



POLITECNICO

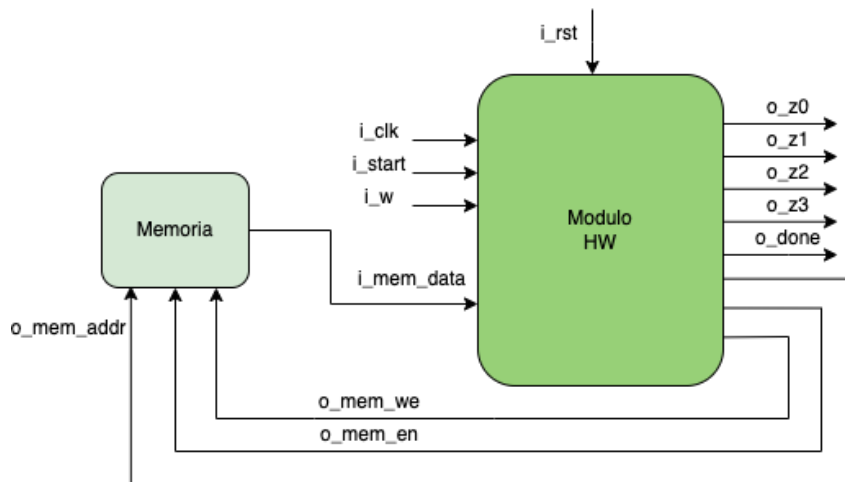
MILANO 1863

Progetto di Reti Logiche
Anno Accademico 2022/2023

Federica Maria Laudizi
Codice persona: 10724111

Introduzione

Il modulo HW è stato progettato per indirizzare il contenuto di un dato indirizzo di memoria, passato in input tramite un ingresso seriale i_w , su uno dei quattro canali d'uscita, anche questo specificato da i_w .



In particolare, all'istante iniziale la macchina verrà resettata: le uscite Z0, Z1, Z2 e Z3 e il segnale o_done vengono impostati a 0. L'inizio della trasmissione è indicato dal segnale i_start che rimarrà attivo fino alla fine di questa. La sequenza di dati è letta sul fronte di salita del clock ed è formata da:

- ⇒ 2 bit di intestazione che indicano il canale di uscita.
- ⇒ N bit di indirizzo di memoria dal quale prendere il messaggio da indirizzare. Questi bit di indirizzo possono variare da 0 fino ad un massimo di 16 bit; se il numero di bit di N è inferiore a 16, l'indirizzo viene esteso con 0 sui bit più significativi. Ad esempio:


$$(N = 5) \ 10110 \rightarrow \ 0000 \ 0000 \ 0001 \ 0110$$

Nel momento in cui i_start torna a 0 il modulo cesserà di leggere da i_w e si metterà in comunicazione con la memoria. Quando il messaggio verrà correttamente ricevuto, il segnale o_done passa da 0 a 1 per un solo ciclo di clock e il valore del canale corrispondente cambia, mentre gli altri canali mostreranno l'ultimo valore trasmesso derivato dai precedenti messaggi associati ad essi. Se il segnale o_done è a 0, tutti i canali di uscita devono essere a 0.

Il modulo deve essere progettato in modo da gestire il segnale di RESET in modo adeguato. Quando il segnale di i_rst viene attivato ($i_rst = 1$), il modulo esegue tutte le operazioni necessarie per riportare lo stato del modulo alle condizioni iniziali. Questo assicura che il modulo sia pronto per elaborare nuovi dati e garantire un funzionamento corretto.

Tuttavia, la macchina non richiede un ulteriore RESET ogni volta che viene ricevuto un segnale di START successivo. In altre parole, una volta che il modulo è stato inizializzato correttamente, non è necessario eseguire un nuovo RESET prima di elaborare i dati successivi. Per raggiungere questo obiettivo, il modulo è dotato di un meccanismo che permette al modulo di memorizzare il proprio stato corrente (i valori dei quattro canali di uscita) e di utilizzarlo in seguito all'elaborazione di dati successivi. In seguito, sono riportati due **esempi** esplicativi di entrambe le situazioni:


Dopo il primo *i_start* il sistema viene resettato nuovamente



<i>i_start</i>	0	0	0	0	1	1	1	1	1	0	0	0	0	0	1	1	1	0	0	0	0
<i>i_rst</i>	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
<i>o_done</i>	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
<i>i_w</i>	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	1	1	0	0	0	0
<i>o_z0</i>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
<i>o_z1</i>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	b1	0
<i>o_z2</i>	0	0	0	0	0	0	0	0	0	0	0	0	a3	0	0	0	0	0	0	0	0
<i>o_z3</i>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Canale: z2 Indirizzo: 0000 0000 0000 0110

Dopo il primo *i_start* il sistema NON viene resettato



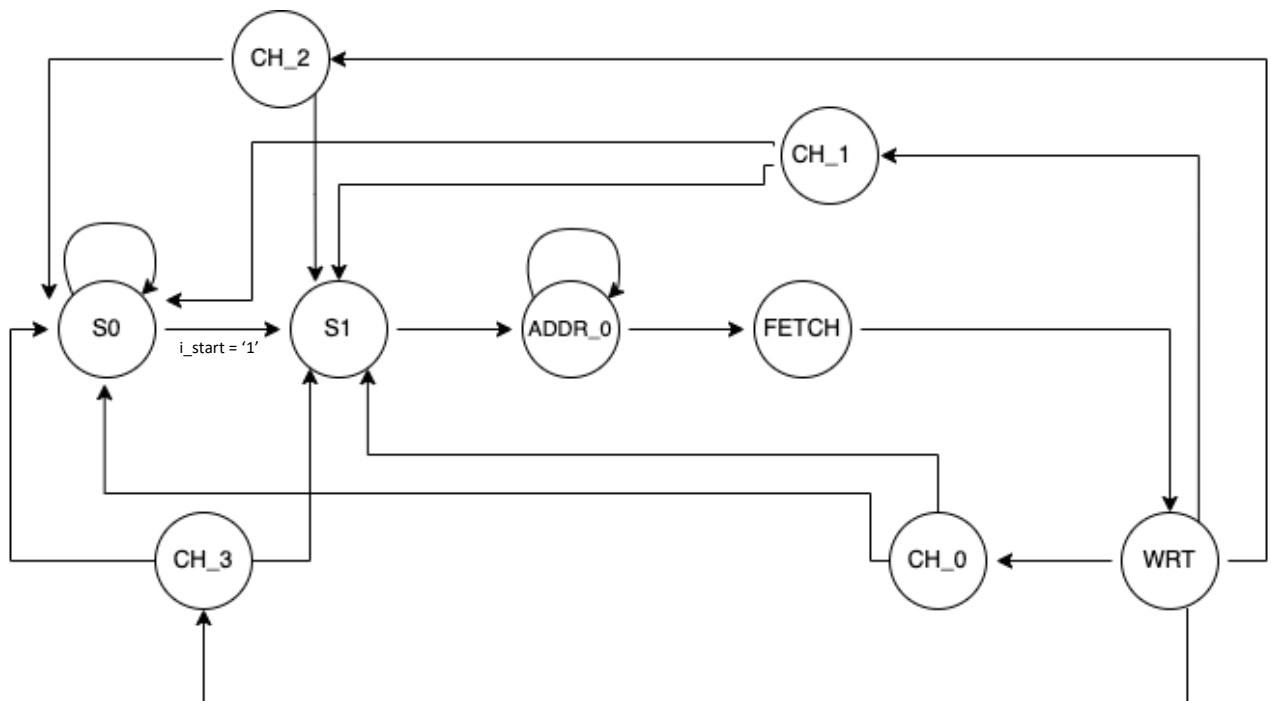
<i>i_start</i>	0	0	0	0	1	1	1	1	1	0	0	0	0	0	1	1	1	0	0	0	0
<i>i_rst</i>	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
<i>o_done</i>	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
<i>i_w</i>	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	1	1	0	0	0	0
<i>o_z0</i>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
<i>o_z1</i>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	b1	0
<i>o_z2</i>	0	0	0	0	0	0	0	0	0	0	0	0	a3	0	0	0	0	0	0	a3	0
<i>o_z3</i>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Architettura

Il modulo è stato implementato come una macchina a stati finiti in cui avviene una transizione di stato ad ogni ciclo di clock, quindi per eseguire un indirizzamento, nel caso peggiore (indirizzo in ingresso da 16 bit), si impiegano 22 cicli di clock. Inoltre, la scelta di mettere il reset in un'istruzione *if* piuttosto che creare un altro stato è stata fatta per vari motivi:

- Prestazioni: in alcuni casi, l'utilizzo di un reset branch in un'istruzione **if** può essere più veloce dell'utilizzo di uno stato di reset, poiché la macchina a stati non deve transitare attraverso uno stato di reset aggiuntivo.
- Flessibilità: se è necessario modificare la funzionalità di reset in seguito, può essere più facile apportare modifiche a un'istruzione **if** anziché modificare uno stato di reset separato.

