Prova finale (Progetto reti logiche)

Prof. Gianluca Palermo – Anno accademico 2019/2020

Lorenzo Ye (Codice persona 10610223 – Matricola 890472)

Indice

Introduzione

Scopo del progetto

Si definisce una “Working Zone” (WZ) un intervallo di dimensioni finite che parte da un indirizzo di base.

Il progetto ha come obiettivo l’implementazione di un componente Hardware descritto in VHDL che, dato in ingresso un indirizzo (che chiameremo ADDR) a 7 bit, sia in grado di codificare tale indirizzo seguendo il metodo di codifica “Working Zone”.

Specifiche ed esempio

Per il progetto sono considerate 8 WZ di dimensione 4 compreso l’indirizzo base.

La codifica di ADDR è differente a seconda del fatto che esso appartenga o meno a una delle 8 WZ, di seguito verranno mostrati 2 esempi di codifica con ADDR = 33.

Esempio ADDR non appartenente a nessuna delle 8 WZ:

8	7	6	5	4	3	2	1
0	0	1	0	0	0	0	1

Se ADDR non appartiene a nessuna WZ viene trasmesso così com’è con l’aggiunta di un 0.

Bit 8: è il bit che indica l’appartenenza a una WZ, in questo caso è 0 perché ADDR non appartiene a nessuna WZ

Bit 7 – 1: ADDR originale invariato.

Esempio ADDR appartenente a una delle 8 WZ:

8	7	6	5	4	3	2	1
1	0	1	1	0	1	0	0

Se ADDR appartiene a una delle WZ viene codificando con 3 informazioni al suo interno.

Bit 8: è il bit che indica l’appartenenza a una WZ, in questo caso è 1 perché ADDR appartiene a una WZ.

Bit 7 – 5: è l’indirizzo di memoria in cui è contenuto l’indirizzo base della WZ a cui ADDR appartiene, codificato in binario su 3 bit. Nel caso dell’esempio l’indirizzo base della WZ è salvato nell’indirizzo 3 della memoria.

Bit 4 – 1: è l’offset rispetto all’indirizzo basse della WZ a cui appartiene ADDR codificato in one hot (0001 se è 0, 0010 se è 1, 0100 se è 2, 1000 se è 3). Nel caso dell’esempio l’offset dall’indirizzo base è 2 quindi ADDR appartiene alla WZ avente indirizzo base 31.

Descrizione della memoria

Le WZ sono contenute nei primi 8 indirizzi della memoria (dal 0 al 7), ADDR è contenuto all'interno dell'indirizzo 8 e l'indirizzo 9 è quello dove la macchina andrà a scrivere la codifica.

WZ 1	0
WZ 2	1
WZ 3	2
WZ 4	3
WZ 5	4
WZ 6	5
WZ 7	6
WZ 8	7
ADDR	8
CODIFICA	9

Interfaccia

Il componente ha la seguente interfaccia:

```
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;
```

Dove:

`i_clk` è il segnale di clock di ingresso.

`i_start` è il segnale di ingresso di start che indica l'inizio della computazione.

`i_rst` è il segnale di ingresso che inizializza la macchina e la porta in uno stato dove aspetta che il segnale di start si alzi.

`i_data` è il vettore di ingresso che contiene il dato in lettura richiesto alla memoria.

`o_address` è il vettore di uscita contenente l'indirizzo di memoria che la macchina vuole leggere.

`o_done` è il segnale di uscita che indica la fine della computazione.

`o_en` è il segnale di enable che indica quando la macchina vuole leggere o scrivere dalla memoria.

`o_we` è il segnale di write enable che indica quando si vuole scrivere in memoria, quando si vuole solo leggere deve essere a 0.

`o_data` è il vettore di ingresso contenente la codifica che la macchina scrive nella memoria.

Architettura

All'inizio dell'esecuzione, dopo che `i_rst` è stato portato a 1, la macchina si troverà nello strato WAIT_START dove attende che `i_start` venga alzato. Dopo aver codificato ADDR e scritto in memoria il risultato, la macchina alza il segnale `o_done`. A questo punto la test bench porterà `i_start` a 0 e la macchina tornerà nello stato WAIT_START in attesa che `i_start` sia postato nuovamente a 1.

Di seguito saranno riportati tutti gli stati della macchina con una breve descrizione.

WAIT_START
È lo stato iniziale dove si attende che `i_start` venga portato a 1. Si torna in questo stato ogni volta che `i_rst` viene alzato.

