Progetto Reti Logiche

A.A. 2022/2023

Ye Xin 10727797 - 956040
Zacchetti Simone 10767415 - 959220

1. Introduzione

L'obiettivo del progetto era realizzare un componente hardware che si interfacciasse con una memoria, leggendo valori contenuti in essa, per poi indirizzarli su un canale d'uscita fra quattro disponibili.
Tale componente dispone di un ingresso seriale da un bit con cui sceglie il canale d'uscita e l'indirizzo di memoria da cui prelevare il dato.

1.1 Specifiche in dettaglio

Il componente ha due ingressi da 1 bit (i_w, i_start), il canale di uscita e l'indirizzo di memoria vengono costruiti sulla base del contenuto del segnale i_w, il quale viene considerato valido soltanto quando i_start è alto (i_start = 1).
Dal momento in cui i_start è alto, i primi due bit letti da i_w andranno a comporre l'indirizzo del canale di uscita (o_z0 = 00, o_z1 = 01, o_z2 = 10, o_z3 = 11); i restanti bit letti da i_w finché i_start è alto andranno a comporre l'indirizzo di memoria da 16 bit. Se il numero di bit è inferiore a 16, l'indirizzo viene esteso con degli 0 sui bit più significativi.
Esempio:

(N = 7))	1010111	–>	0000000001010111
(N = 16)	1010111	–>	0000000001010111
(N = 0)	0000000000000000	–>	0000000000000000

Tutti i bit di i_w sono letti sul fronte di salita del clock.
In uscita è presente un segnale i_done; quando tale segnale è basso (o_done = 0) tutti gli 8 bit di tutte le uscite sono a 0. Quando o_done è alto il canale associato al messaggio viene aggiornato con il nuovo valore mentre sulle altre uscite viene mostrato il valore che avevano in precedenza.
È presente anche un segnare di reset (i_rst) che verrà sempre dato prima del primo segnale di start e ogni volta che si vuole re-inizializzare il modulo.

2. Architettura

L'architettura del componente segue il seguente algoritmo:

1. Il bit letto da i_w viene salvato nel registro reg_0.
2. Il dato in uscita da tale registro viene inviato ad un multiplexer che, per i primi due cicli di clock in cui i_start = '1', lo manda ad un concatenatore per costruire l'indirizzo di memoria, il cui risultato parziale (cioè quello dopo la lettura del solo primo bit) verrà salvato nel registro reg_1.
3. L'indirizzo completo di memoria viene salvato nel registro reg_3 per poi essere mandato al decoder.
4. Il dato letto da i_w viene mandato verso un secondo contatenatore, il quale, seguendo la stessa logica del primo, costruisce l'indirizzo di memoria.
5. Una volta che la memoria fornisce il dato richiesto, questo viene mandato a tutti i registri d'uscita ma verrà caricato soltanto su quello corrispondente all'uscita voluta.
6. A valle dei registri d'uscita ci sono dei multiplexer che inviano il dato ricevuto dai registri sulle uscite soltanto sei il segnale o_done è alto.

Il componente è stato sviluppato partendo da un datapath e da una macchina a stati finiti che permettesse di comandare i segnali del datapath.

2.1 Datapath

Per il datapath sono stati utilizzati i seguenti moduli (vedi Figura 1):

1. Concatenatore: esso prende in input un segnale da 2 bit (concat_1) o 16 bit (concat_2) e un segnale da 1 bit, in output ha un segnale da 2 bit (concat_1) o 16 bit (concat_2). Tale modulo concatena il singolo bit di input al vettore di 2 o 16 bit nella posizione meno significativa e rimuove il bit più significativo dal vettore dato in input.
2. Decoder: prende in input un vettore da 2 bit che contiene l'indirizzo della porta di uscita, ha quattro segnali di output da 1 bit, i quali andranno a comandare il segnale di load nei quattro registri di uscita (reg_z0, reg_z1, reg_z2, reg_z3).
   Ad esempio, se il decoder riceve in input il segnale "00", l'uscita collegata a reg_z0 andrà a '1' mentre le altre saranno '0'.
3. Registri: permettono di memorizzare un valore di un segnale dato in input e restituirlo in output.

