Progetto di Reti Logiche AA 2022-2023

Prof. Gianluca Palermo

Luca Simei
`Codice persona: 10714016`

Alessio Spineto
`Codice persona: 10739526`

Marzo 2023

POLITECNICO
MILANO 1863

Indice

1 Introduzione

1.1 Scopo del progetto

Lo scopo del progetto è quello di realizzare un modulo che riceve i seguenti segnali di ingresso e produce le seguenti uscite:

- due ingressi primari da 1 bit: W e START
- un segnale di clock unico: CLK
- un segnale di reset unico: RESET
- quattro uscite primarie da 8 bit: Z0, Z1, Z2, Z3
- una uscita primaria da 1 bit: DONE

Ogni evento del circuito avviene in corrispondenza del fronte di salita del segnale di clock. All'istante iniziale, relativo al segnale di RESET, le uscite da 8 bit ciascuna sono tutte inizializzate a zero ed il segnale di DONE è 0. Il funzionamento del circuito è il seguente: appena il segnale di START diventa 1, comincia ad essere letta la sequenza di bit dell'ingresso primario W. Nello specifico, come abbiamo detto, viene letto un bit ad ogni fronte di salita del ciclo di clock. Verrà letto un minimo di 2 bit ed un massimo di 18 bit, fintanto che il segnale di START è 1: verrà letto un numero inferiore di bit nel caso in cui il segnale di START diventa 0 prima di 18 cicli di clock da quando è diventato alto. In questo caso, la sequenza in ingresso viene estesa con degli 0 sui bit più significativi, fino ad ottenere una stringa di ingresso della lunghezza di 18 bit. Di tale sequenza, i primi due bit rappresenteranno l'uscita del nostro messaggio: rispettivamente:

- 00: Z0
- 01: Z1
- 10: Z2
- 11: Z3

Gli altri 16 bit della sequenza rappresentano invece l'indirizzo di memoria dal quale prendere un valore in binario. A questo punto, tale valore viene scritto sul canale di uscita precedentemente individuato. Contemporaneamente, il segnale di DONE diventa 1 per un solo ciclo di clock. Fino a quando il segnale di DONE non è tornato a 0, il segnale di START è garantito essere 0. Il tempo trascorso per produrre il risultato sull'uscita, quindi da quando viene preso il messaggio dalla memoria a quando viene scritto sull'uscita, deve essere inferiore a 20 cicli di clock.