Prima di presentare dettagliatamente la macchina progettata, riporto qui un suo grafo, al fine di illustrare in maniera chiara i diversi stadi di funzionamento (non sono stati riportati i segnali di transizione per chiarezza illustrativa).



Di seguito una tabella descrittiva degli stati e la loro funzione, riportando anche i valori assunti dai segnali principali.

Stato	Descrizione
S0	Attende l'inizio di una trasmissione e non appena questa comincia, prima di transitare al prossimo stato, legge il primo bit da i_w (quello corrispondente al bit più significativo del canale d'uscita). Inoltre, inizializza tutte le uscite a zero nel caso in cui dopo il primo $i_{start}=1$ non si dovesse passare dal reset.
S1	Legge il secondo bit di ingresso ovvero il bit meno significativo del canale d'uscita.

ADDR_0	Memorizza in un segnale interno l'indirizzo dal quale recuperare il dato dalla memoria, iterando su se stesso finché i_start non torna a zero (max 17 volte), momento in cui viene impostato o_mem_en = 1 (per comunicare alla memoria) e l'indirizzo viene passato in output alla memoria.
FETCH	L'unica utilità di questo stato è quella di attendere un ciclo di clock per ottenere il dato dalla memoria.
WRT	Memorizza il dato dalla memoria in un segnale interno per evitare di collegare direttamente un input con un output che potrebbero generare comportamenti indesiderati.
CH_0	Imposta o_done = '1' e scrive sui quattro canali di uscita i valori corrispondenti.
CH_1	Memorizza il dato proveniente dalla memoria in un segnale interno per utilizzarlo in seguito all'elaborazione di dati successivi. (Per il canale z0)
CH_2	Memorizza il dato proveniente dalla memoria in un segnale interno per utilizzarlo in seguito all'elaborazione di dati successivi. (Per il canale z1)
CH_3	Memorizza il dato proveniente dalla memoria in un segnale interno per utilizzarlo in seguito all'elaborazione di dati successivi. (Per il canale z2)

Nonostante la disponibilità di 20 cicli di clock per produrre il risultato (ovvero il tempo trascorso tra i_start=0 e o_done=1), in questa implementazione del modulo HW ne saranno necessari solo 3.

Risultati sperimentali

Elementi sintetizzati

Di seguito si riporta una tabella dei componenti RTL sintetizzati.

	Numero input	Bit	Numero elementi
Registri		16	2
		8	8
		2	1
		1	3
Mux	9	16	2
	2	8	3
	4	8	2
	9	8	8

	9	4	1
	2	4	6
	4	4	1
	3	2	1
	2	2	2
	9	2	1
	2	1	1
	9	1	11

Area Occupata

Dalla “report utilization” possiamo estrarre le informazioni riguardo l’area occupata dal design sintetizzato.

Risorsa	Utilizzo	Disponibilità	Utilizzo in %
Look Up Table	57	134600	0.04
Flip Flop	104	269200	0.04
Latch	0	269200	0.00

Sono stati consapevolmente evitati i latch, infatti, questi sono fondamentalmente un elemento di archiviazione asincrono che non ha un ingresso di clock con cui possa essere sincronizzato. Gli elementi asincroni necessitano di essere trattati con molta attenzione, altrimenti si rischiano timing errors e il loro output potrebbe cambiare in risposta a qualcosa diversa da un clock, per questo motivo è stato preferito non utilizzarli.

Note aggiuntive

Per questo progetto è stato utilizzato “Vivado 2022.2 WebPACK Edition”, e la FPGA xc7a200tfg484-1.