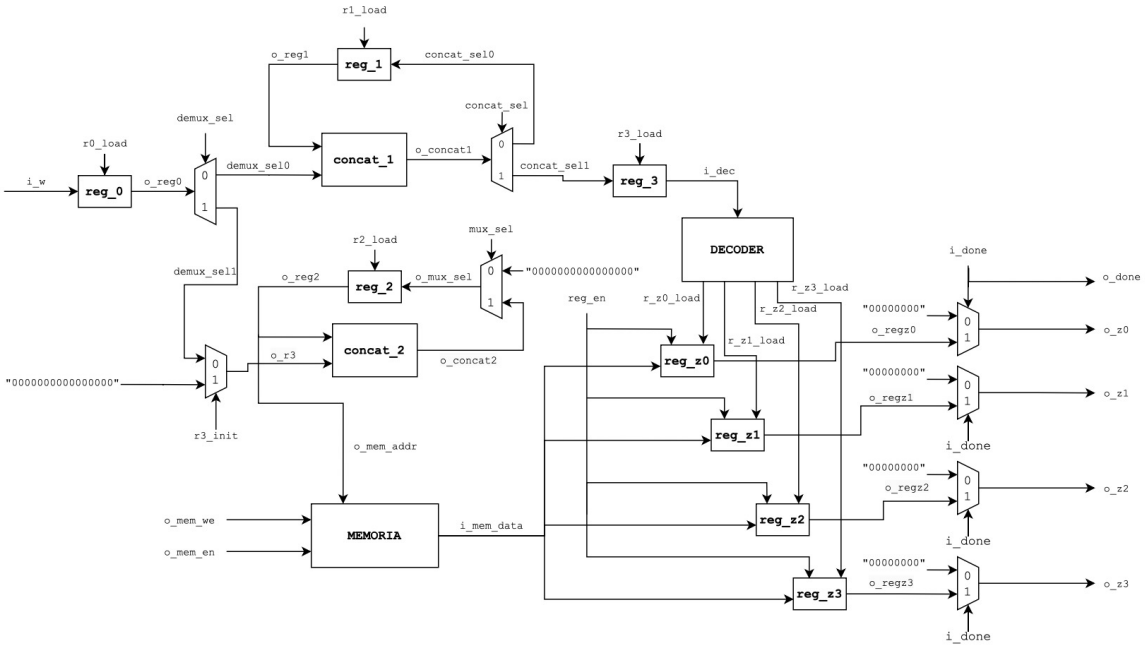


Figura 1: Datapath

2.2 Macchina a Stati Finiti

Al datapath è stata affiancata una FSM a 8 stati (vedi Figura 2) con cui controllare i segnali interni al modulo.

Gli 8 stati sono:

- S0: è lo stato iniziale, quando arriva il segnale i_rst la FSM si trova in questo stato.
- S1: è lo stato in cui si trova la macchina dopo il primo ciclo di clock in cui il segnale di start è alto; il primo bit, dopo essere stato concatenato (si noti che inizialmente il valore contenuto nel registro reg_2 non è significativo), viene salvato nel registro reg_2.
- S2: il secondo bit letto da i_w viene concatenato al primo, l'indirizzo della porta di uscita è completo. Tramite il demultiplexer comandato dal segnale concat_sel il dato viene inviato e salvato nel registro reg_3. A questo punto se i_start va a 0, e quindi l'indirizzo di memoria sarà composto da soli 0, la macchina va in S5, altrimenti in S3.

  Viene anche posto a '1' il segnale o_mem_en che permette di leggere il dato da memoria nel caso limite di lettura da i_w del solo indirizzo d'uscita; se il caso corrente non corrisponde con il caso limite, il dato letto dalla memoria verrà poi sovrascritto.
- S3: il dato letto da i_w viene mandato verso il secondo concatenatore, per poi essere salvato in reg_2. Il processo continua finché i_start è alto; quando si abbassa la FSM va nello stato S4.
- S5: il segnale reg_en è alto per tutti i registri; il dato proveniente dalla memoria viene caricato soltanto nel registro per cui anche il segnale di load proveniente dal decoder è

alto (ad esempio, il segnale verrà caricato in reg_z0 solo se reg_en = ‘1’ and reg_z0_load = ‘0’).