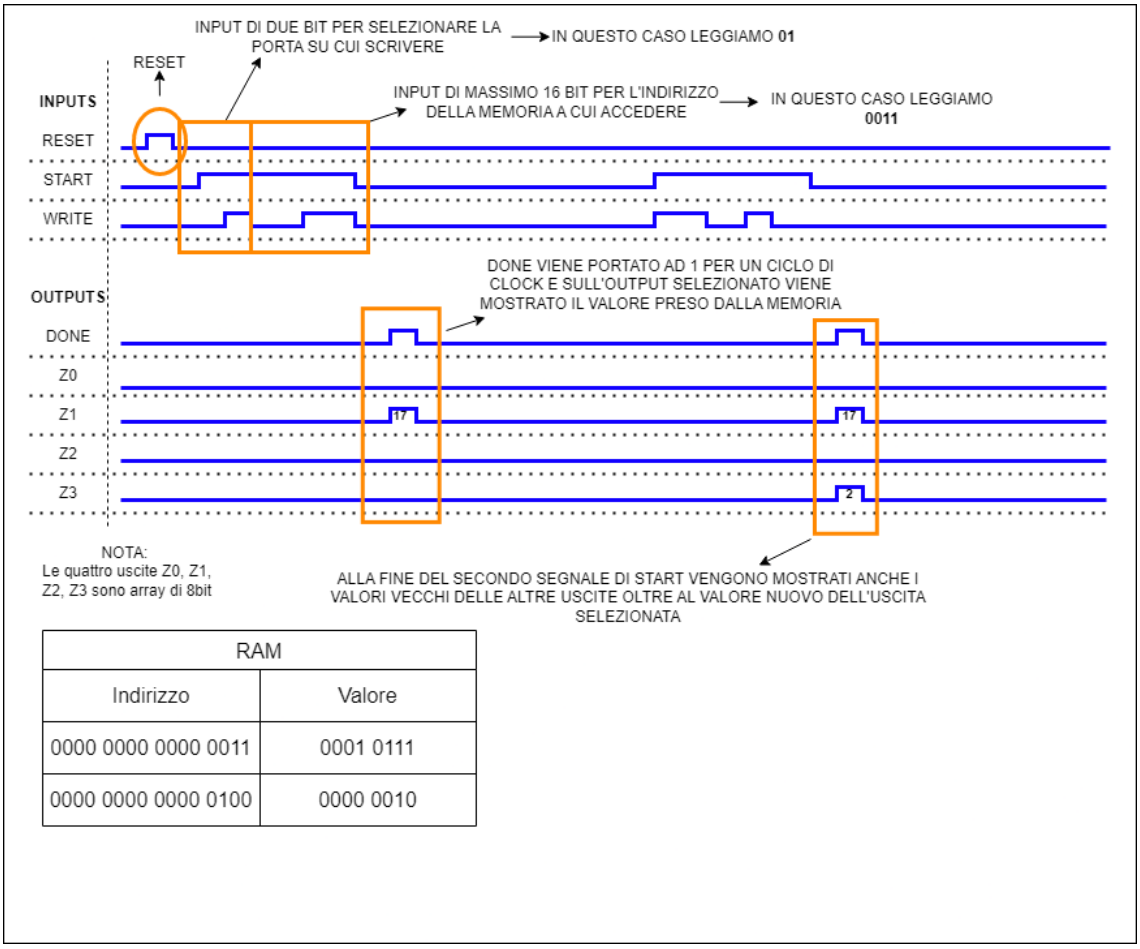
1.2 Esempio di funzionamento

Si consideri una situazione iniziale, in cui il segnale di RESET ha inizializzato tutte le uscite. START dopo un istante di tempo t passa da 0 a 1, e rimane tale per sei cicli di clock. Vengono quindi letti sei bit in ingresso dal segnale W: nell'esempio, viene letta, un bit alla volta, partendo dal bit più significativo, la sequenza 010011. I primi due bit letti individuano l'uscita: nel nostro caso 01, quindi Z1. Gli altri quattro bit della sequenza vengono allungati con dei bit 0 nelle posizioni piú significative in modo da ottenere una sequenza di 16 bit. L'indirizzo così ottenuto

è 0000000000000011. Ciò significa che andrà indirizzato verso l'uscita Z1 il messaggio contenuto all'indirizzo di memoria 0000000000000011. Tale messaggio sarà una sequenza binaria di 8 bit, per esempio 00010111. Ora il segnale di START è 0, e entro 20 cicli di clock il segnale di DONE diverrá 1. Questo significa che é dato in output, sull'uscita Z1, il valore contenuto nell'indirizzo di memoria individuato. Il segnale di DONE rimarrà 1 per un solo ciclo di clock. Quello che si vedrà sarà quindi:

- Z0: 00000000
- Z1: 00010111
- Z2: 00000000
- Z3: 00000000

É importante osservare il fatto che fintanto che DONE è 0, sulle uscite si vedranno solo stringhe inizializzate a zero (00000000), mentre quando DONE è 1 si vedrà su ogni canale l'ultimo valore ad esso trasmesso. Si puó osservare graficamente l'esempio descritto nella pagina seguente:



RAM	
Indirizzo	Valore
0000 0000 0000 0011	0001 0111
0000 0000 0000 0100	0000 0010

Figure 1: Diagramma di flusso

2 Architettura

Per la realizzazione del progetto, abbiamo deciso di optare su una macchina a stati finiti che regoli il funzionamento del circuito. In particolare, la transizione da uno stato all'altro di tale macchina dipende dallo stato corrente in cui si trova la macchina e in alcuni casi dal valore del segnale START. Gli stati vengono aggiornati rispetto al segnale di clock. Tale macchina a stati finiti presenta ben 5 stati, e abbiamo deciso di utilizzare nel nostro codice la codifica a minor numero di bit per rappresentarli.