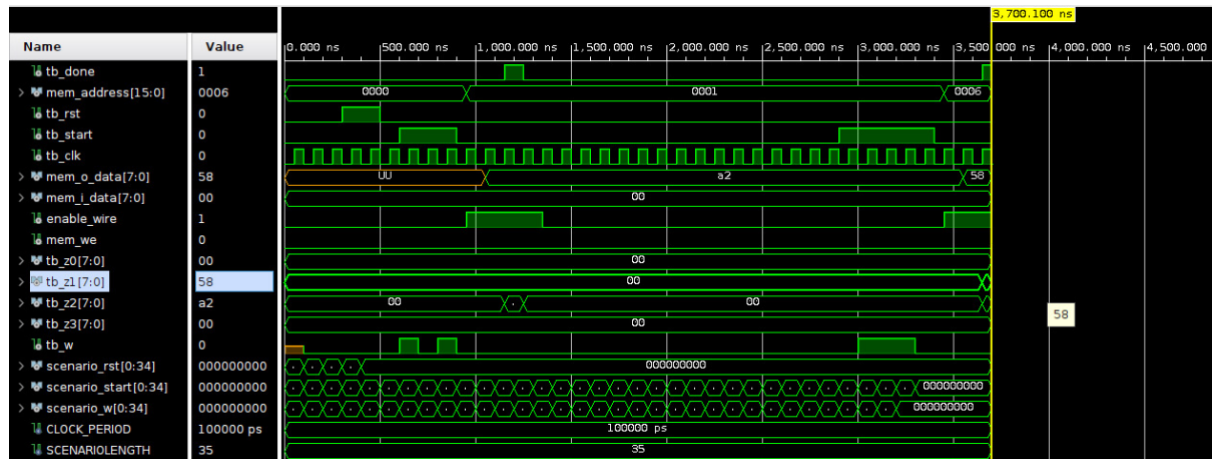
Simulazioni

Ho sottoposto il componente sia ai testbench ufficiali sia ad ulteriori testbench da me creati al fine di verificare il corretto funzionamento del componente in condizioni critiche. Inoltre sono stati effettuati dei test con input e indirizzi di memoria casuali (ordine dei milioni di test generati automaticamente). Tutte le simulazioni sono state verificate sia in pre- che in post-sintesi; per concisione verranno riportate le immagini solo di quelle in post-sintesi.

Di seguito i casi critici testati:

1. MULTI-START

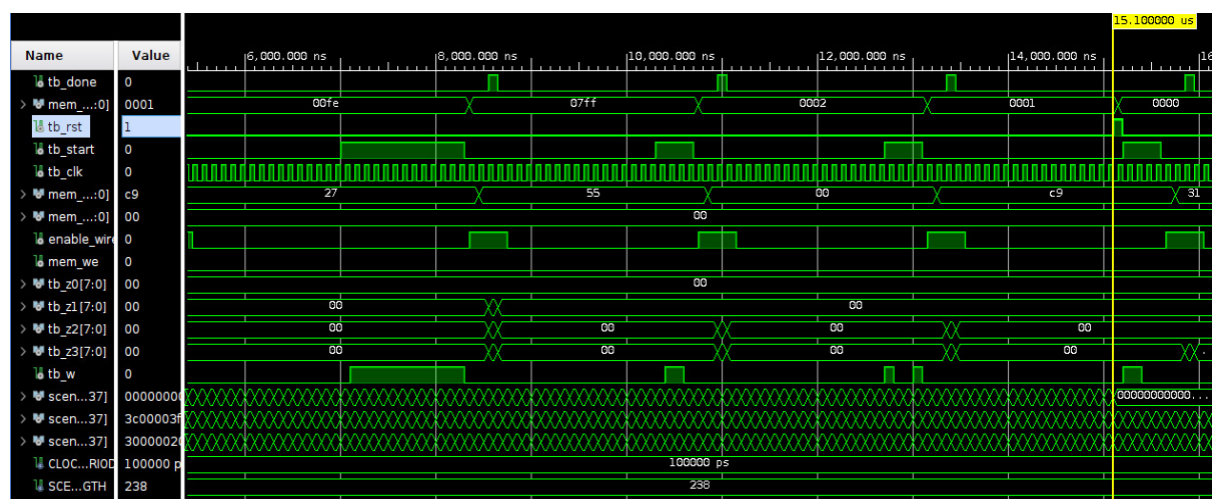
Dopo una prima lettura da memoria viene dato un nuovo segnale di start con un nuovo indirizzo da cui recuperare il dato senza passare dal reset.