- S6: in questo stato i_done, e quindi anche o_done, è alto e permette di visualizzare i valori sulle uscite.
- S7: è uno stato di “attesa” che permette ai segnali di sincronizzarsi correttamente.

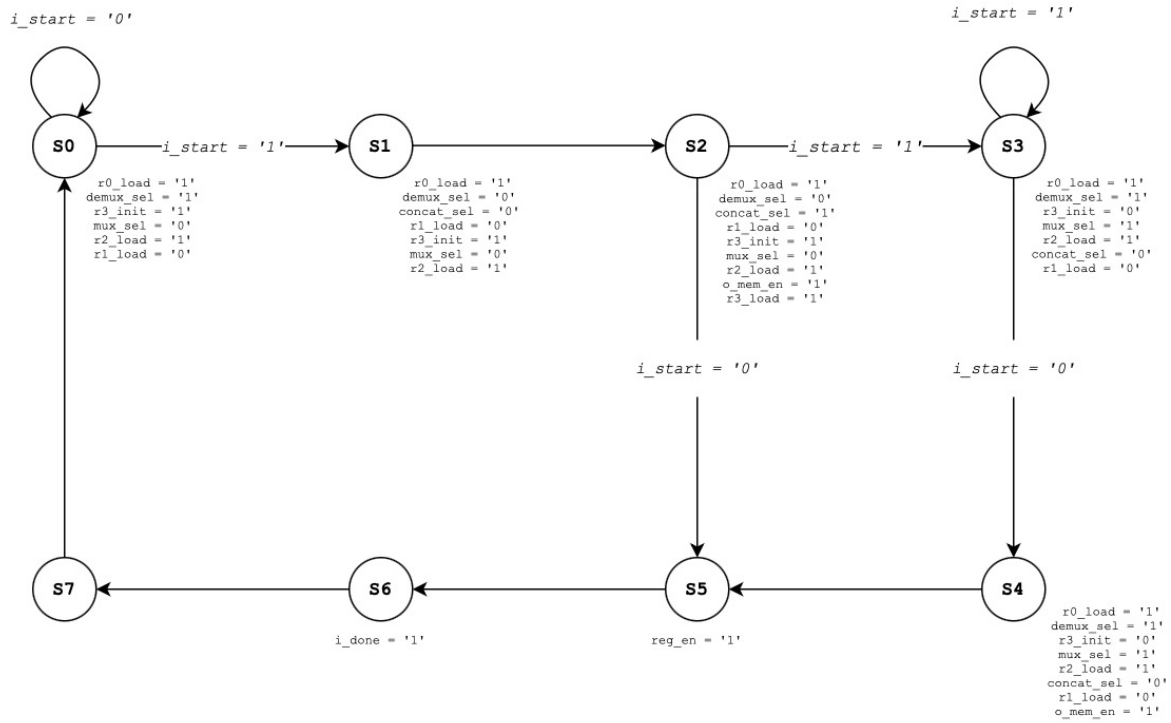


Figura 2: Macchina a Stati Finiti

2.3 Schema dell'implementazione

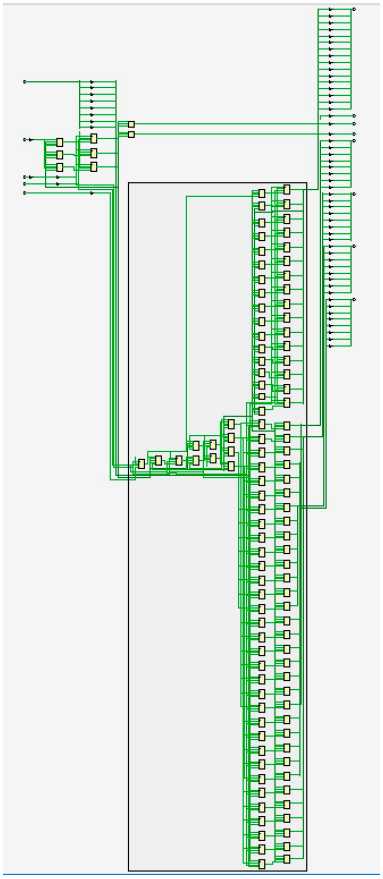


Figura 3: Schema dell'implementazione

3. Risultati sperimentali

3.1 Testing

Conclusa la descrizione del componente si è proceduto con l'eseguire una simulazione pre-sintesi del componente (Behavioral Simulation) utilizzando i test bench forniti.
L'obiettivo era testare il corretto funzionamento generale del modulo e il caso particolare di lettura da i_w del solo indirizzo di uscita.
Una volta passati tutti i test, si è proceduto con la sintesi del componente per poi testarne il corretto funzionamento anche in simulazione post-sintesi.