2.1 FSM

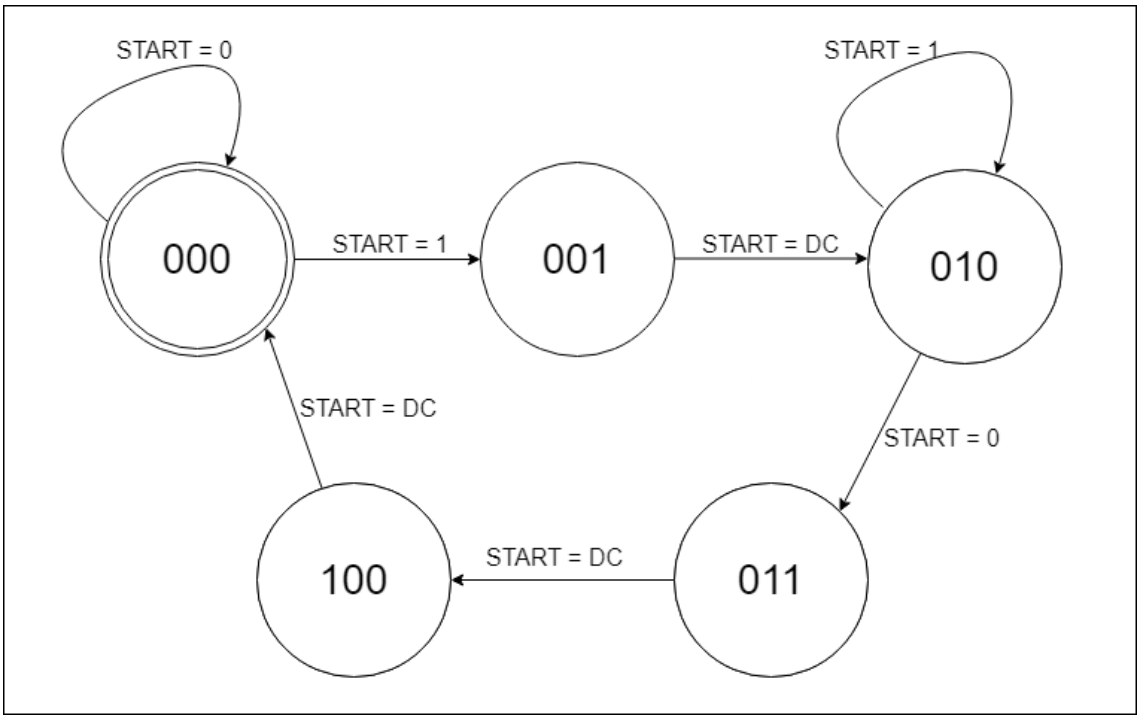


Figure 2: Diagramma FSM

2.1.1 Stato 000

Siamo nello stato iniziale. In questo stato il segnale di DONE é portato a 0, e analogamente le quattro uscite sono portate a 00000000. Nel momento in cui il segnale di START passa da 0 a 1 viene letto un bit dall'ingresso W. In particolare, il bit letto é il primo dei due bit che rappresentano l'uscita verso cui indirizzare il nostro messaggio. Quindi, se da input leggiamo 1 l'uscita sarà temporaneamente 01; se invece il segnale W è 0 l'uscita sarà inizialmente 00. Lo stato rimane tale fino al sopraggiungere di un segnale alto di START: in questo caso lo stato viene mantenuto per un ciclo di clock in modo da leggere l'input, passando poi allo stato 001.

2.1.2 Stato 001

In questo stato viene eseguita la lettura da input W del bit meno significativo dei due bit che rappresentano l'uscita verso cui indirizzare il nostro messaggio. Sul segnale ottenuto nello stato precedente viene prima applicato uno shift logico verso sinistra, e poi viene sommato il bit letto in input. A questo punto sappiamo su quale uscita scrivere il nostro valore e si passa allo stato 010. Anche in questo stato la lettura da input W e il salvataggio del nostro bit sfrutta un solo ciclo di clock.