RAM_REQ
In questo stato si imposta l'indirizzo da richiedere alla memoria. Inizialmente, quando `addr_received = false`, verrà richiesto l'indirizzo contenente ADDR, dopodiché verranno richiesti gli indirizzi contenenti le WZ uno alla volta.

WAIT_FOR_RAM
Questo stato ha l'unico scopo di attendere la risposta della memoria.

MEM
In questo stato si salvano i dati ricevuti dalla memoria, all'inizio (`addr_received = false`) verrà salvato ADDR in `mem_addr` e si ritornerà nello stato RAM_REQ. Se ADDR è già stato salvato verranno salvati i vari WZ in `mem_wz` e si procede passando allo stato CHECK_WZ.

CHECK_WZ
Questo stato controlla se ADDR (salvato in `mem_addr`) appartenga o meno alla WZ salvata in `mem_wz`. Da questo stato si passa a ENCODE solo se ADDR appartiene alla WZ in questione (verrà posto a true il segnale `addr_in_wz`) oppure tutte le WZ sono già state analizzate e quindi ADDR non appartiene a nessuna di esse. Altrimenti si torna allo stato RAM_REQ incrementando l'indirizzo da richiedere alla memoria di 1.

ENCODE
In questo stato avviene la codifica vera e propria di ADDR che viene scritta sull'indirizzo 9 della memoria. Viene portato a 1 il segnale `o_done`.

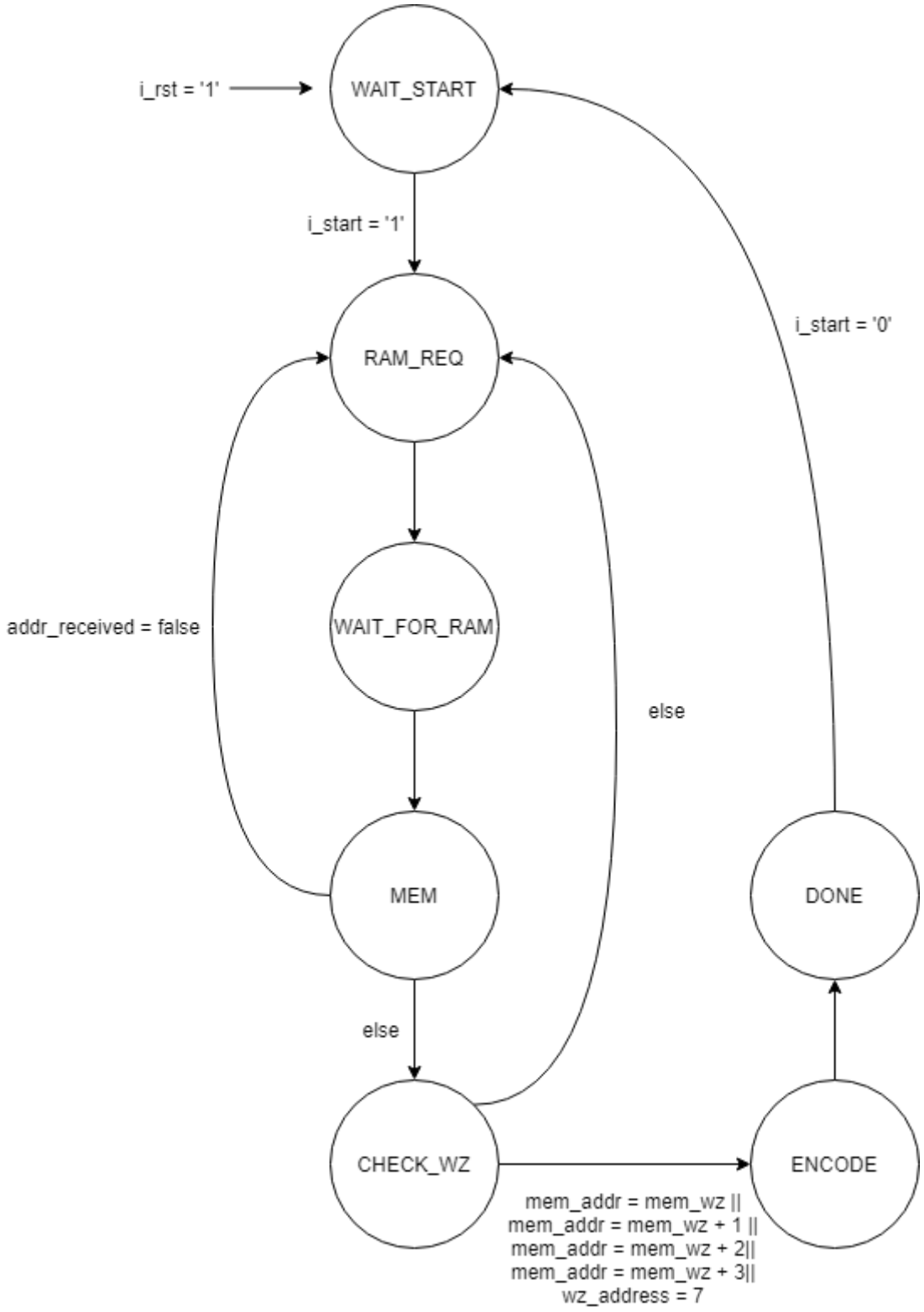
DONE
Questo stato si mette in attesa che `i_start` venga abbassato per tornate nello stato WAIT_START e abbassare `o_done`, re inizializzando tutti i segnali opportuni.