È qui che si è reso necessario introdurre un segnale di enable sui registri di uscita poiché inizialmente la sintesi ha introdotto un ritardo dei segnali rispetto al clock, causando il caricamento sui registri non previsto.
Una volta inserito tale segnale, gestito nella FSM e uno stato finale di attesa, anche i test post-simulazione restituivano il risultato voluto.

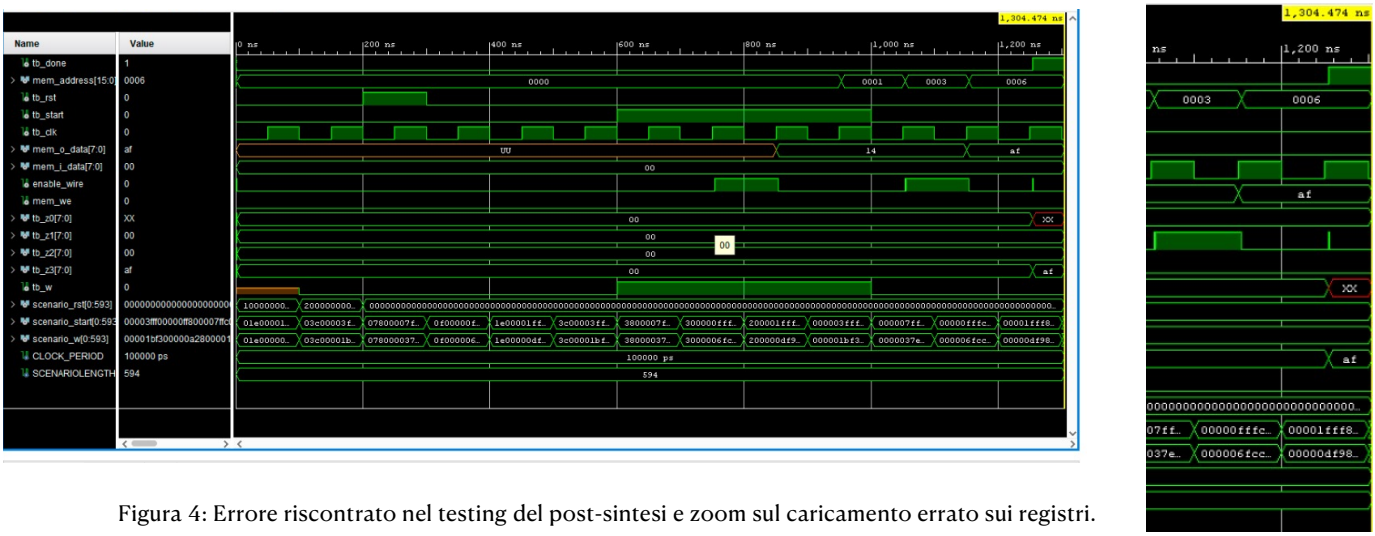
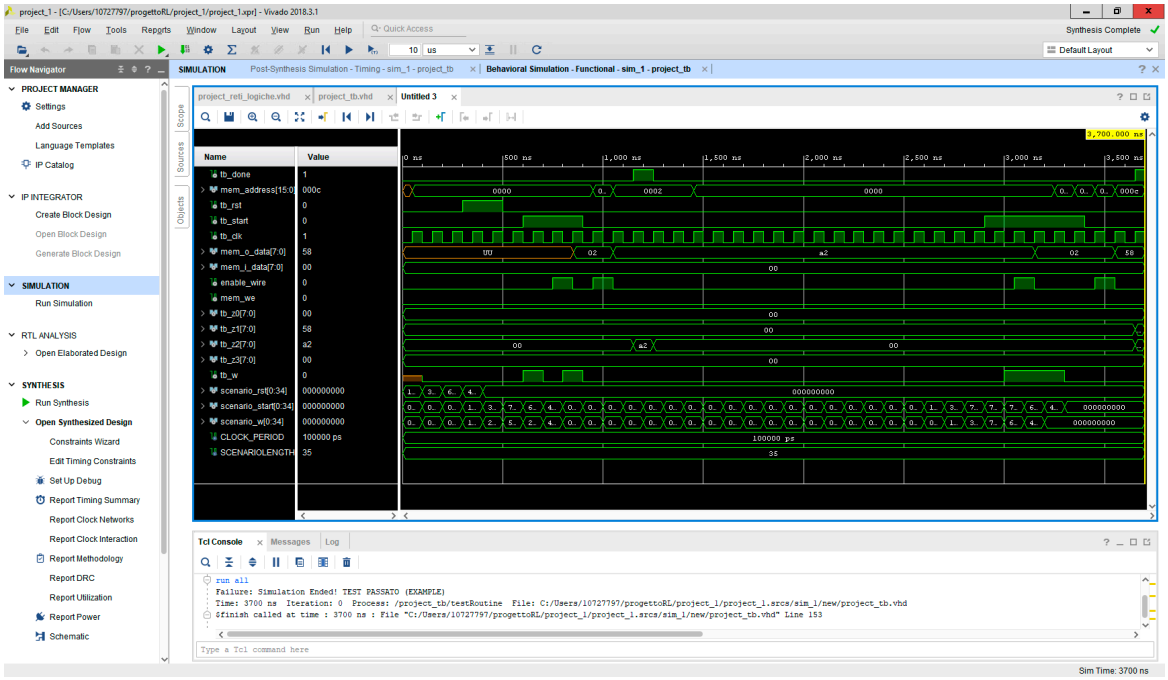