Si nota come all'istante considerato (3,700,100 ns), cioè dopo la scrittura del dato preso dalla seconda trasmissione, non essendo passati dal reset, entrambi i canali Z1 e Z2 riportano in uscita dati presi della memoria.

2. SOVRASCRITTURA

La macchina scrive più di una volta su uno dei canali senza passare dal reset.

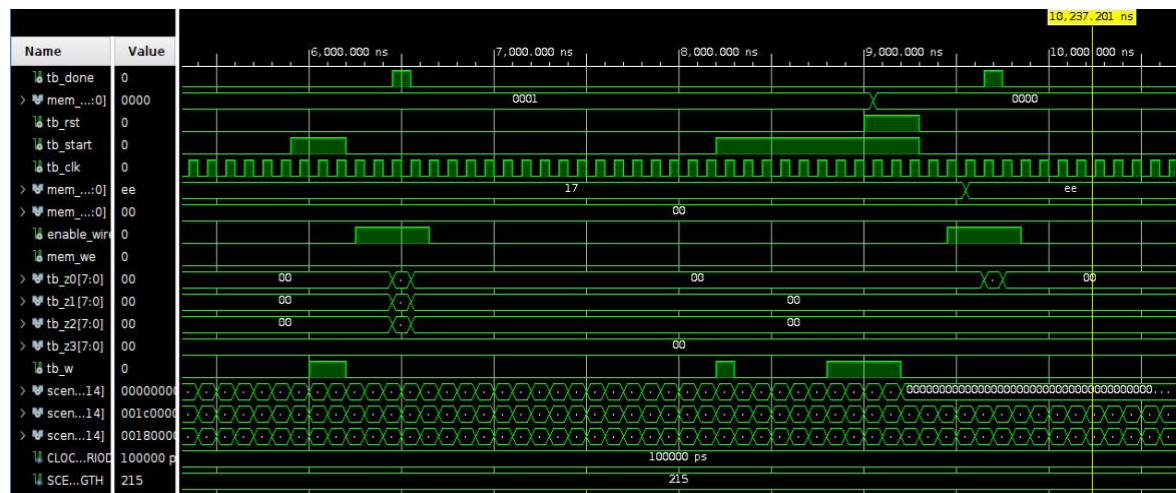


Questa testbench riporta chiaramente come si comporta in modulo nel caso di reset. Dopo un reset infatti tutti i canali di uscita vengono impostati a 0, e verrà scritto in uscita solo ciò che viene letto a partire dalle prossime trasmissioni.

3. RESET CRITICI

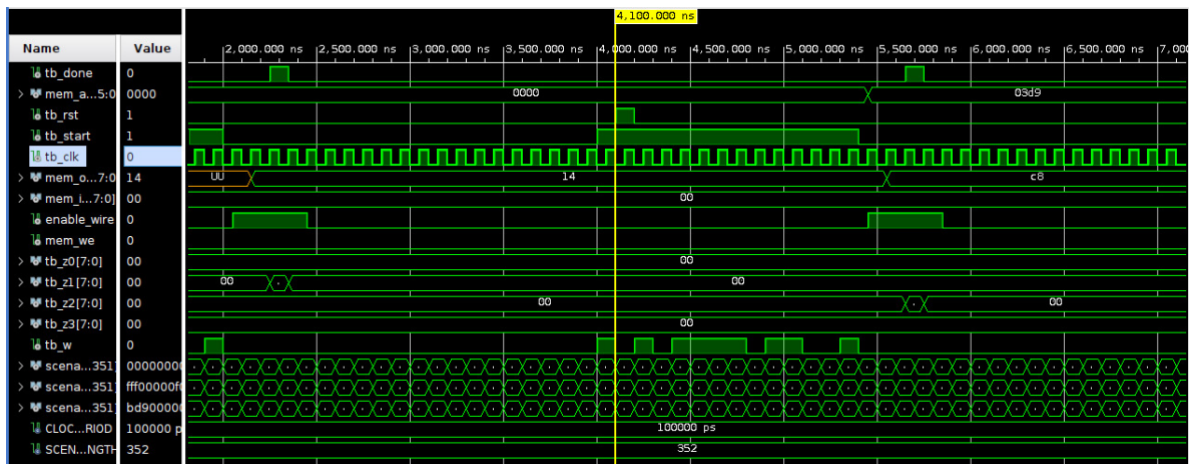
Dopo una prima lettura da memoria viene dato un nuovo segnale di reset mentre i_start è attivo.