2.1.3 Stato 010

Fintanto che START è 1, viene letto da input W una sequenza di bit in ordine decrescente di significatività. Tale sequenza rappresenterà l'indirizzo di memoria da cui estrarre il valore da indirizzare verso l'uscita. Si ricordi che per questo stato verrà letto un bit dall'input per ogni ciclo di clock, per un minimo di 0 e un massimo di 16 cicli di clock. Nel momento in cui START passa da 1 a 0, termina la lettura dei bit in ingresso. Per realizzare l'indirizzo di memoria di 16 bit, il meccanismo a livello implementativo é analogo a quello usato nello stato precedente: inizializzata una variabile `in_address` a 0000000000000000, viene salvato ad ogni ciclo di clock il bit letto in ingresso nella posizione meno significativa. Dal secondo bit letto in poi, viene eseguito uno shift logico a sinistra sul segnale e viene poi sommato il bit letto. In questo modo, una volta finita la fase di lettura si otterrá l'indirizzo di memoria già prolungato a 16 bit, con bit pari a 0 nelle posizioni più significative. Nel momento in cui START passa da 1 a 0, la macchina passa allo stato 011.

2.1.4 Stato 011

Quest stato non é strettamente necessari al funzionamento del circuito. É stato aggiunto in un secondo momento rispetto allo sviluppo della FSM per coonsentire la corretta sintetizzazione del circuito. In questo stato viene selezionato dalla memoria il valore corrispondente all'indirizzo trovato nello stato precedente. Il valore ottenuto viene poi memorizzato nel segnale old dell'uscita selezionata (ad esempio se viene selezionata l'uscita Z0, il valore sará memorizzato nel segnale `old_z0`). Si passa quindi all'ultimo stato.

2.1.5 Stato 100

In questo stato DONE viene portato ad 1, mentre le uscite passano da 00000000 al valore contenuto nel loro rispettivo segnale old. Lo stato viene mantenuto per un solo ciclo di clock per poi tornare allo stato iniziale 000.

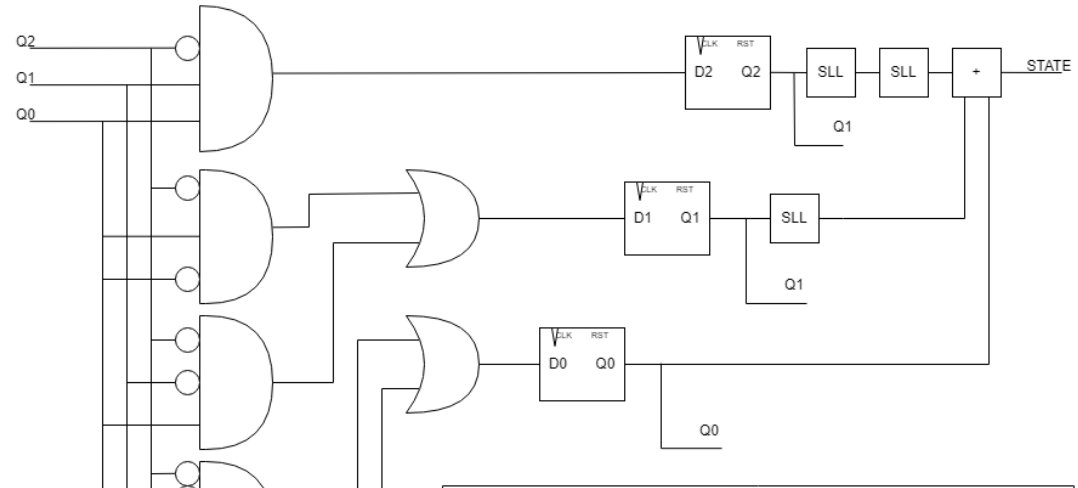


Tabella delle transizioni e delle eccitazioni (FF di tipo D)

	START=0	START=1	START=0	START=1
Q2 Q1 Q0	Q2* Q1* Q0*	Q2* Q1* Q0*	D2 D1 D0	D2 D1 D0
0 0 0	0 0 0	0 0 1	0 0 0	0 0 1
0 0 1	0 1 0	0 1 0	0 1 0	0 1 0
0 1 0	0 1 1	0 1 0	0 1 1	0 1 0
0 1 1	1 0 0	1 0 0	1 0 0	1 0 0
1 0 0	0 0 0	0 0 0	0 0 0	0 0 0

