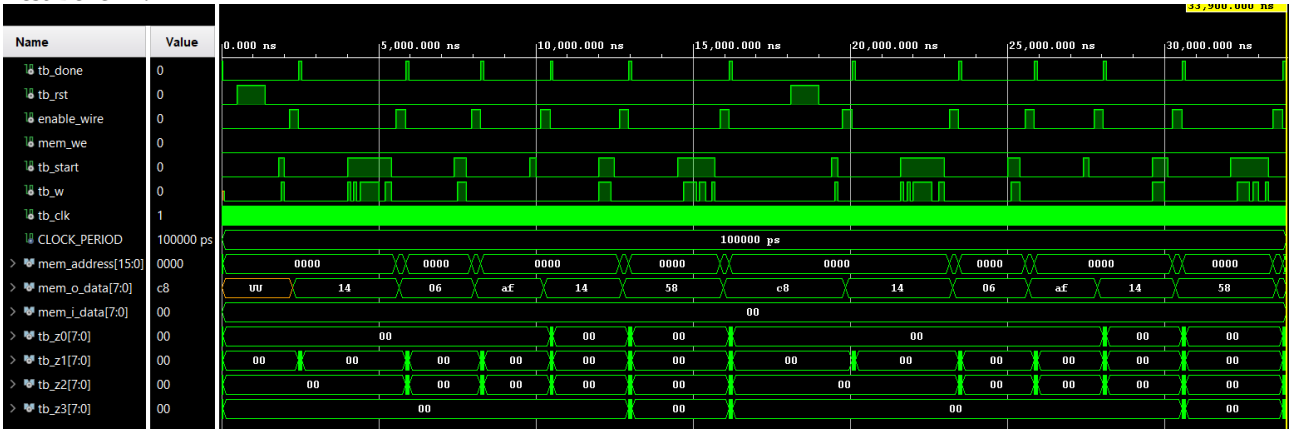

Figura 4.1: Estratto del Test Bench.

- **Test Bench 2:**

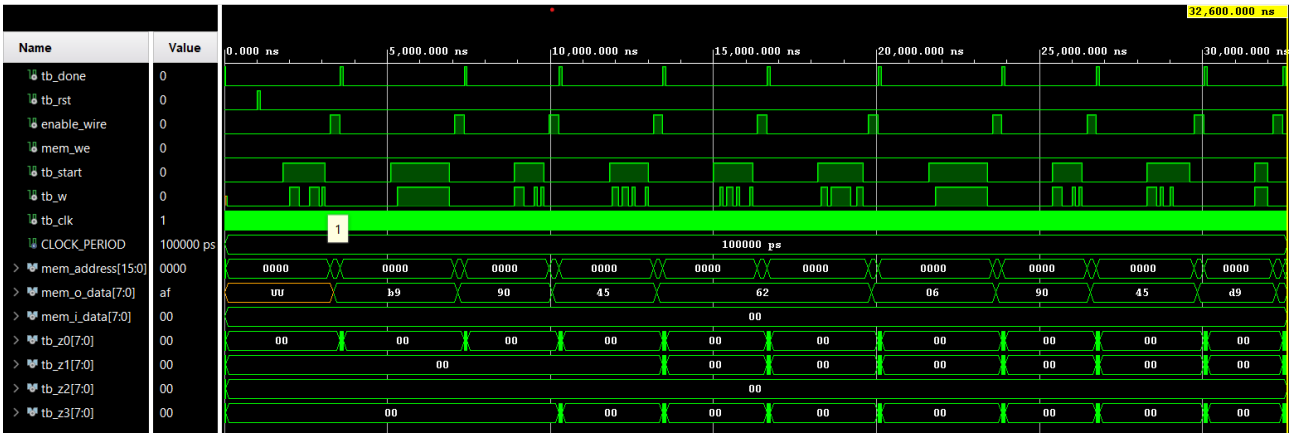

Figura 4.2: Estratto del Test Bench.