Scelte progettuali

La descrizione della macchina avviene in 2 processi, il primo ha il compito di aggiornare i segnali ad ogni fronte di salita del clock e la gestione del segnale di reset. Il secondo processo rappresenta la FSA che si occupa di analizzare i segnali in ingresso e determinare quale sarà il prossimo stato.

L'algoritmo che si utilizza per la computazione del problema consiste nel memorizzare l'ADDR, per tutta la durata dell'analisi, e man mano memorizzare ed analizzare le singole WZ contenute nella memoria. Si verifica l'appartenenza di ADDR alla WZ esaminata semplicemente controllando se è uguale a uno degli indirizzi all'interno della WZ (indirizzo base ottenuto dalla memoria e gli altri ottenuti incrementando man mano di 1).

Di seguito viene riportata un'immagine sintetica della FSA con i segnali che determinano il passaggio di stato.


i_rst = '1'
WAIT_START
i_start = '1'
RAM_REQ
WAIT_FOR_RAM
addr_received = false
MEM
else
CHECK_WZ
else
i_start = '0'
DONE
ENCODE
mem_addr = mem_wz ||
mem_addr = mem_wz + 1 ||
mem_addr = mem_wz + 2||
mem_addr = mem_wz + 3||
wz_address = 7

Risultati dei test

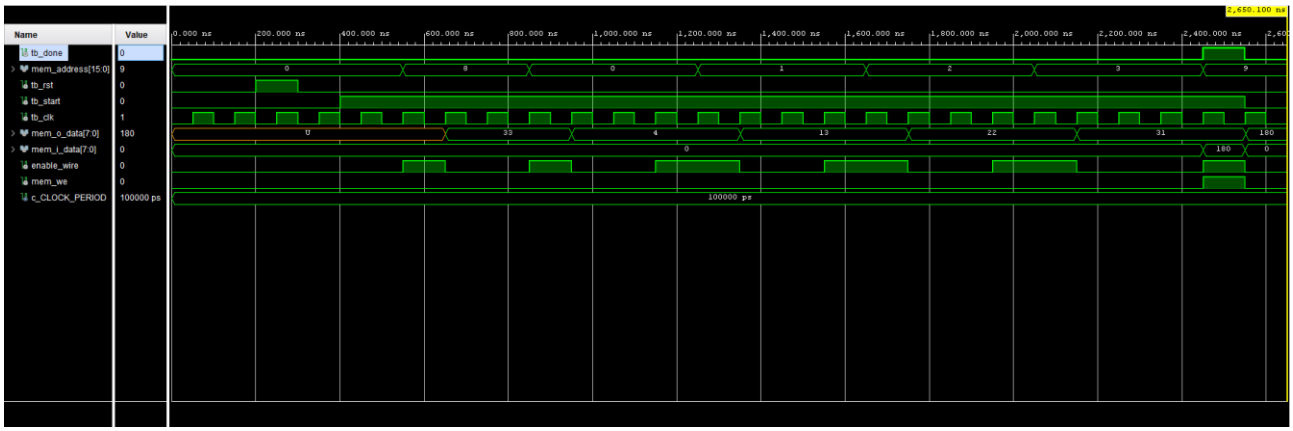
Per testare il funzionamento della macchina post sintesi, oltre alle test bench fornite del Prof. Palermo, è stati definito un test con ADDR appartenente alla prima WZ salvata in memoria. Per verificare il corretto funzionamento dei segnali start e reset sono stati fatti alcuni test nelle seguenti modalità:

Start consecutivi

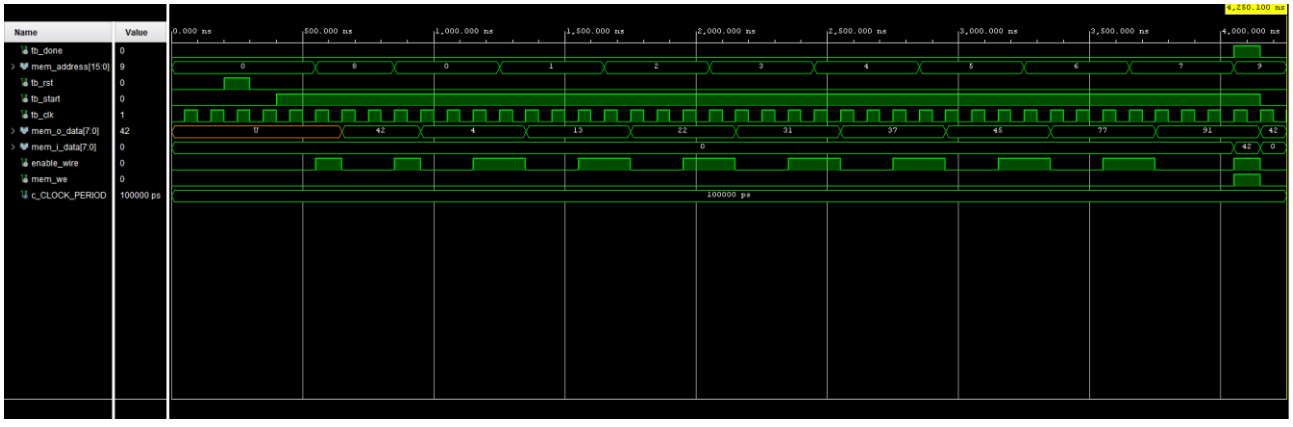
Presenza di reset asincroni

(Tutte le immagini delle wave form sotto riportate sono riferite al post synthesys functional simulation)

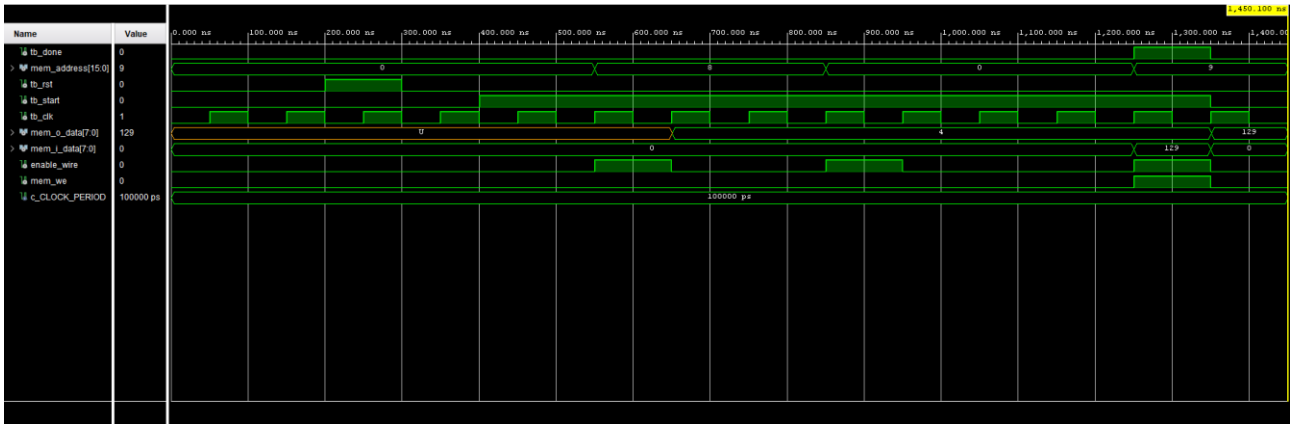
1. **Appartenenza a una WZ**:
   ADDR = 33 appartenente a una WZ (codifica in uscita = 180)



