POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

PROVA FINALE - PROGETTO DI RETI LOGICHE

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

Mattia Piccinato - 10717274 - 957845
Jacopo Piazzalunga - 10747067 - 960405

Advisor:
Prof. Palermo Gianluca

Co-Advisors:
Prof. Antonio Miele

Academic year:
2022-23

Abstract: La presente relazione illustra al dettaglio la nostra soluzione alla specifica assegnata. Ad elevato livello di astrazione, il sistema riceve indicazioni circa una locazione di memoria, il cui contenuto deve essere indirizzato verso un canale di uscita fra i quattro disponibili. Le indicazioni circa il canale da utilizzare e l'indirizzo di memoria a cui accedere vengono forniti mediante un ingresso seriale da un bit, mentre le uscite del sistema, ovvero i succitati canali, forniscono tutti i bit della parola di memoria in parallelo.

1. Specifica di Progetto

1.1 Interfacce

Il modulo da implementare ha due ingressi primari da 1 bit (W e START) e 5 uscite primarie.

Le uscite sono le seguenti: quattro da 8 bit (Z0, Z1, Z2, Z3) e una da 1 bit (DONE). Inoltre, il modulo ha un segnale di clock CLK, unico per tutto il sistema, e un segnale di reset RESET, anch'esso unico.

1.2 Funzionamento

All'istante iniziale, quello relativo al reset del sistema, le uscite hanno i seguenti valori: Z0, Z1, Z2 e Z3 sono 0000 0000, DONE è 0.

I dati in ingresso, ottenuti come sequenze sull'ingresso primario seriale W lette sul fronte di salita del clock, sono organizzati nel seguente modo:

- 2 bit di intestazione (i primi della sequenza)
- N bit di indirizzo della memoria.

Gli N bit permettono di costruire un indirizzo di memoria (si legga qui di seguito la specifica per questi N bit). All'indirizzo di memoria è memorizzato il messaggio da 8 bit che deve essere indirizzato verso un canale di uscita.

I due bit di intestazione identificano il canale d'uscita (Z0, Z1, Z2 o Z3) sul quale deve essere indirizzato il messaggio. Il primo bit è il bit più significativo del canale di uscita, il secondo quello meno significativo, più in dettaglio: 00 identifica Z0, 01 identifica Z1, 10 identifica Z2 e, infine, 11 identifica Z3.

Gli N bit di indirizzo possono variare da 0 fino ad un massimo di 16 bit. Gli indirizzi di memoria sono tutti di 16 bit. Se il numero di bit di N è inferiore a 16, l'indirizzo viene esteso con 0 sui bit più significativi.

La sequenza di ingresso è valida quando il segnale START è alto (=1) e termina quando il segnale START è basso (=0). Il segnale START rimane alto per almeno di 2 cicli di clock e non più di 18 cicli di clock (2 bit del canale e 16 bit per il massimo numero di bit per indirizzare la memoria). Si assuma questa condizione sempre verificata (non è necessario gestire il caso in cui il segnale di START rimanga attivo meno di 2 cicli di clock o più di 18).

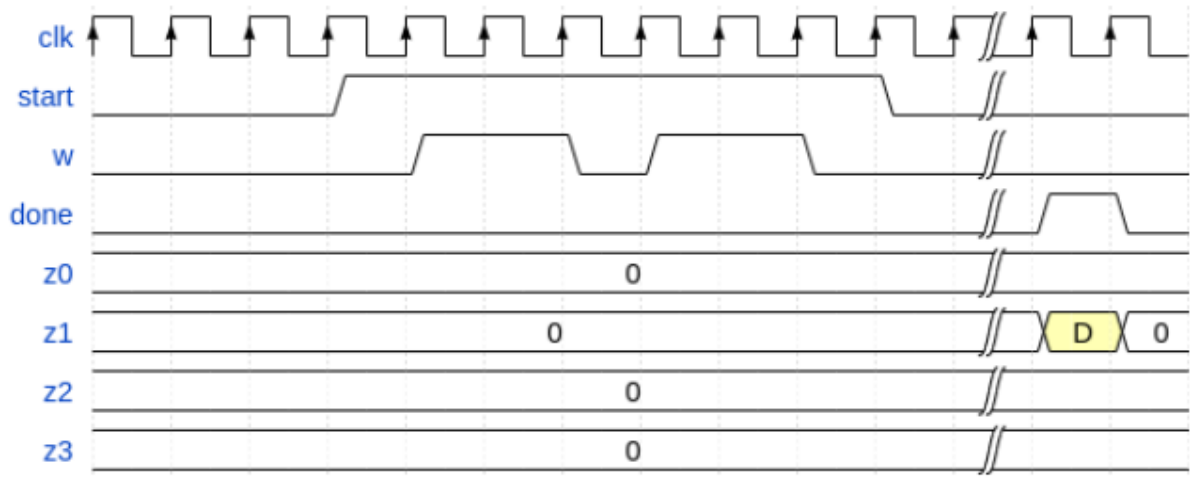
Le uscite Z0, Z1, Z2 e Z3 sono inizialmente 0. I valori rimangano inalterati eccetto il canale sul quale viene mandato il messaggio letto in memoria; i valori sono visibili solo quando il valore di DONE è 1, mentre quando il segnale DONE è 0 tutti i canali Z0, Z1, Z2 e Z3 devono essere a zero (32 bit a 0).