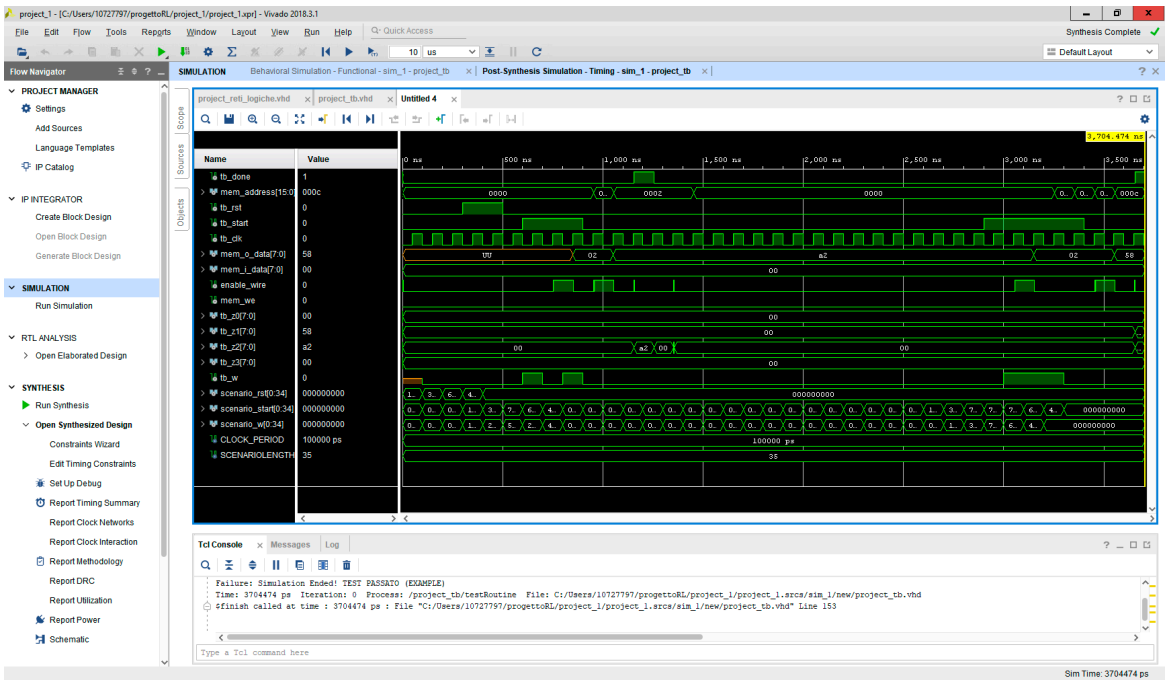
Figura 4: Errore riscontrato nel testing del post-sintesi e zoom sul caricamento errato sui registri.