- **Test Bench 3:**

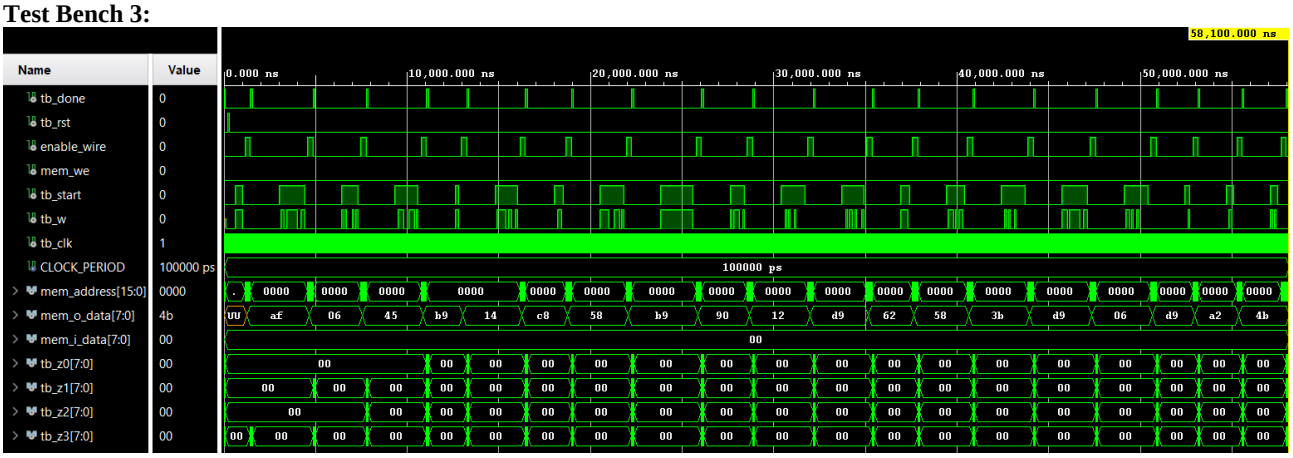

Figura 4.3: Estratto del Test Bench.

- **Test Bench 4:**

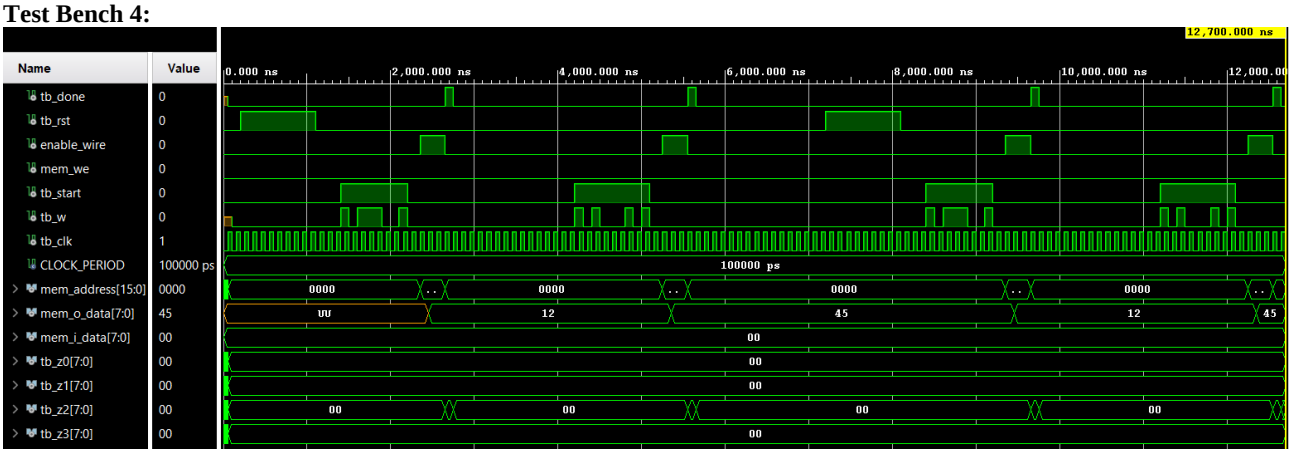

Figura 4.4: Estratto del Test Bench.

- **Test Bench 5:**

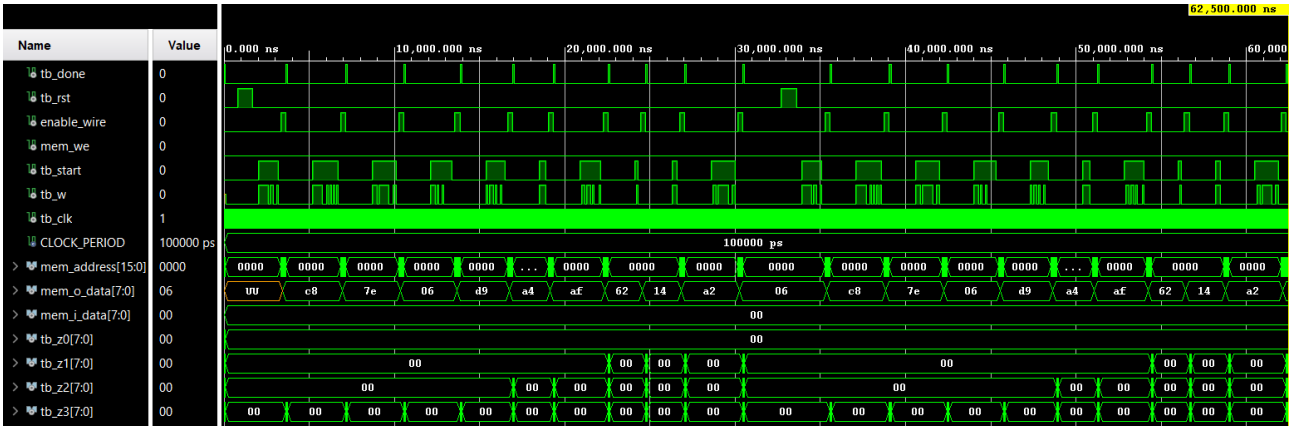

Figura 4.5: Estratto del Test Bench.

