PROGETTO DI RETI LOGICHE

POLITECNICO DI MILANO
AA 2019-2020

Riccardo Pellini: 10633686
Giuseppe Paolini: 10619082

POLITECNICO
MILANO 1863

1.1 INTRODUZIONE

Lo scopo del progetto è volto alla descrizione in linguaggio VHDL di un componente hardware in grado di codificare indirizzi a 8 bit secondo il metodo *"Working Zone".* Tale codifica, pensata per caratterizzare un Bus Indirizzi, consente una trasmissione di indirizzi più efficiente dal punto di vista energetico, permettendo infatti al mittente di inviare solo un offset piuttosto che un intero indirizzo, a patto che le cosiddette working-zones siano note ad entrambi gli estremi di trasmissione (mittente e ricevente).

1.2 ANALISI DELLA SPECIFICA

Gli indirizzi da codificare sono a 7 bit, la dimensione delle working-zone (WZ) è Dwz = 4, mentre il numero di WZ è fissato a 8. Gli indirizzi base delle WZ sono caricati su una memoria RAM con parole di memoria a 8 bit, in cui il bit più significativo è sempre a 0. Si avranno quindi valori di indirizzi compresi tra 0 e 127.

L'indirizzo da codificare è collocato nell'indirizzo 8 della memoria RAM, mentre l'indirizzo codificato verrà salvato all'indirizzo 9 della medesima memoria.

Indirizzo	Contenuto
0	WZ numero 0
1	WZ numero 1
2	WZ numero 2
…	…
7	WZ numero 7
8	Indirizzo da codificare
9	Indirizzo codificato

La memoria si avvale di 1 bit aggiuntivo ai 7 necessari agli indirizzi: esso svolge la funzione di **flag,** per segnalare o meno la presenza della codifica.
Si supponga che tale bit sia sempre a 0 per gli indirizzi della RAM da 0 a 8.

Ci sono due possibilità di codifica:

I. L'indirizzo da codificare appartiene ad una WZ

In questo caso, tale indirizzo viene codificato nel seguente modo:

i. Bit 7: il più significativo, posto a 1;
ii. Bit 6-4: 3 bit rappresentanti la WZ in questione, codificati in binario (si consideri l'indirizzo della RAM come numero rappresentativo);
iii. Bit 3-0: 4 bit in codifica one-hot, che indicano l'offset dell'indirizzo rispetto alla base della WZ;

Esempio

Indirizzo base WZ: `0-0010010 (0-18)`
Numero WZ: `4`
Indirizzo da codificare: `0-00100100 (20)`
Offset: `20-18 = 2`

Indirizzo codificato: `1-100-0100 (1-4-2)`

II. L'indirizzo da codificare non appartiene ad alcuna WZ

In questo caso, si pone semplicemente il bit più significativo a 0, lasciando immutati i restanti 7 bit. L'indirizzo viene in pertanto tramesso per intero.

Esempio

Indirizzo da codificare: `0-0100110 (0-38)`

Supponendo che non vi siano WZ che contengano tale indirizzo,
si avrà:

Indirizzo codificato: `0-0100110(0-38)`

1.3 INTERFACCIA DEL COMPONENTE

L'interfaccia del componente descritto è la seguente:

```
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 project_reti_logiche;
```

In particolare:

- `i_clk` è il segnale di CLOCK in ingresso generato dal testbench;
- `i_start` è il segnale di START generato dal testbench;
- `i_rst` è il segnale di RESET che inizializza la macchina pronta per ricevere il primo segnale di START;
- `i_data` è il segnale (vettore) che arriva dalla memoria in seguito ad una richiesta di lettura;
- `o_address` è il segnale (vettore) di uscita che manda l'indirizzo alla memoria;
- `o_done` è il segnale di uscita che comunica la fine dell'elaborazione e il dato di uscita scritto in memoria;
- `o_en` è il segnale di ENABLE da dover mandare alla memoria per poter comunicare (sia in lettura che in scrittura);
- `o_we` è il segnale di WRITE ENABLE da dover mandare alla memoria (=1) per poter scriverci. Per leggere da memoria esso deve essere 0;
- `o_data` è il segnale (vettore) di uscita dal componente verso la memoria.