3.2 Test bench

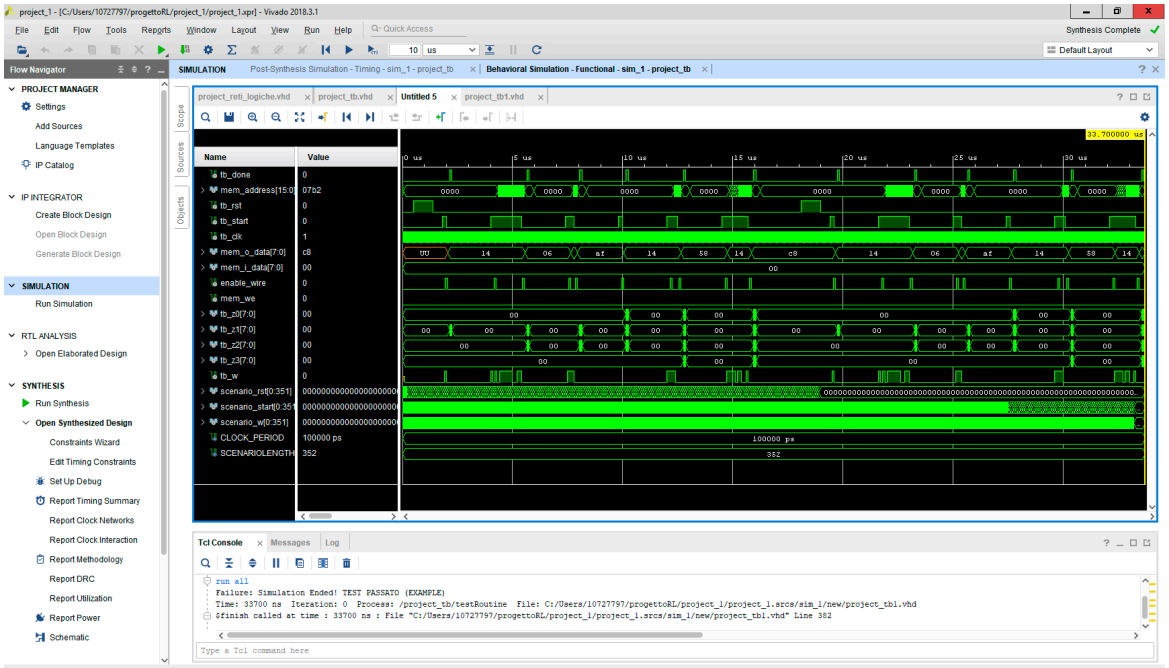
Di seguito alcuni risultati delle simulazioni dei test bench utilizzati durante il testing, forniti dal professor Gianluca Palermo.

Simulazione pre e post-sintesi del test bench 2022-2023.





Simulazione pre e post-sintesi del test bench 1.