i. Reset Durante La Lettura Dell'indirizzo Di Memoria



Lo scopo di questo testbench è stato testare se il componente HW funzionasse anche nel caso limite di ricevere un segnale di reset mentre è sta leggendo l'indirizzo da input.

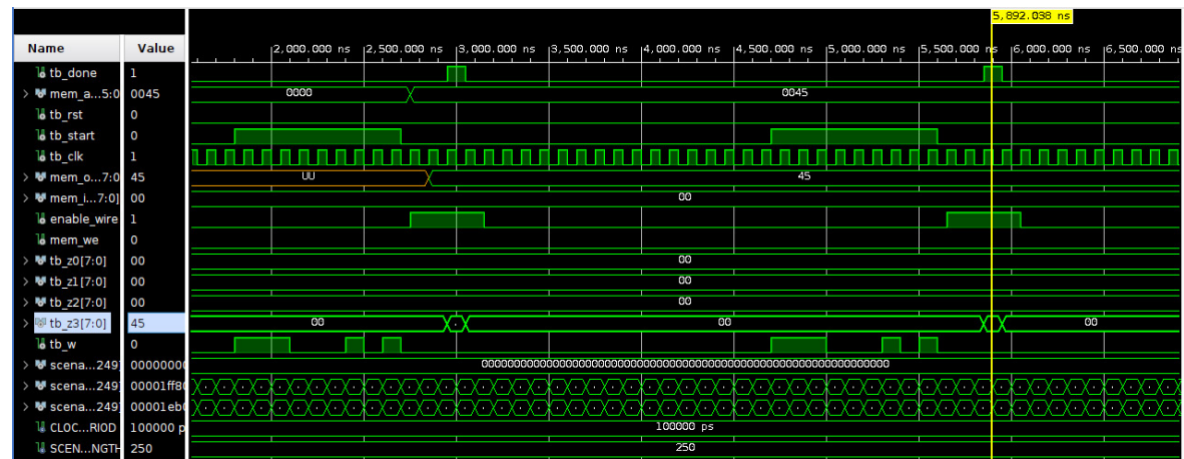
ii. Reset Durante La Lettura Del Canale Di Uscita



In questo caso ho voluto testare la macchina quando riceve il segnale di reset durante la lettura dei primi due bit d'ingresso. È evidente, infatti, come "ignori" i primi due cicli di clock in cui start = 1 per ricominciare ad elaborare i_w solo quando il reset torna a 0.

4. STESSO DATO

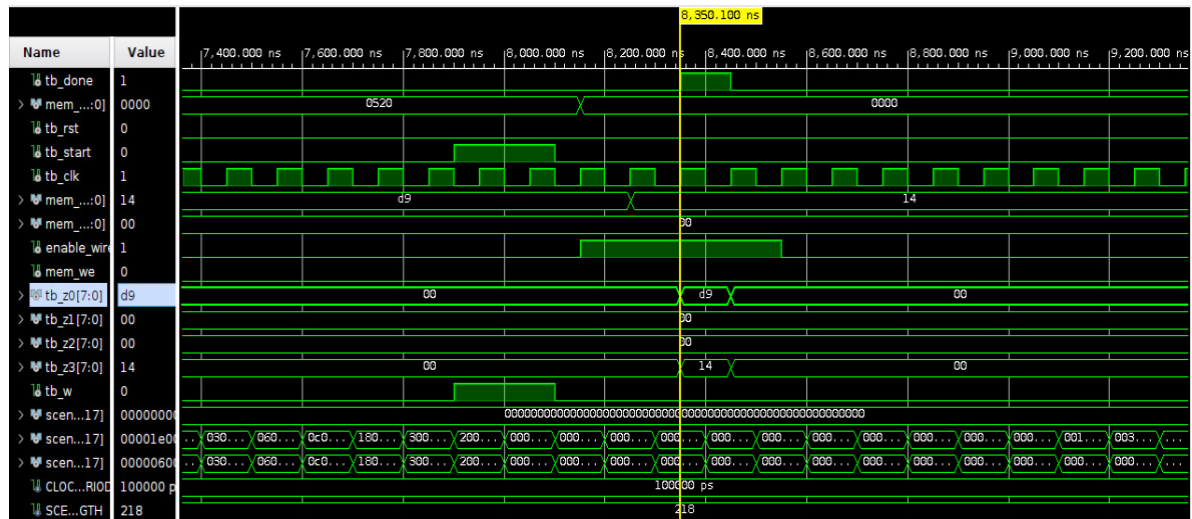
La macchina riceve da input lo stesso indirizzo due volte consecutivamente