Figure 3: Diagramma del circuito dell'FSM

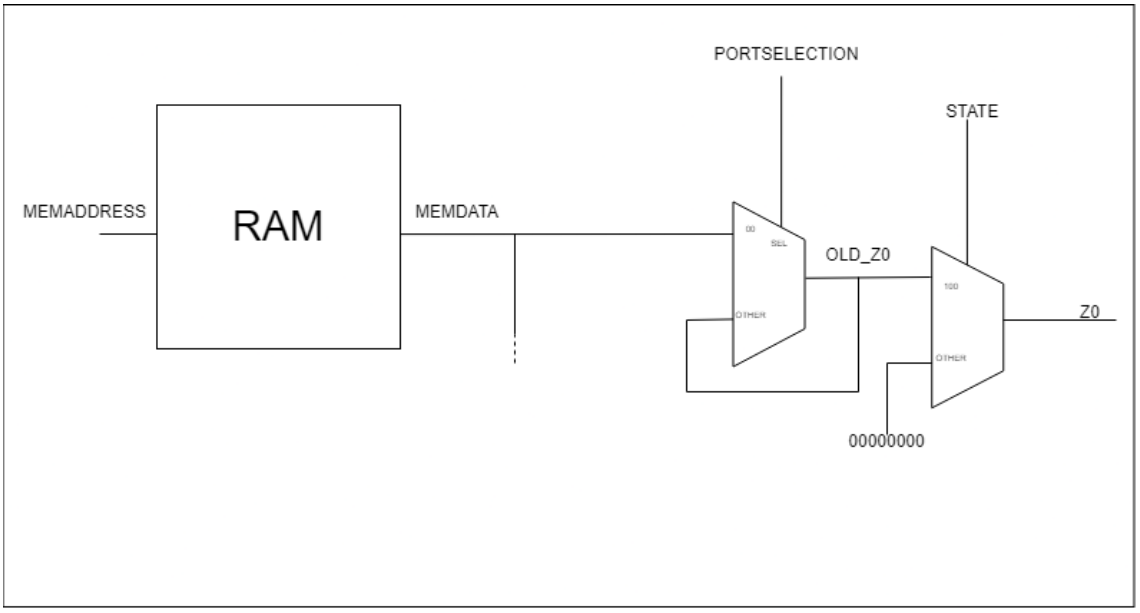


Figure 4: Diagramma output

3 Risultati Sperimentali

In questa sezione saranno presentati i risultati ottenuti dalla sintetizzazione del circuito descritto in VHDL e come si é arrivati a tali risultati. Verrá infine descritto lo sviluppo della fase di simulazioni del circuito, dalla creazione dei testbench utilizzati allo svolgimento stesso delle simulazioni.

3.1 Sintesi

Usufruendo dello strumento incorporato di Vivado é stato verificato che il circuito sia sintetizzabile e successivamente ne é stata realizzata la sintesi vera e propria utilizzando la strategia default di Vivado sia per la sintesi stessa, sia per i report. Grazie alla sintesi, tutti i testbench che verranno approfonditi nel prossimo paragrafo sono stati effettuati sia in simulazione post-sintesi funzionale (Post-Synthesis Functional Simulation), che in simulazione post-sintesi sincronizzata (Post-Synthesis Timing Simulation). É stato utilizzato lo strumento di Report Timing Summary per verificare la corretta sintetizzazione del circuito ed é stato applicato un delay di 1ns al segnale `i_mem_data` in uscita dalla memoria per la simulazione sincronizzata. Per sintetizzare correttamente il circuito, come é stato giá accenato nella sezione relativa alla FSM, é stato inserito uno stadio (lo stato 011) aggiornato al fronte di discesa del clock, il quale assicura che l'acquisizione del segnale in uscita dalla memoria avvenga corretamente e senza instabilitá durante la Simulazione Post-Sintesi Sincronizzata.

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	48.200 ns	Worst Hold Slack (WHS):	0.061 ns	Worst Pulse Width Slack (WPWS):	49.650 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	170	Total Number of Endpoints:	170	Total Number of Endpoints:	90

All user specified timing constraints are met.