Contemporaneamente alla scrittura del messaggio sul canale, il segnale DONE passa da 0 passa a 1 e rimane attivo per un solo ciclo di clock (dopo 1 ciclo di clock DONE passa da 1 a 0). In pratica, quando DONE=1 il canale associato al messaggio cambierà il suo valore, mentre gli altri canali mostreranno l'ultimo valore trasmesso derivato dai messaggi ad essi associati. Il segnale START è garantito rimanere a 0 fino a che il segnale DONE non è tornato a 0.

Il modulo deve essere progettato considerando che prima del primo START=1 (e prima di richiedere il corretto funzionamento del modulo) verrà sempre dato il RESET (RESET=1). Una seconda (o successiva) elaborazione con START=1 non dovrà invece attendere il reset del modulo. Ogni qual volta viene dato il segnale di RESET (RESET=1), il modulo viene re-inizializzato.

1.3 Esempio

Lettura di un dato "D" dall'indirizzo di memoria 0000000000010110.
La scrittura del dato "D" avviene sull'uscita specificata Z1 (bit di intestazione "01").



1.4 Interfaccia del modulo in VHDL

Lettura di un dato "D" dall'indirizzo di memoria 0000000000010110.
La scrittura del dato "D" avviene sull'uscita specificata Z1 (bit di intestazione "01").

```
entity project_reti_logiche is
    port (
        i_clk   : in std_logic;
        i_rst   : in std_logic;
        i_start : in std_logic;
        i_w     : in std_logic;

        o_z0    : out std_logic_vector(7 downto 0);
        o_z1    : out std_logic_vector(7 downto 0);
        o_z2    : out std_logic_vector(7 downto 0);
        o_z3    : out std_logic_vector(7 downto 0);
        o_done  : out std_logic;

        o_mem_addr : out std_logic_vector(15 downto 0);
        i_mem_data : in std_logic_vector(7 downto 0);
        o_mem_we   : out std_logic;
        o_mem_en   : out std_logic
    );
end project_reti_logiche;
```

1.5 Descrizione della memoria

Lettura di un dato "D" dall'indirizzo di memoria 0000000000010110.
La scrittura del dato "D" avviene sull'uscita specificata Z1 (bit di intestazione "01").

