Prova finale di Reti Logiche

Prof. Fabio Salice – AA 2019/20

Davide Mantegazza

Sommario

Introduzione

Specifica del progetto

Lo scopo di questo progetto è l'implementazione in linguaggio VHDL di un componente hardware che consente di codificare indirizzi di memoria per mezzo del metodo di codifica a bassa dissipazione di energia Working-Zone, il quale consiste nel convertire una stringa di bit in input da trasmettere nel bus indirizzi nel caso in cui essa appartenga ad un certo intervallo di indirizzi. Questo intervallo, detto per l'appunto working-zone (WZ), è identificato da un indirizzo base ed è dotato di una dimensione fissa, che fornisce il numero di indirizzi tra di loro consecutivi che compongono la working-zone. Inoltre, nello schema di codifica è possibile definire un numero arbitrario di working-zone.

Interfaccia del componente

L'interfaccia del componente è così descritta:

```
entity project_reti_logiche is
    port (
        i_clk: in std_logic;
        i_start: in std_logic;
        i_rst: in std_logic;
        i_data: in std_logic_vector(7 downto 0);
        o_address: out std_logic_vector(15 downto 0);
        o_done: out std_logic;
        o_en: out std_logic;
        o_we: out std_logic;
        o_data: out std_logic_vector(7 downto 0)
    );
end entity project_reti_logiche;
```

Dove:

- `i_clk` rappresenta il segnale di clock generato dal test bench
- `i_start` è il segnale di inizio elaborazione generato dal test bench
- `i_rst` è il segnale generato dal test bench che riporta la macchina nello stato di reset
- `i_data` rappresenta il segnale in arrivo dalla memoria a seguito di una richiesta di lettura
- `o_address` rappresenta l'indirizzo di memoria su cui si vuole scrivere o che si vuole leggere
- `o_done` è il segnale di fine elaborazione generato dal componente
- `o_en` è il segnale generato dal componente usato per comunicare con la memoria (sia in lettura che in scrittura)
- `o_we` è il segnale generato dal componente che viene inviato alla memoria per poter scrivere su di essa
- `o_data` rappresenta il segnale di uscita dal componente

Dati e memoria

In questa implementazione si adoperano indirizzi da 7 bit, che quindi coprono il range di indirizzi da 0_{10} a 127_{10}, e si definiscono 8 working-zone da 4 indirizzi ciascuna, non sovrapposte tra di loro (ovvero un indirizzo può essere incluso in una sola working-zone). Queste informazioni sono salvate in una memoria composta da 2^{16} parole, ciascuna da 8 bit, con indirizzamento al byte nella seguente maniera:

- Nelle prime posizioni da 0 a 7 della memoria si salvano gli indirizzi base delle working-zone che si utilizzeranno nella codifica dell'indirizzo
- Nella posizione 8 si salva l'indirizzo da convertire
- Nella posizione 9 viene memorizzato il risultato della conversione dell'indirizzo

Occorre precisare che essendo gli indirizzi considerati lunghi 7 bit, essi verranno salvati in memoria ponendo il bit più significativo della parola a 0.

Figura 1 Rappresentazione della memoria

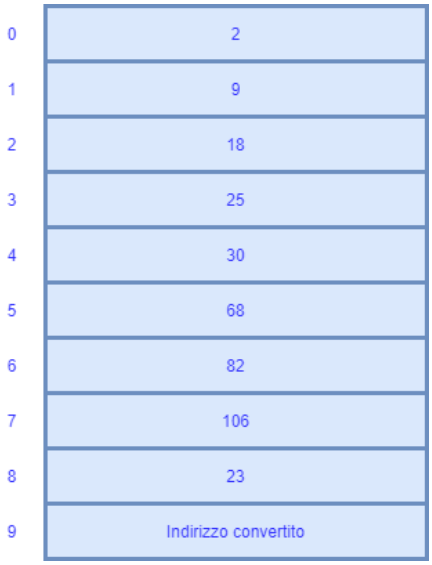
Codifica dell'indirizzo

L'indirizzo viene codificato nella seguente maniera:

- Se non appartiene a nessuna working-zone, il componente si limita a copiare il contenuto della posizione 8 della memoria nella posizione 9 (in altri termini viene generata una stringa di 8 bit composta da uno zero concatenato all'indirizzo di partenza)
- Se appartiene ad una delle working-zone presenti in memoria, il componente genera in uscita una stringa da 8 bit così composta:
    - Il bit più significativo (in posizione 7) viene posto uguale a 1
    - Nelle posizioni dalla 6 alla 4 si memorizza il numero della working-zone, codificato in binario
    - Nelle posizioni dalla 3 alla 0 si memorizza l'offset tra l'indirizzo da convertire e l'indirizzo base della working-zone, codificato in one hot

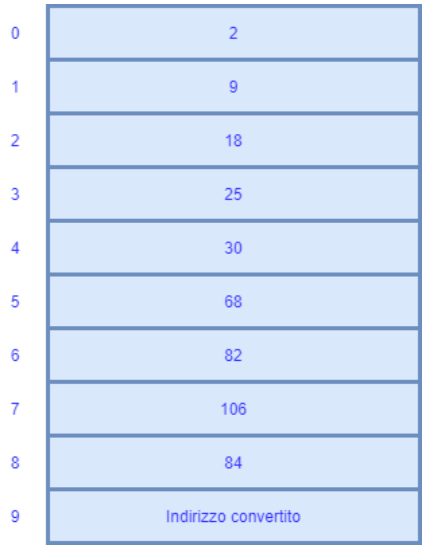
Esempio

Si consideri il seguente stato della memoria, prima dell'implementazione da parte del componente:

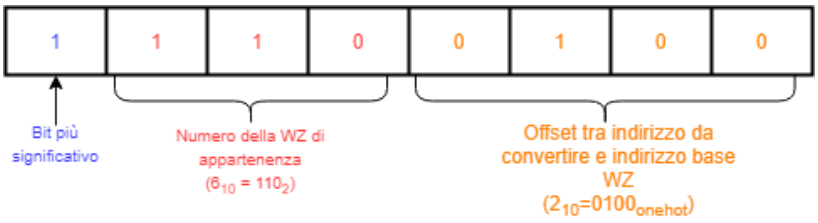


In questo caso l'indirizzo da convertire, ovvero 23_{10}, non appartiene a nessuna delle working-zone presenti in memoria (o in altri termini, la differenza tra esso e l'indirizzo base di ogni working-zone è maggiore o uguale a 4), perciò il componente si limiterà a salvare il valore 23_{10} nella cella 9 della memoria.

Consideriamo ora la seguente situazione



In questo caso si nota che l'indirizzo da convertire (84_{10}) appartiene alla working-zone 6, in quanto l'offset tra l'indirizzo da convertire e l'indirizzo base della working zone considerata è pari a 2. Di conseguenza l'indirizzo convertito è pari a 11100100_2 (equivalente a 228_{10}), e viene ottenuto in questo modo:



Design del componente

Stati del componente

Il componente è stato implementato progettando una macchina di Moore a stati finiti composta da 13 stati diversi. Di seguito viene fornita una descrizione sintetica di questi ultimi