Figure 5: Sommario report timing

Per verificare il corretto funzionamento del circuito é stato inserito sotto forma di costraint un ritardo di propagazione rispetto al clock del segnale in uscita dalla memoria, la cui specifica é la seguente:

```
1 create_clock -period 100.000 -waveform {50.000 100.000} [get_ports i_clk]
2 set_input_delay -clock [get_clocks *] 1.000 [get_ports {{i_mem_data[0]} {i_mem_data[1]}
3  {i_mem_data[2]} {i_mem_data[3]} {i_mem_data[4]} {i_mem_data[5]} {i_mem_data[6]} {i_mem_data[7]}}]
```

Figure 6: Specifica del costraint

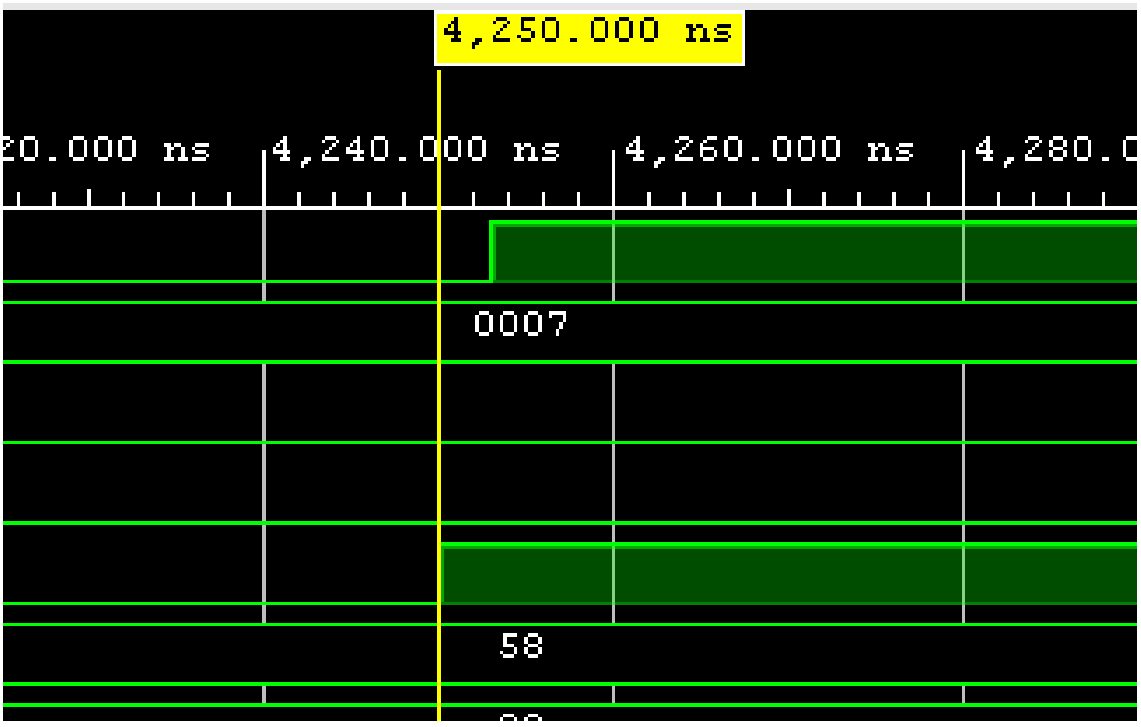


Figure 7: Particolare di una simulazione in cui é presente il ritardo di propagazione.
Il segnale superiore rappresenta l'andamento del segnale DONE, mentre il segnale inferiore rappresenta l'andamento del segnale di clock.

3.1.1 Componenti utilizzate

Il risultato finale della sintesi é stato simulato considerando come target un FPGA della famiglia Kintex-7, in particolare il modello xc7k70tfbv676-1, già presente nelle librerie standard di Vivado 2022.2. Dal report di sintesi vediamo che sono stati utilizzati come componenti 44 LUT e 89 FF:

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs*	44	0	0	41000	0.11
LUT as Logic	44	0	0	41000	0.11
LUT as Memory	0	0	0	13400	0.00
Slice Registers	89	0	0	82000	0.11
Register as Flip Flop	89	0	0	82000	0.11
Register as Latch	0	0	0	82000	0.00
F7 Muxes	0	0	0	20500	0.00
F8 Muxes	0	0	0	10250	0.00

Figure 8: Particolare del report di sintesi

3.2 Simulazioni

Per simulare il circuito descritto é stato utilizzato in fase di sviluppo il set di testbench fornito, ricevendo un feedback sui cambiamenti apportati al codice. Una volta superati tali test, siamo poi passati alla fase di sviluppo dei testbench, trovando casi limite e casi particolari per assicurare il funzionamento corretto del circuito sotto le specifiche di progetto. Abbiamo così prodotto 3 testbench. Tutti e tre i test verificano che l'ultimo valore dato in output di ogni uscita venga mantenuto e che il numero di cicli di clock fra un fronte di discesa del segnale di START ed il successivo fronte di salita del segnale in output DONE sia minore di 20.