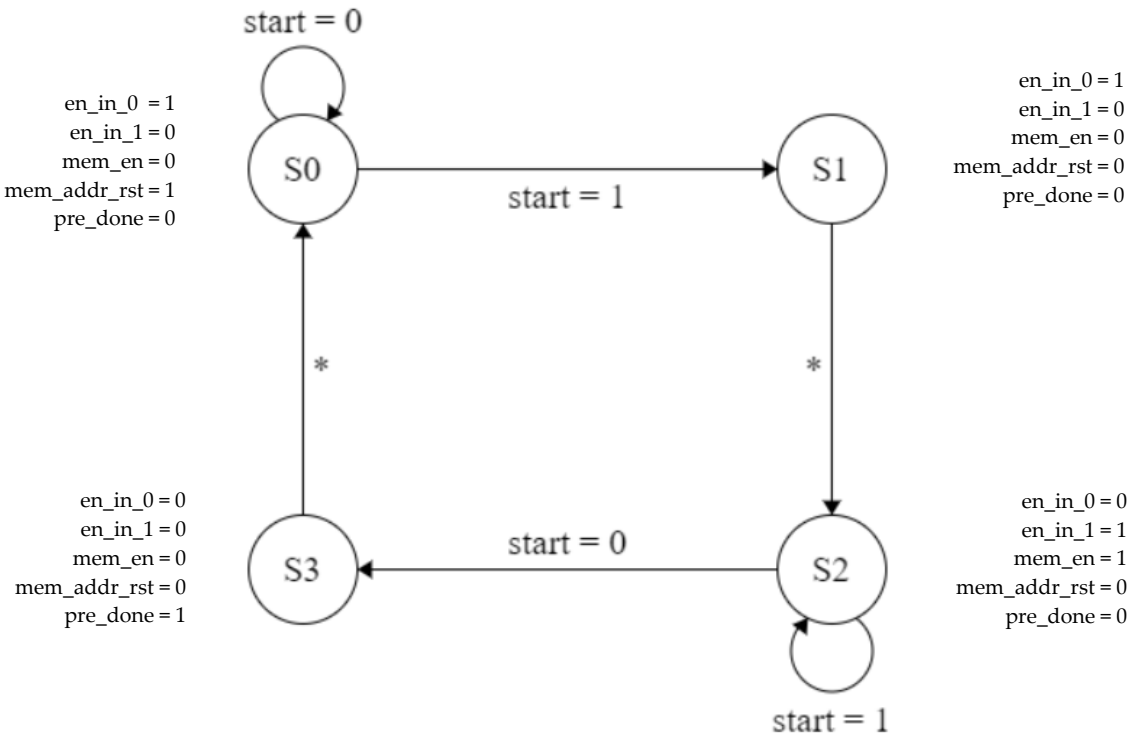
```
-- Single-Port Block RAM Write-First Mode (recommended template)
--
-- File: rams_02.vhd
--
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity rams_sp_wf is
port(
  clk  : in  std_logic;
  we   : in  std_logic;
  en   : in  std_logic;
  addr : in  std_logic_vector(15 downto 0);
  di   : in  std_logic_vector(7 downto 0);
  do   : out std_logic_vector(7 downto 0)
);
end rams_sp_wf;

architecture syn of rams_sp_wf is
type ram_type is array (65535 downto 0) of std_logic_vector(7 downto 0);
signal RAM : ram_type;
begin
  process(clk)
    begin
    if clk'event and clk = '1' then
      if en = '1' then
        if we = '1' then
          RAM(conv_integer(addr)) <= di;
          do                      <= di after 2 ns;
        else
          do <= RAM(conv_integer(addr)) after 2 ns;
        end if;
      end if;
    end if;
  end process;
end syn;
```

2. Architettura

L'architettura concepita è costituita dal *Datapath* e da una macchina a stati che controlla gli ingressi e gestisce il funzionamento del modulo.
Il componente è stato progettato e simulato utilizzando *Xilinx Vivado*; l'FPGA target è la *Xilinx* Artix-7 xc7a200tfbg484-1.

2.1 Macchina a stati

Lo stato di reset S0 attende che venga attivato l'ingresso *start*, leggendo iterativamente ad ogni ciclo tutti i bit di *W* che si presentano prima del fronte di salita di *start*; lo stato S1, in corrispondenza di tale evento, legge il bit più significativo dell'indirizzo del canale di uscita. In generale, S0 ed S1 hanno lo scopo di mantenere alto il segnale di *enable* del registro *reg_output_address* del *Datapath*.



Di conseguenza, lo stato S2 legge e memorizza il bit meno significativo dell'identificativo del canale di uscita, e, di seguito, uno per volta i bit dell'indirizzo della cella di memoria che deve essere letta, fintanto che il segnale *start* vale 1 (eventualmente nessuno), tenendo alto il segnale di enable del rispettivo registro del *Datapath*; lo stato S2 alza e mantiene alto anche il segnale di enable della memoria *mem_en*, in modo che il valore letto in memoria venga pubblicato già al ciclo di clock successivo, se necessario.