- `IDLE`: in questo stato il componente si limita ad attendere la ricezione di un segnale `i_start` per dare inizio alla computazione. `IDLE` costituisce lo stato iniziale della FSM e può essere raggiunto da qualunque altro stato di quest'ultima ponendo a 1 il segnale `i_rst`
- `INPUT_ADDRESS_FETCHING`: questo stato è usato per fornire alla memoria l'indirizzo della cella che contiene la stringa di bit da convertire (in questo caso essa si trova sempre nella cella di memoria di indice 8)
- `INPUT_ADDRESS_WAIT_FOR_RAM`: questo stato viene adoperato per verificare che il componente abbia effettivamente acquisito la stringa da convertire dalla memoria
- `SAVE_INPUT_ADDRESS`: in questo stato il componente memorizza la stringa di bit da elaborare
- `WZ_FETCHING`: questo stato è usato per fornire alla memoria l'indirizzo della cella contenente l'indirizzo di base della WZ richiesta dal componente
- `WZ_WAIT_FOR_RAM`: questo stato viene adoperato per verificare che il componente abbia effettivamente acquisito dalla memoria l'indirizzo base della working-zone al momento considerata dal componente
- `SAVE_WZ`: in questo stato il componente memorizza l'indirizzo di base della working-zone che verrà adoperata dal componente
- `WZ_CHECK`: il compito di questo stato consiste nel verificare se la stringa data in input appartiene ad una delle working-zone presenti in memoria. Se ciò è verificato, il componente procede con la conversione della stringa, altrimenti si provvede a riscrivere la stringa data in memoria
- `WRITE_UNMODIFIED_OUTPUT_ADDRESS`: questo stato viene utilizzato per salvare nella posizione 9 della memoria una stringa di bit che si è verificato non appartenere ad alcuna working-zone

- CONVERT_OFFSET: questo stato viene usato per convertire la codifica dell'offset tra la stringa da convertire e la working-zone a cui appartiene da binario a one hot
- PREPARE_MODIFIED_OUTPUT_ADDRESS: questo stato si occupa di generare la stringa da salvare in memoria nel caso in cui l'indirizzo fornito in input al componente appartenga ad una working-zone
- WRITE_MODIFIED_OUTPUT_ADDRESS: in questo stato il componente provvede a salvare in memoria la stringa di bit generata nello stato PREPARE_MODIFIED_OUTPUT_ADDRESS
- DONE: questo stato viene raggiunto quando il segnale o_done viene posto a 1. Il componente si occuperà poi di attendere che i_start venga abbassato a 0, per ritornare nello stato IDLE