La computazione effettiva avviene quando il segnale `i_start` è alto. La computazione termina quando il segnale o_done si alza. Solo allora, il segnale di start potrà abbassarsi e, conseguentemente, si abbasserà anche il segnale di done;

2.1 FSM

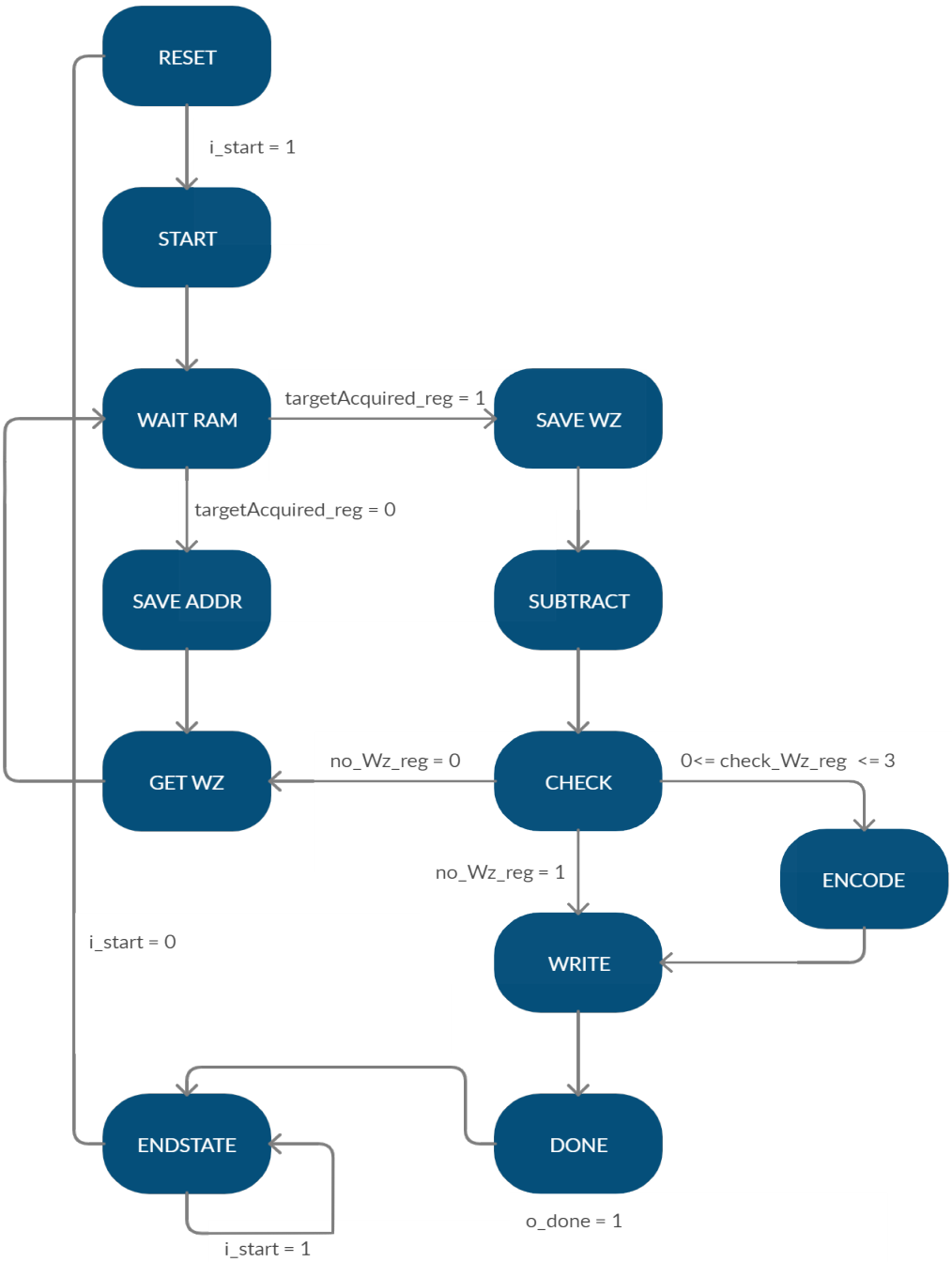
La FSM da noi progettata opera in modo tale che, non appena il segnale di ingresso `i_start` viene alzato a 1, il componente si sposti nello stato `START` per la prima volta e inizi l'elaborazione. In seguito l'indirizzo da confrontare verrà letto da memoria e salvato in un registro.
Iterativamente, da questo punto in poi, l'indirizzo verrà confrontato volta per volta con tutte le WZ lette per verificarne l'appartenenza a una di esse. Alla fine del processo, dopo la scrittura del dato in memoria e subito dopo l'abbassamento del segnale di `i_start` da parte del test bench, la FSM abbassa `o_done` a 0 e si sposta nello stato di RESET, dove si mette in attesa della ricezione di un nuovo segnale di `i_start` per cominciare una nuova elaborazione.
Si è ipotizzato un valore di default dei vari segnali a 0, ad eccezione di `o_data`, il cui valore di default è posto a `10000000`: esso è infatti l'unico valore che non rispetta la codifica (Poiché non ha gli ultimi 4 bit in notazione one-hot).