3.2.1 Reset TestBench

Il primo test riguarda la risposta corretta del circuito ad un segnale di RESET. Secondo le specifiche é sicuramente presente un segnale alto di RESET antecedente al primo segnale alto di START, ma sono possibili casi in cui vengono ricevuti in input uno o piú segnali alti di RESET fra un segnale alto di START ed il successivo. Il testbench ha verificato tre scenari:

- nel primo viene testato il funzionamento del circuito al ricevimento del primo ed iniziale segnale alto di RESET,
- nel secondo scenario invece il testbench verifica che il circuito risponda correttamente alla presenza di un segnale alto di RESET presente fra due segnali alti di START,
- infine nel terzo scenario del testbench fra un segnale alto di START ed il successivo sono presenti due segnali alti distinti di RESET.

In ciascun scenario il circuito ha risposto correttamente. I risultati sono riportati nell'immagine seguente:.

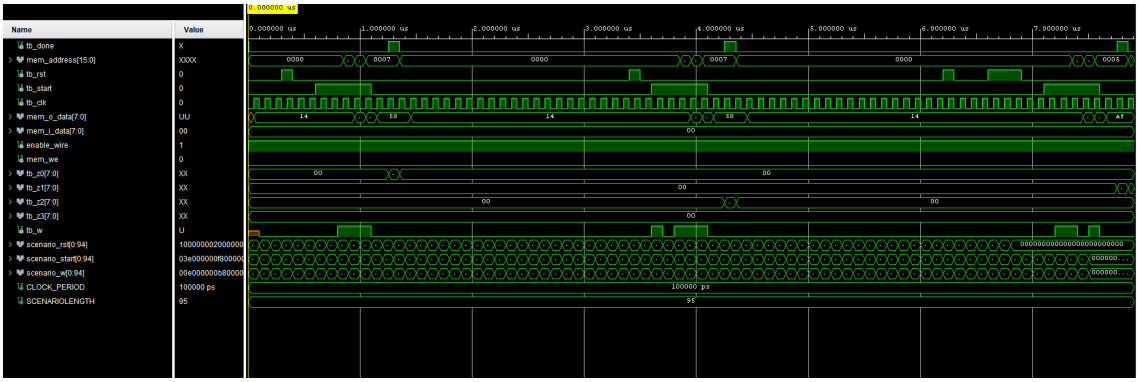


Figure 9: Risultati Testbench 1

3.2.2 Overwrite and Extension TestBench

Il secondo testbench verifica due proprietá del circuito necessarie per rispettare le specifiche del progetto:

1. l'indirizzo letto dal segnale W deve essere necessariamente formattato in 16 bit, ponendo dei bit 0 nelle posizioni più significative

2. in caso di selezione di una porta che é stata precedentemente oggetto di output, il valore deve essere sostituito con il nuovo valore letto dalla memoria

Il testbench fornisce quindi in input al circuito diversi indirizzi di memoria tramite il segnale W, tutti di lunghezza variabile e sempre strettamente maggiore di 0 e strettamente minore di 16 bit: questo processo é ripetuto per ogni porta di uscita e due volte per porta con indirizzi differenti in modo da verificare la seconda proprietá sopra riportata.