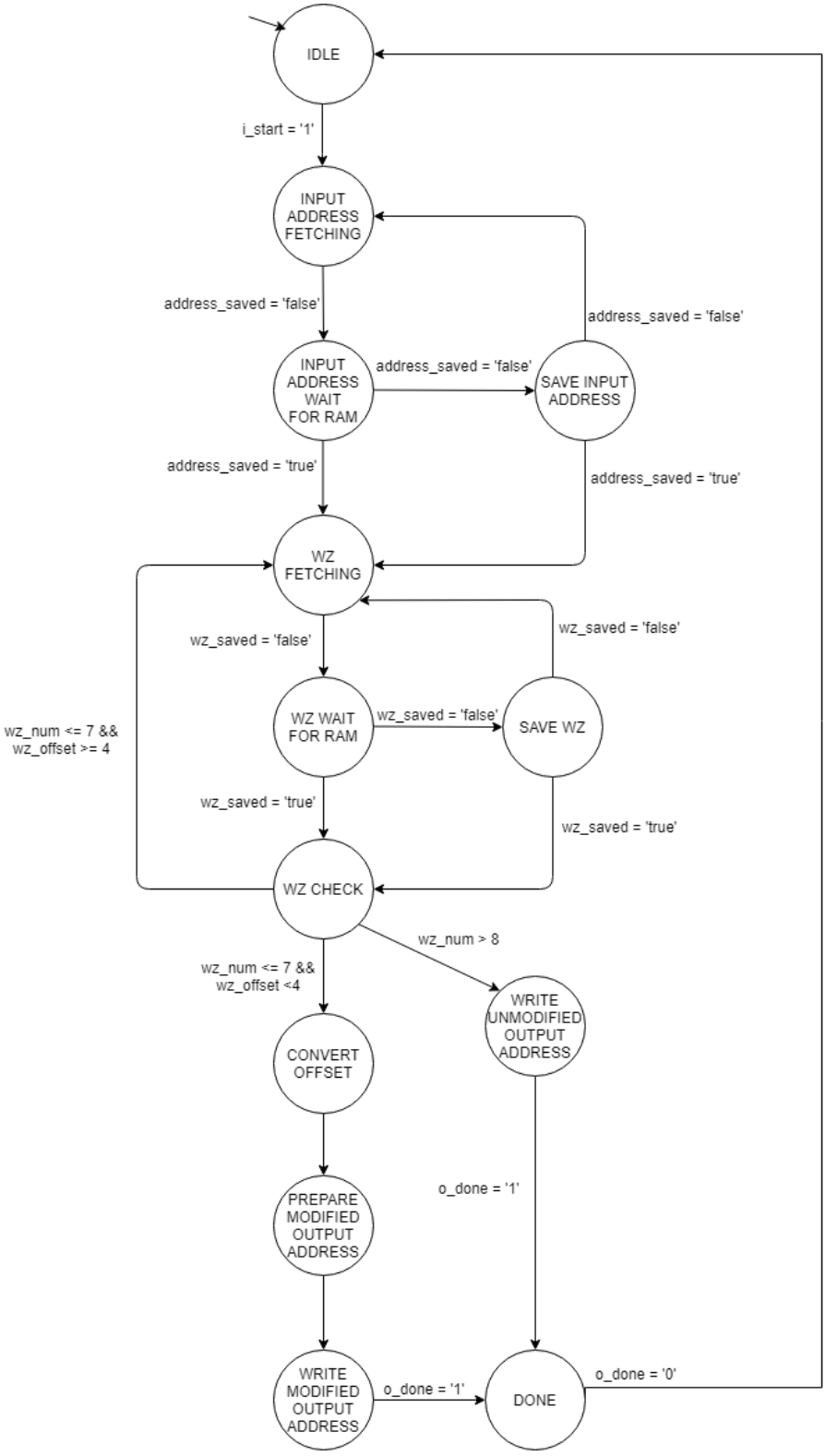


Figura 2 Rappresentazione del componente in forma di macchina a stati finiti

Esecuzione della computazione

Il componente da inizio alla computazione quando il test bench ad esso fornito pone il segnale `i_start` a 1; in questo istante il componente passerà dallo stato `IDLE` allo stato `INPUT_ADDRESS_FETCHING`. Una volta terminata l'elaborazione e dopo aver salvato in memoria il risultato di quest'ultima, indipendentemente dal fatto che la stringa fornita in input sia stata convertita o meno, il componente alza ad 1 il segnale `o_done`, e si posiziona nello stato `DONE`. A questo punto il test bench reagisce ponendo a 0 il segnale `i_start`, stimolando il componente a resettare tutte le variabili dichiarate, a porre `o_done` a 0 e a riposizionarsi nello stato `IDLE` per poi attendere nuovamente che `i_start` venga alzato a 1. Il componente può inoltre ricevere in input un segnale `i_rst`, che se portato ad 1 porta il componente nello stato `IDLE`, indipendentemente dallo stato precedentemente occupato.