2.2 STATI DELLA FSM

La macchina è costituita da 12 stati.

- `RESET`: Stato iniziale in cui la macchina è in attesa del segnale di `i_start`. In caso di un segnale di reset, si torna in questo stato.
- `START`: Stato in cui viene scritto in `o_address` l'indirizzo di memoria che deve essere letto, contenente il dato da codificare.
- `WAIT_RAM`: Stato di attesa: si attende la risposta della memoria dopo la richiesta di lettura di un dato eseguita precedentemente. La prima volta che si entra in questo stato, alla memoria verrà chiesto l'indirizzo da analizzare e solo successivamente le varie WZ.
- `SAVE_ADDR`: Stato in cui viene salvato nel registro `target` l'indirizzo da codificare ricevuto dalla memoria.
- `GET_WZ`: Stato dove si tiene conto delle WZ già analizzate e dove vengono settati i registri per richiedere la lettura di quella successiva.
- `SAVE_WZ`: Stato in cui viene salvata nel registro `curr_WZ` la WZ appena letta da memoria.
- `SUBTRACT`: Stato in cui viene calcolato e salvato l'offset tra l'indirizzo da codificare e la working zone corrente nel registro `check_WZ`.
- `CHECK`: Viene eseguito un controllo sull'offset: se è minore di 4 allora lo stato successivo sarà `ENCODE`, sennò bisognerà leggere la prossima WZ e quindi andare nello stato di `GET_WZ`. Nel caso in cui tutte le WZ in memoria siano state controllate, si andrà direttamente nello stato `WRITE`.

- `ENCODE`: Stato in cui viene scritto nel registro `target` il dato codificato secondo la specifica.
- `WRITE`: Stato in cui viene scritto in memoria il dato codificato.
- `DONE`: Stato in cui il segnale `o_done` viene settato a 1 .
- `ENDSTATE`: Stato in cui si attende che il segnale `i_start` venga settato a zero. Non appena succede, la FSM si sposta nello stato di `RESET`.



Ad ogni iterazione, delle righe di codice a inizio processo sono volte all'inizializzazione di tutti i segnali, sfruttando il fatto che essi vengono effettivamente modificati solo a processo finito; questo onde evitare ripetizioni di codice. Si è ritenuto pertanto valido inizializzare il segnale di uscita `o_done` sempre a 0, alzandolo esplicitamente solo nei casi richiesti dalla specifica.

Per l'implementazione della macchina si è scelto un approccio "*Behavioral*".
Sono stati pertanto introdotti diversi segnali con i rispettivi registri, così da distinguere la logica sequenziale dalla logica combinatoria, manipolando così i segnali e memorizzandoli all'interno dei registri.
Allo scopo di facilitare la scrittura della macchina a stati, abbiamo distinto due processi:

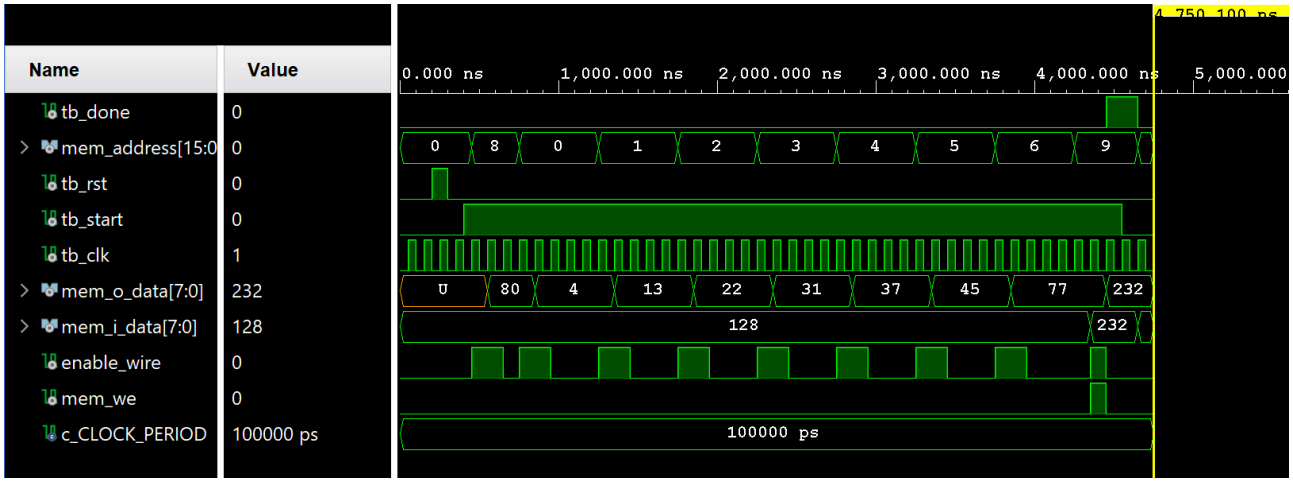
- Un primo processo denominato `STATE_REG`, nel quale viene modificato lo stato dei registri creati memorizzando il valore dei segnali corrispondenti e nel quale avviene la transizione di stato.
- Un secondo processo denominato `WZ_ANALYSIS` dove, tramite l'analisi dei registri, avviene la vera e propria funzione di analisi e codifica dell'indirizzo, determinando lo stato prossimo della macchina e manipolando i diversi segnali, il cui valore verrà poi salvato all'interno dei registri corrispondenti nel processo precedentemente descritto.

3. TESTING

(N.B.: in numeri sono scritti in notazione decimale)

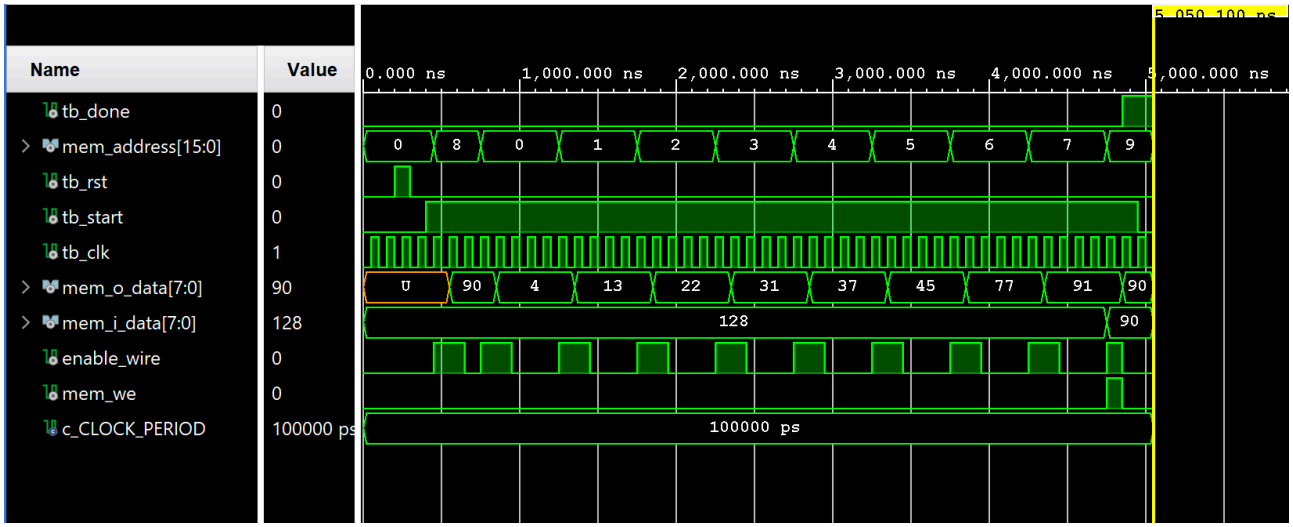
1. INDIRIZZO APPARTENENTE A UNA WZ

   Testa le funzionalità base descritte nella specifica. Assicura che la codifica avvenga in modo corretto.