In questo caso di test, invece, l'interesse è sulla sequenza di i_w che risulta la stessa per due volte consecutive. Il componente risponde in maniera corretta e ri-scrive in output il dato giusto.

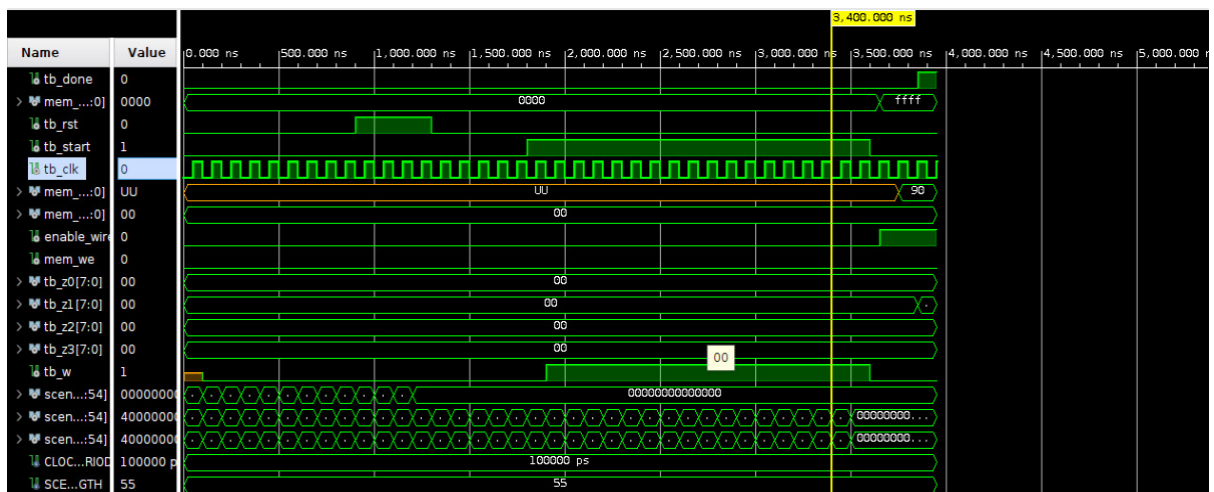
5. LUNGHEZZA O_MEM_ADDR

i. Minimo: Start = 1 Per Soli 2 Cicli Di Clock.



Come ultimi due casi limite sono stati riportati quelli relativi alla lunghezza della trasmissione. Questo in particolare testa il modulo quando fondamentalmente non viene passato nessun bit di indirizzo e di conseguenza scriverà sul canale d'uscita quello che c'è in 0000.

ii. Massimo: Start = 1 Per 18 Cicli Di Clock.



In questo testbench, viene verificato che il modulo riesca a leggere correttamente da w un in ingresso di 16 bit.

Conclusioni

In conclusione, il progetto ha raggiunto tutti gli obiettivi prefissati. È stato scelto di utilizzare una macchina a stati finiti, suddividendo il progetto in sottocomponenti indipendenti e riutilizzabili. Questo ha permesso di ottenere un codice efficiente e facilmente manutenibile.

I risultati dei test e delle simulazioni mostrano che il progetto funziona come previsto, soddisfacendo tutte le specifiche richieste. Durante lo sviluppo, sono stati riscontrati alcuni problemi, principalmente in sintesi, che sono stati affrontati eliminando tutti i latch esistenti nel circuito.

Infine, è stato ampiamente rispettato il vincolo sul periodo di clock di 100ns, infatti, il modulo ne impiega circa tre.