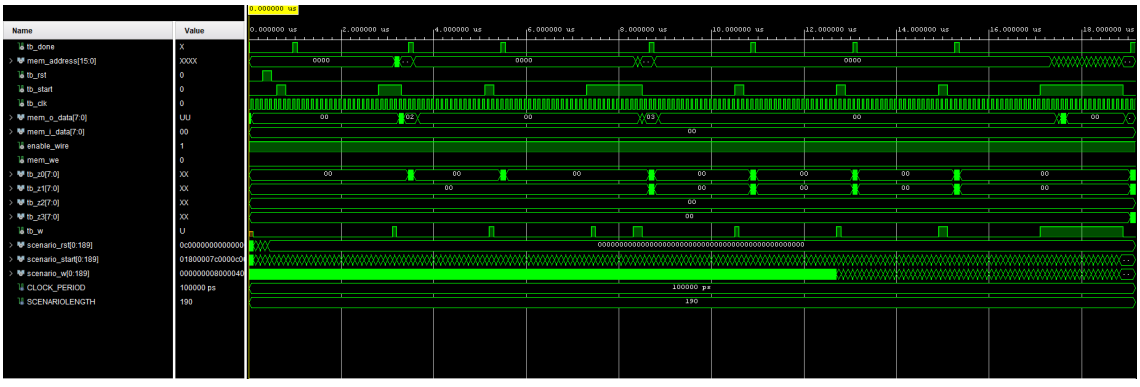


Figure 10: Risultati Testbench 2

3.2.3 Min and Max Values of START TestBench

Il terzo ed ultimo testbench verifica due condizioni limite del segnale di START derivate dalle specifiche di progetto:

- segnale di START alto della durata di 2 cicli di clock (durata minima possibile)
- segnale di START alto della durata di 18 cicli di clock (durata massima possibile)

Questo testbench verifica inoltre che il circuito si comporti correttamente in caso il valore in output della memoria sia identico al valore dato in output precedentemente sulla stessa porta di output.

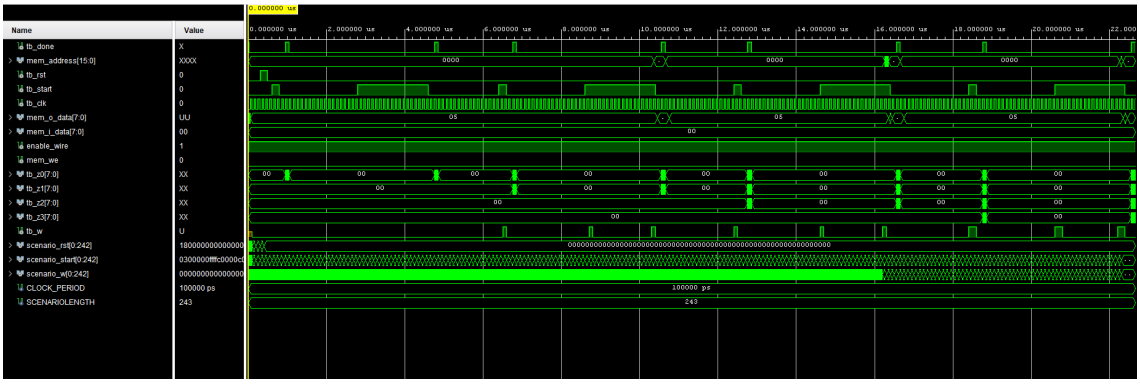


Figure 11: Risultati Testbench 3

3.3 Link ai vari testbench in formato VHD:

https://tinyurl.com/PRL10714016-10739526

4 Conclusioni

Come mostrano i risultati dei testbench, sia in pre che in post sintesi, il circuito descritto soddisfa notevolmente le specifche di progetto: infatti é immediato notare che non sussiste la necessitá di aspettare 20 cicli di clock fra la fine del segnale alto di START e l'inizio del segnale alto di DONE, poiché il circuito impiega solamente un ciclo e mezzo di clock a fornire l'output. Con ulteriori test é anche verificabile il corretto funzionamento del circuito in caso di frequenza di clock superiori anche di un ordine di grandezza. Grazie ai risultati di sintesi ottenuti in fase di sviluppo é stato appurata una possibile ottimizzazione del circuito che permetterebbe l'eliminazione di uno stato dalla FSM in maniera tale da ottenere un output corretto dopo solo mezzo ciclo di clock dalla fine del segnale alto di START. Questa ottimizzazione renderebbe però instabile la critica fase di memorizzazione del valore in output della memoria e per questo é stata scelta un'implentazione leggermente meno performante, ma stabile.