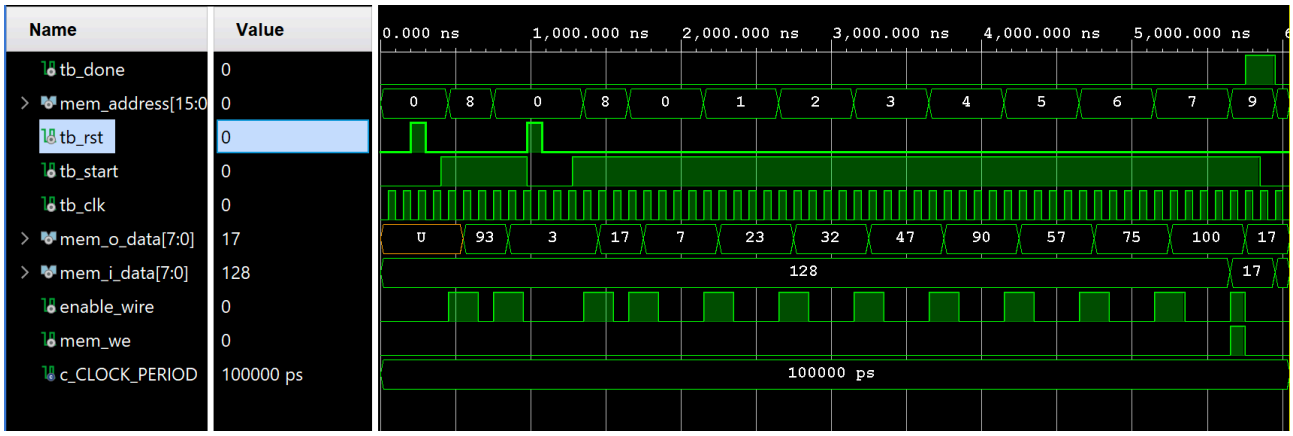
2. INDIRIZZO NON APPARTENENTE A UNA WZ

   Anch'esso testa le funzionalità di base, anche se in condizioni contrarie.



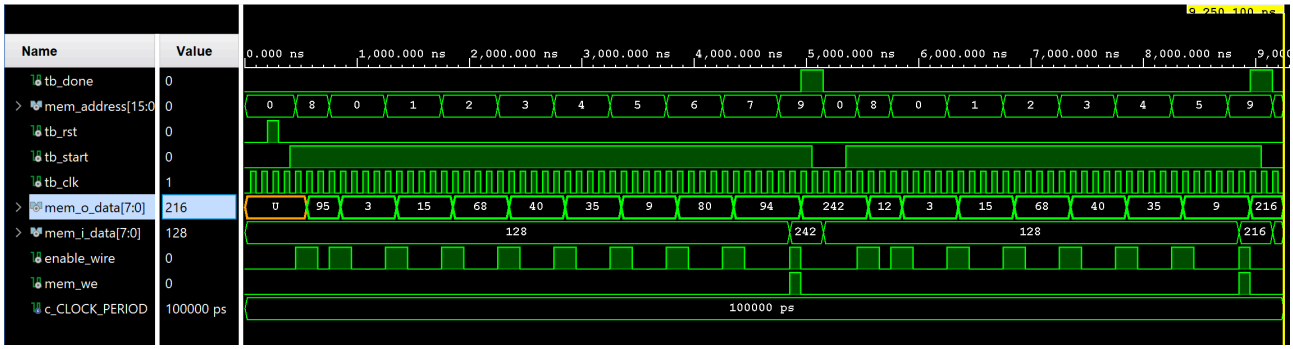
3. SEGNALE DI RESET ASINCRONO

Il testbench genera un reset asincrono per valutare la reazione del componente a tale segnale. Il componente interrompe subito la computazione fino ad un nuovo segnale di start (si noti che i valori contenuti nella RAM, ovvero quelli delle WZ, sono cambiati). La computazione termina normalmente.