- **Test Bench 6:**

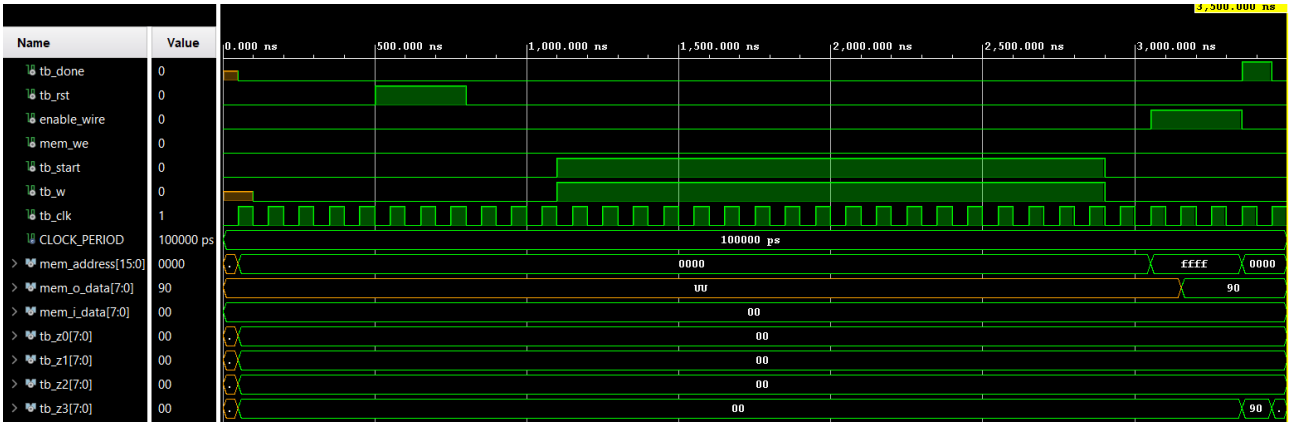

Figura 4.6: Estratto del Test Bench.

- **Test Bench 7:**

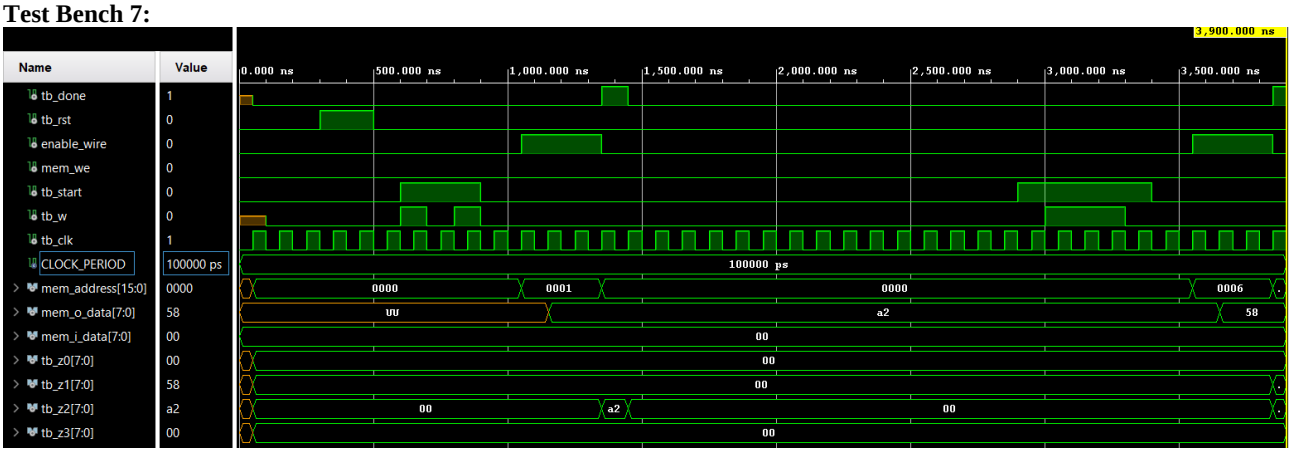

Figura 4.d: Estratto del Test Bench.

5 Considerazioni finali

Si ritiene che l'architettura progettata rispetti le specifiche fornite, cosa che è stata verificata mediante test che andavano a coprire una buona parte dei casi limite. Oltre a ciò, abbiamo impiegato la maggior parte del tempo alla progettazione dell'architettura (in modo tale da evitare anche i LATCH). Dal punto di vista del design, abbiamo optato per una FSM priva di componenti esterni. Il componente, come previsto, non utilizza che una piccola percentuale dell'intera FPGA e il ritardo massimo dell'esecuzione rientra largamente all'interno del margine di clock richiesto.