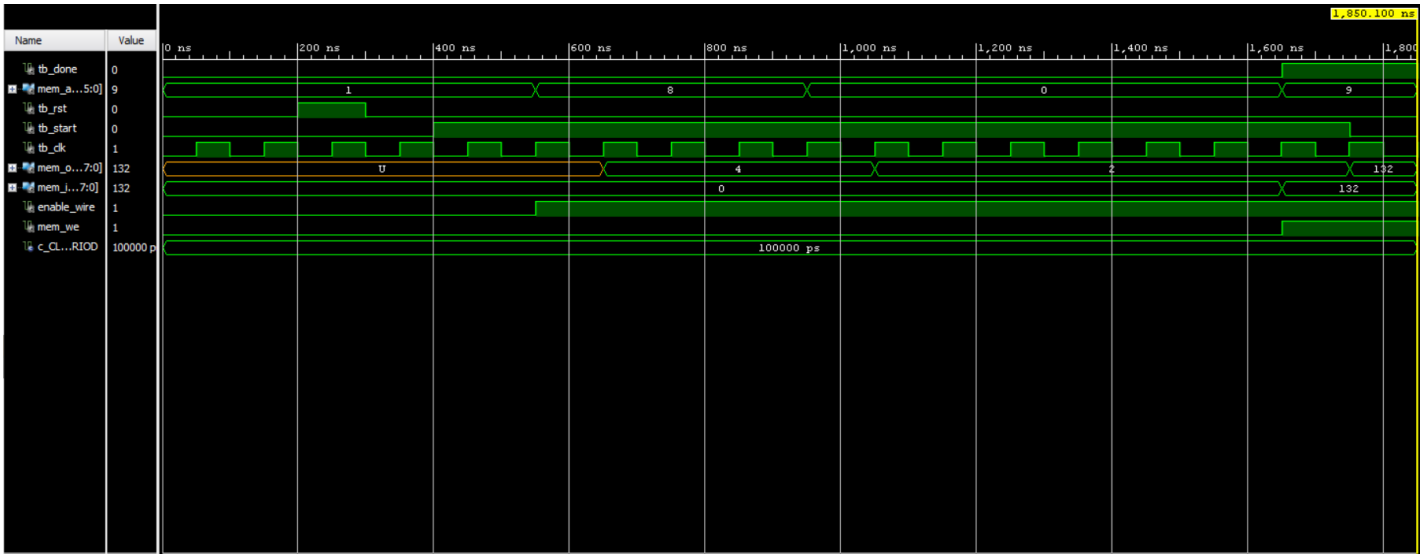
Scelte di design

Il componente è stato progettato seguendo un approccio prevalentemente di tipo comportamentale, utilizzando un unico process sensibile alle variazioni dei segnali `i_clk` e `i_rst`; inoltre è stato fatto in modo che l'aggiornamento degli stati avvenga in prossimità dei fronti di risalita del clock. Per quanto riguarda poi l'algoritmo usato per determinare l'eventuale working-zone a cui appartiene l'indirizzo dato in input, esso si fonda principalmente sul recuperare dalla memoria un solo indirizzo base di una working-zone alla volta, rendendo così il componente facilmente scalabile nel caso in cui venissero modificati il numero delle working-zone presenti in memoria e/o il quantitativo di indirizzi contenute in esse. Nello specifico il componente, una volta salvato in una variabile l'indirizzo da convertire, recupera dalla memoria l'indirizzo base della working-zone 0 e calcola l'offset tra la stringa in input e l'indirizzo base salvato; se esso è minore di 4 il componente memorizza il numero della working-zone e codifica l'offset in one hot, altrimenti il numero della working-zone corrente viene incrementato di uno e si procede a recuperare dalla memoria l'indirizzo di base della nuova working-zone trovata, sfruttando il fatto che i numeri delle working-zone corrispondono agli indici delle celle di memoria in cui sono memorizzati gli indirizzi base delle stesse, per poi ripetere il procedimento appena descritto. Se la stringa in input non appartiene ad alcuna working-zone, condizione verificata quando il numero della working-zone da considerare è pari ad 8 (in questa implementazione), il componente si limita a salvare l'indirizzo di input nella cella di memoria di destinazione.