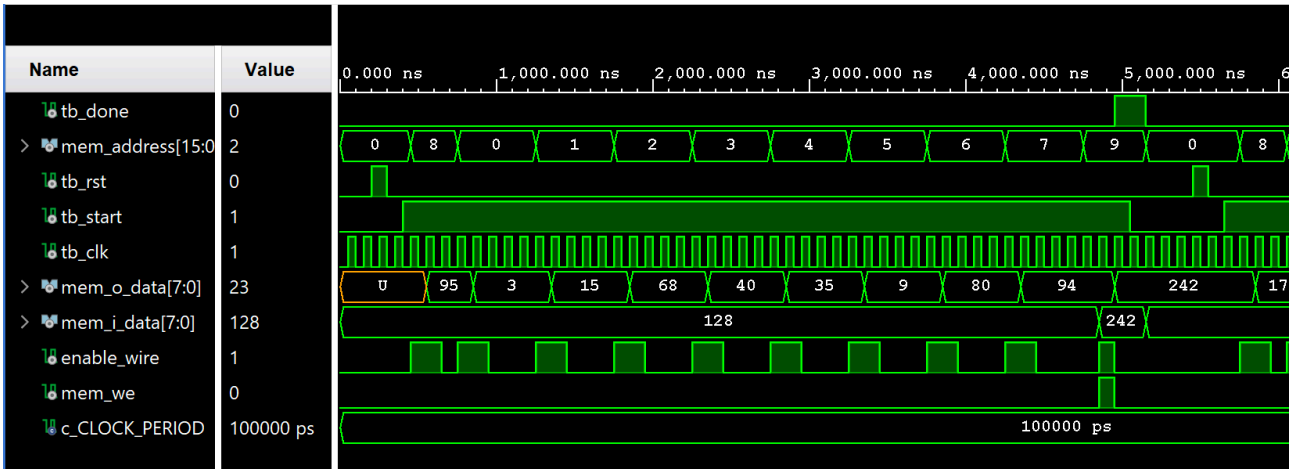
4. SEGNALI DI START MULTIPLI

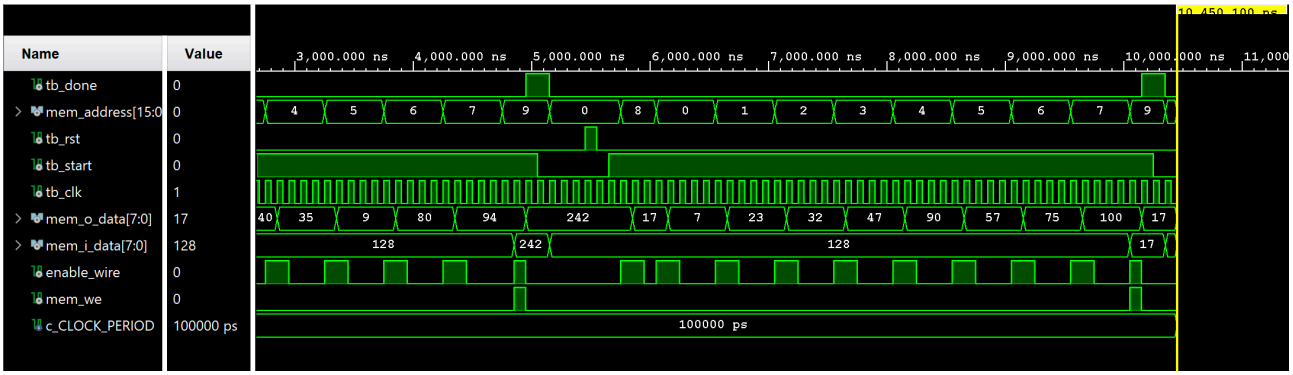
In presenza di WZ a valori costanti tra due computazioni e in assenza di segnali di reset tra esse, può cambiare il valore dell'indirizzo da codificare. Con tale test ci si assicura che i valori di uscita siano congruenti.



5. SEGNALI DI RESET MULTIPLI

Si testa la capacità del componente a cambiamenti nella RAM del testbench tra due computazioni distinte (entrambe portate a termine).
In questo caso infatti, come per il test sul reset asincrono, i valori delle WZ possono cambiare tra una computazione e l'altra.
Il testbench assicura che i risultati siano congruenti a quelli richiesti.





4. CONCLUSIONI

Tutti i test sopra descritti sono stati superati con successo in simulazioni "*Behavioral*" e "*Post-Synthesis Functional*".
Le ottimizzazioni da noi privilegiate sono state rivolte perlopiù all'ottimizzazione della memoria: invece di salvare tutte le working zone in registri appositi prima dell'esecuzione, si è scelto di analizzarle singolarmente volta per volta, permettendo così tempi ridotti di computazione se l'indirizzo da codificare appartiene alle prime WZ in memoria.