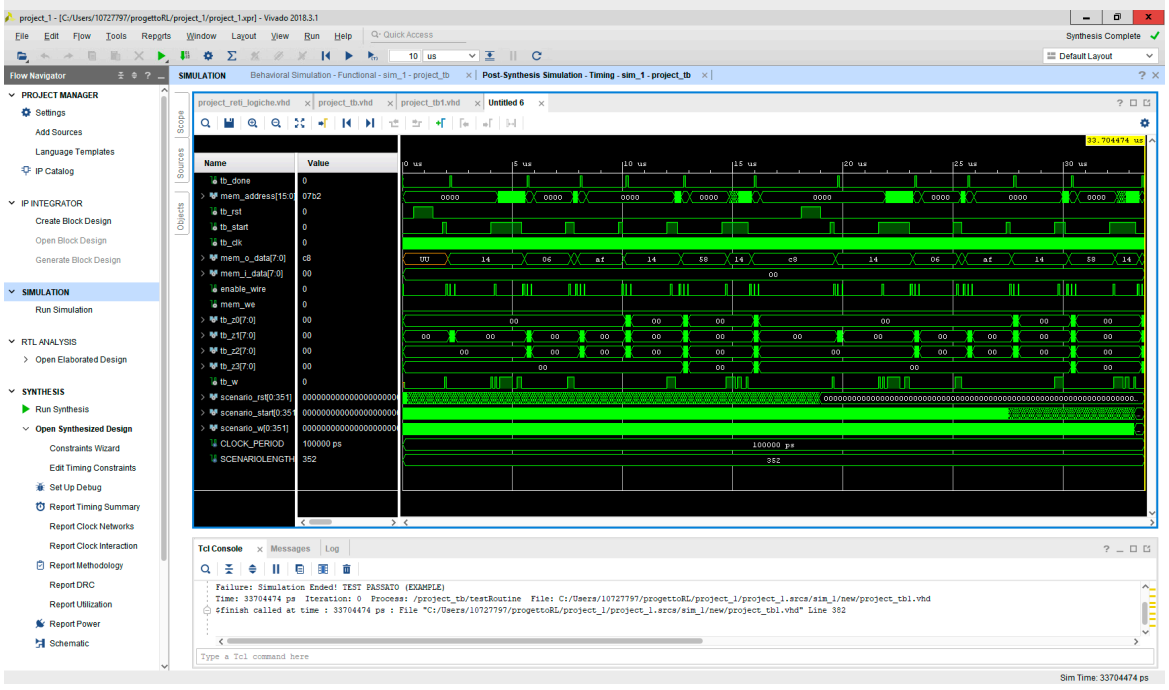
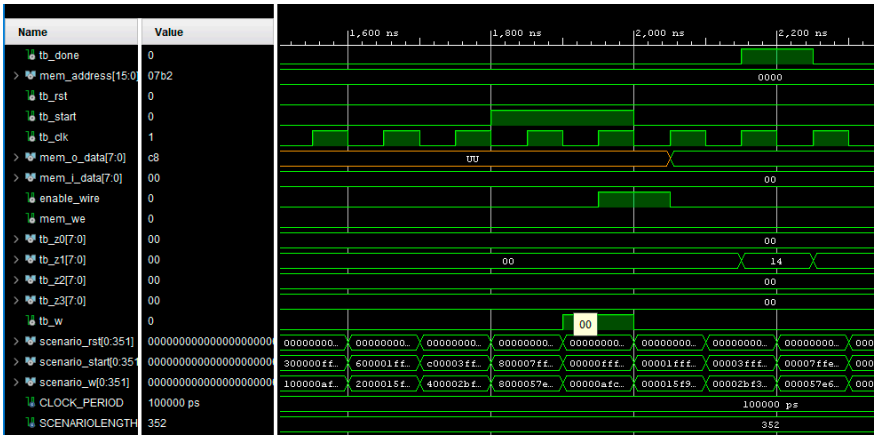


Figura 5: zoom sul caso particolare con solo 2 bit in input (test bench 1).



Simulazione pre e post-sintesi del test bench 2.

3.3 Report di sintesi

Per quanto riguarda l'area utilizzata, la sintesi riporta il seguente utilizzo di componenti:

- LUT: 35
- FF: 55

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	35	0	134600	0.03
LUT as Logic	35	0	134600	0.03
LUT as Memory	0	0	46200	0.00
Slice Registers	55	0	269200	0.02
Register as Flip Flop	55	0	269200	0.02
Register as Latch	0	0	269200	0.00
F7 Muxes	0	0	67300	0.00
F8 Muxes	0	0	33650	0.00

Figura 4: Slice Logic

Si sottolinea l'assenza di latch.

Dal timing report analizzato su tutti i testbench si può vedere uno slack time (tempo di margine, cioè le differenza tra il tempo disponibile per completare un percorso di segnale e il tempo effettivo impiegato per attraversare tale percorso) tendente a infinito.

```
Timing Report

Slack:                    inf
```

Figura 5: Slack Time

4. Note conclusive

L'intero componente è stato sviluppato mediante due architetture. La prima sviluppa e gestisce la macchina a stati e i segnali che comandano i componenti del datapath. La seconda descrive il datapath, e quindi l'aggiornamento dei segnali ad ogni ciclo di clock.
Il componente è stato realizzato rispettando le richieste della specifica e cercando di fornire quanto prima il risultato ottenuto.