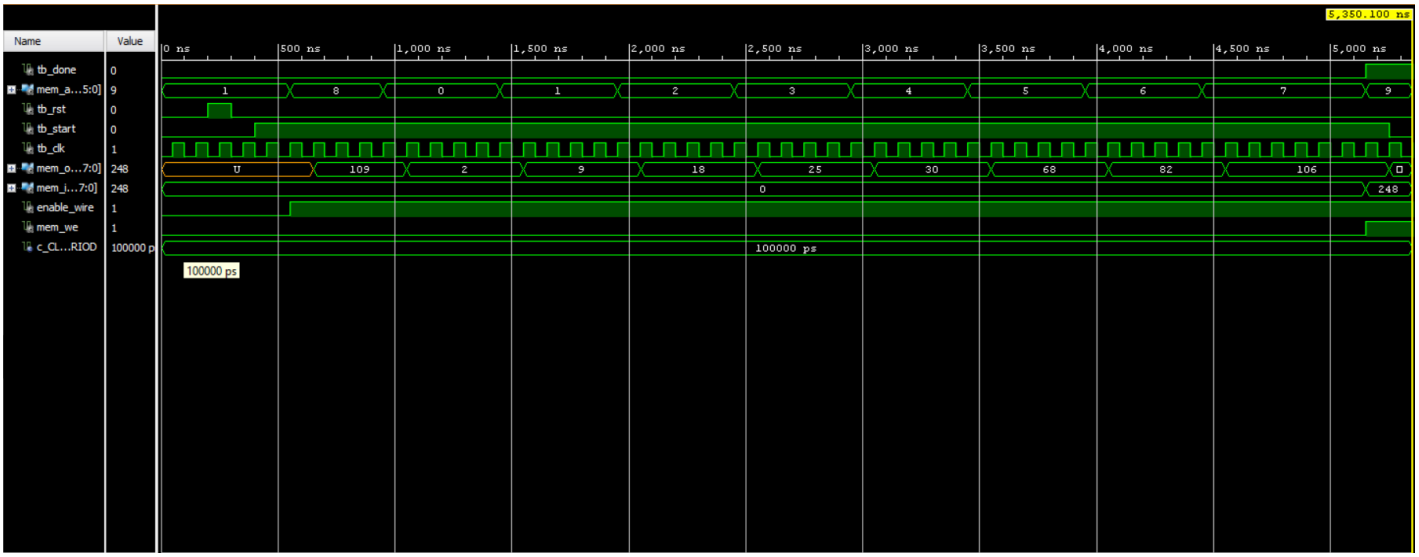
Risultati test

Per quanto riguarda i test effettuati sul componente, si è innanzitutto deciso di farlo operare nel caso in cui l'indirizzo da convertire appartenga alternativamente alla WZ 0 o alla WZ 7; queste situazioni costituiscono infatti i casi limite dell'algoritmo usato per determinare la working-zone a cui appartiene la stringa in input, in quanto esso opera scandendo uno per volta gli indirizzi base delle working-zone partendo dalla cella 0 per arrivare infine alla cella 7. I due test bench, basati su quelli forniti dai docenti, hanno prodotto i seguenti risultati:

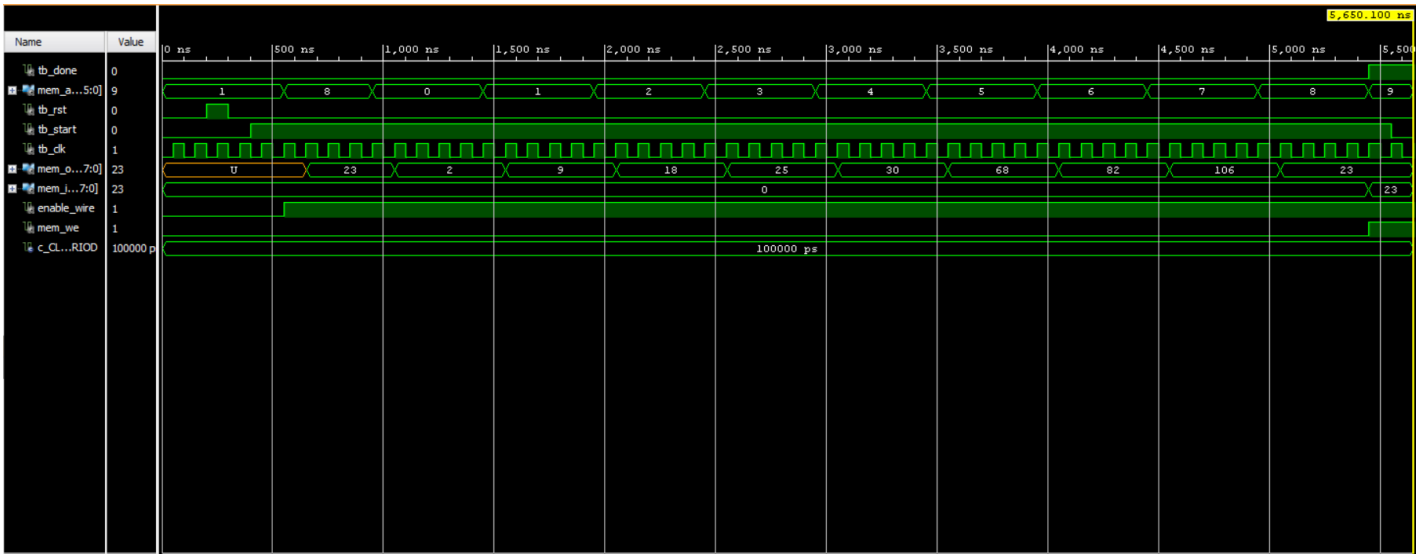
- *Indirizzo appartenente a WZ 0*



- *Indirizzo appartenente a WZ 7*

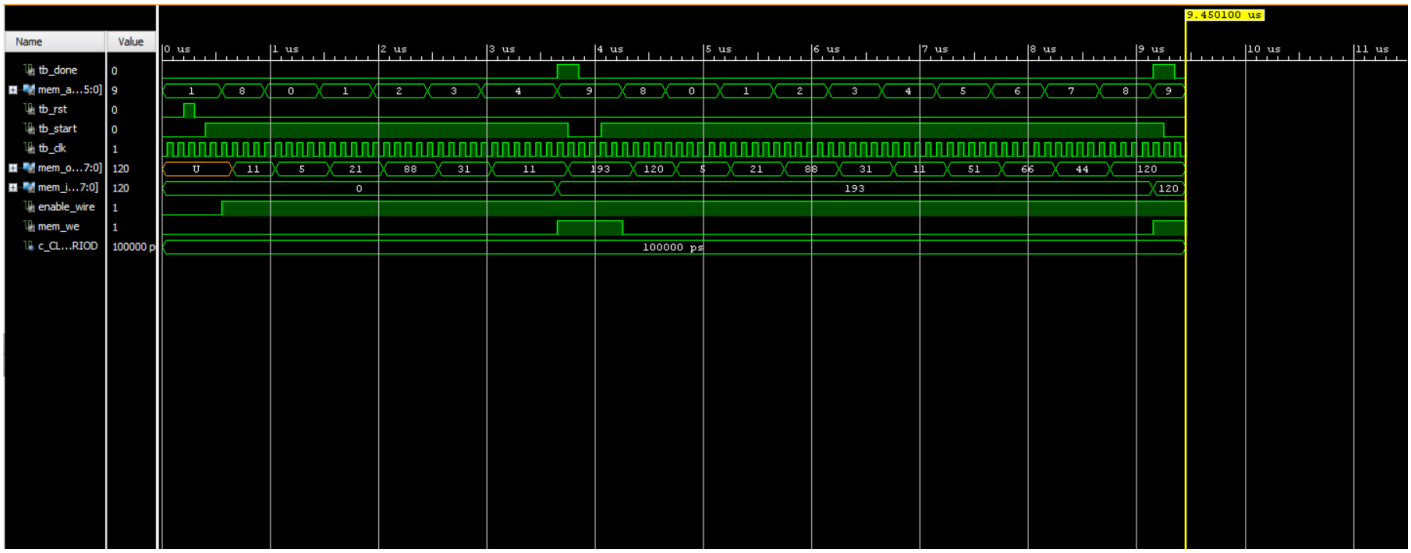


- *Indirizzo non appartenente a nessuna WZ*

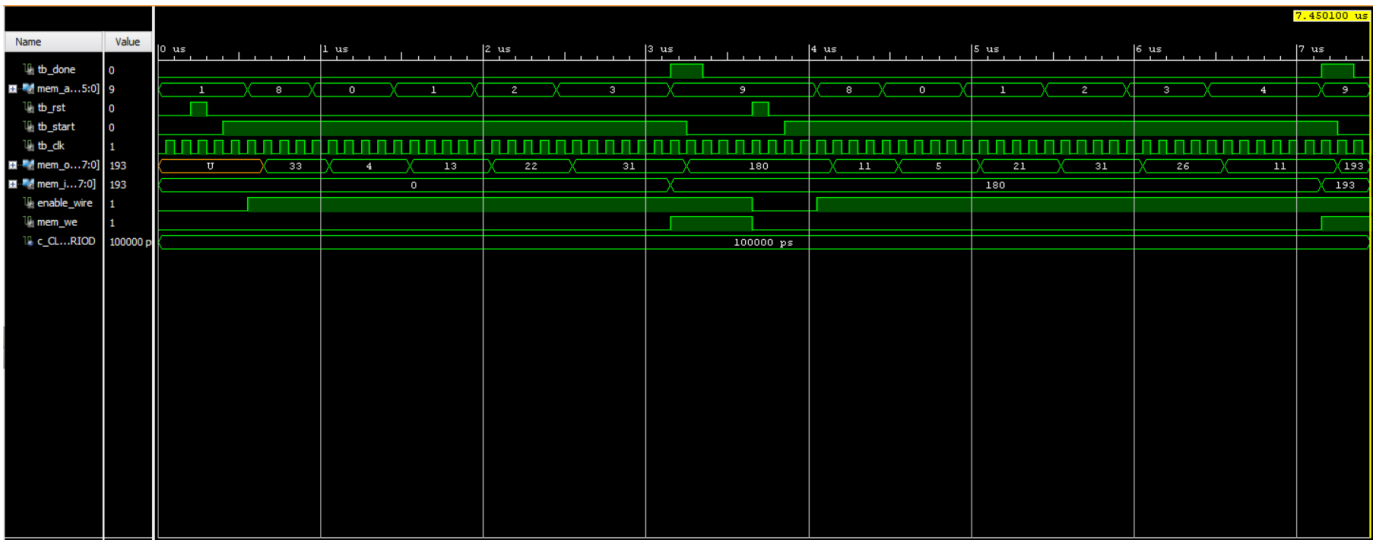


Oltre a questi test, sono state effettuate delle prove per verificare il comportamento del componente nel caso in cui avvenga più di una computazione di seguito, oppure in presenza di molteplici segnali di reset, sia sincroni che asincroni con il segnale di clock:

- *Multi start*



- *Multi reset*



In aggiunta a questi test, il componente è stato sottoposto ad un elevato numero di prove basate sulla generazione in maniera randomica dei valori memorizzati nella RAM, con lo scopo di verificarne il funzionamento in una vasta gamma di situazioni, compresa la ricezione di molteplici segnali `i_start` e di reset asincroni.

Conclusioni

Il componente passa tutti i test precedentemente descritti sia eseguendo simulazioni comportamentali (*Behavioral Simulation*) che con simulazioni funzionali (*Post-synthesis Functional Simulation*). In particolare, sono stati registrati i seguenti tempi di esecuzione:

	Behavioral (ns)	Functional (ns)
In WZ 0	1750	1850,1
In WZ 7	5250	5350,1
No WZ	5550	5650,1
Multi start	9150	9450,1
Multi reset	7150	7450,1

Come si poteva intuire dalla scelta dell'algoritmo usato per determinare l'eventuale working-zone di appartenenza dell'input, quando l'indirizzo dato in input appartenga ad una delle working-zone in memoria, il tempo di esecuzione del componente è direttamente proporzionale al numero della working-zone in questione, mentre nel caso opposto il tempo di computazione è leggermente superiore a quello che si ha con un indirizzo appartenente alla WZ 7.