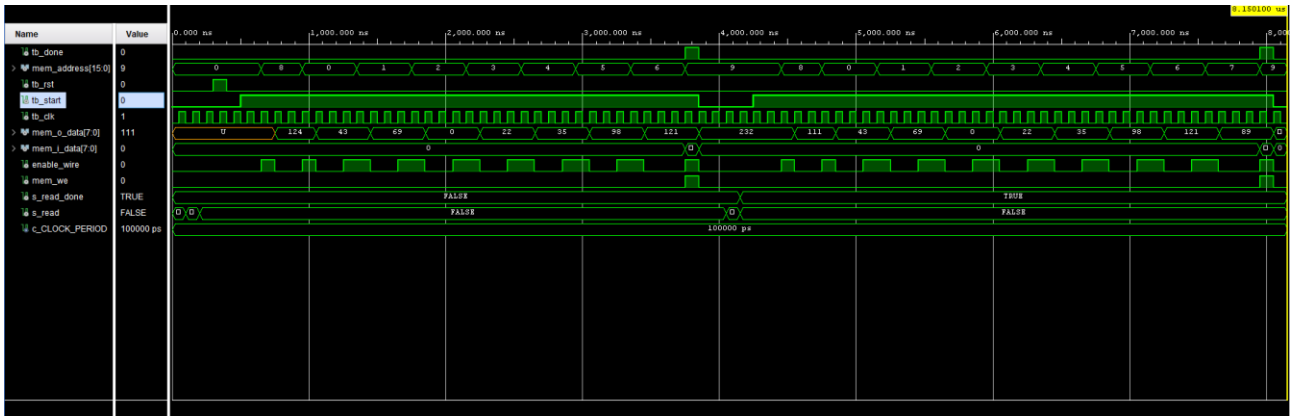
2. **Non appartenenza a una WZ**:
   ADDR = 42 non appartenente a WZ (codifica in uscita = 42)



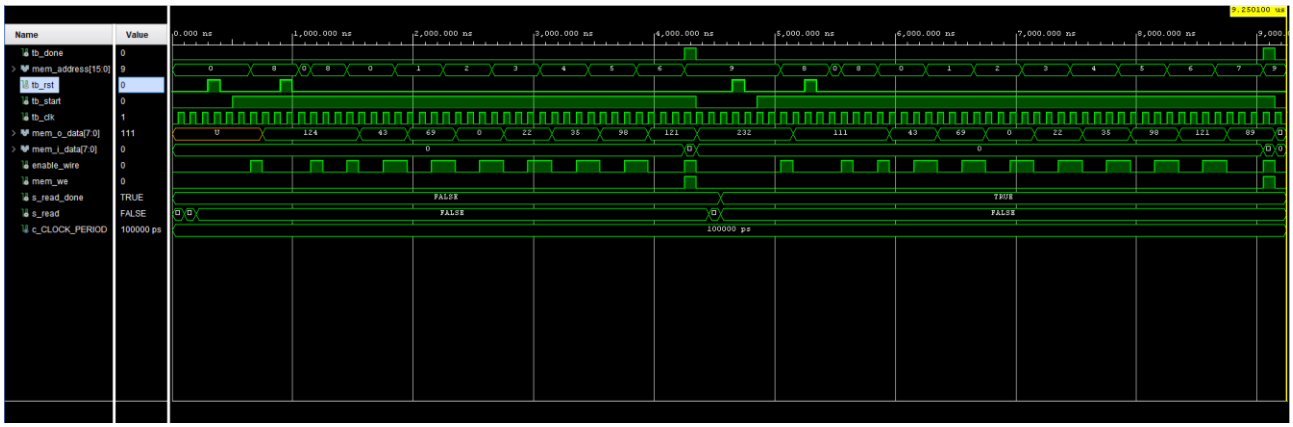
3. **Appartenenza alla prima WZ**:
   ADDR = 4 appartenente alla prima WZ (codifica in uscita = 129)



4. **Start consecutivi**:
   Primo ADDR appartenente a una WZ, il secondo non appartenente:



5. **Reset asincroni**:



Oltre a questi test, per verificare la robustezza del progetto, sono stati effettuati dei test con 1500000 casi randomici.

In fine sono stati creati dei casi di test mirati per coprire ogni cammino della macchina (ogni WZ ogni offset).

Conclusioni

Il componente è sintetizzabile e supera correttamente tutti i test definiti nelle seguenti simulazioni:

Behavioural simulation.

Post Synthesys functional simulation.

Post Synthesys timing simulation.