Lo stato S3 alza il segnale *pre_done*, disattivando i segnali di enable della memoria e dei registri precedentemente menzionati. Dopodiché, torna allo stato S0, in cui viene mostrato il risultato desiderato per una durata di un ciclo di clock.

2.1.1 Segnali di ingresso e uscita

NOME	TIPO	I/O	DESCRIZIONE
i_start	std_logic	input	segnale di ingresso *i_start* del modulo
en_in_0	std_logic	output	segnale di enable del registro dove viene salvato il canale di uscita
en_in_1	std_logic	output	segnale di enable del registro dove viene salvato l'indirizzo di memoria
mem_en	std_logic	output	segnale di enable della memoria
mem_addr_rst	std_logic_vector	output	segnale che forza il reset del solo registro dell'indirizzo di memoria
pre_done	std_logic	output	segnale di uscita *o_done* del modulo anticipato di un ciclo di clock

2.1.2 Lo stato di reset S0

Lo stato di reset S0 itera fintanto che l'ingresso $start$ = 0 e predispone i segnali di controllo del $Datapath$ in modo che il registro di memorizzazione del canale di uscita sia attivo (en_in_0 alto), che il registro di memorizzazione dell'indirizzo sia inattivo (en_in_1 basso) e resettato (mem_addr_rst alto), mantenendo bassa l'uscita pre_done. Di conseguenza, lo stato S0 continua a leggere e memorizzare un bit alla volta l'ingresso W, fino a quando start = 1. Dal momento che la specifica garantisce che il segnale $start$ rimanga alto per almeno due cicli di clock, tutti i bit memorizzati in questo stato verranno sovrascritti dai bit di interesse nei due stati successivi.

2.1.3 Lo stato S1

Lo stato S1 legge e memorizza il bit più significativo del canale d'uscita, mantiene en_in_0 a 1, en_in_1 a 0 e pre_done a 0, e interrompe il reset del registro dove è salvato l'indirizzo della cella di memoria (mem_addr_rst basso) mantenendola inattiva (mem_en basso).

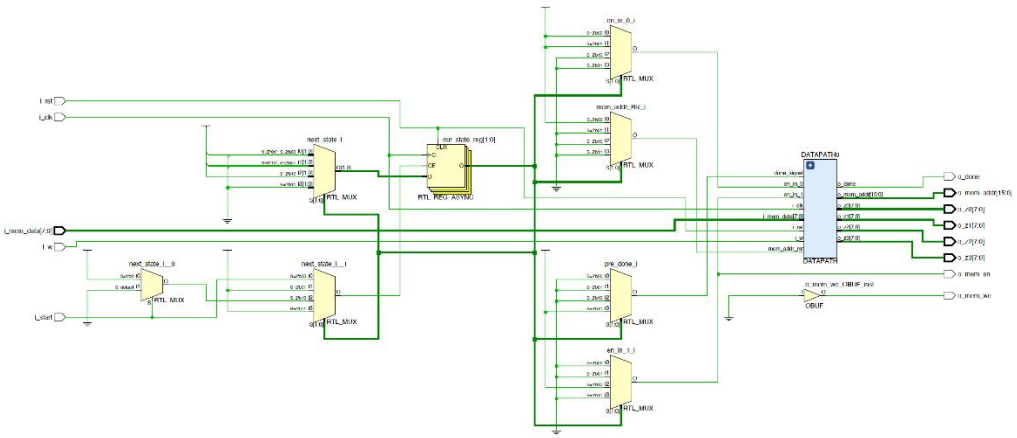
2.1.4 Lo stato S2

Lo stato S2 legge e memorizza il bit meno significativo del canale d'uscita. Pone en_in_0 a 0 (cosicché il valore salvato al suo interno sia congelato), en_in_1 a 1 ed attiva la memoria (mem_en alto), al fine di leggere e memorizzare un bit alla volta l'indirizzo della cella di memoria da W fintanto che $start$ è alto. Nel caso in cui start rimane alto per soli due cicli di clock, S2 transita direttamente ad S3. Inoltre, mantiene basse le uscite mem_addr_rst e pre_done.

2.1.5 Lo stato S3

Lo stato S3 riporta mem_en a 0, cosicché il valore in uscita sulla memoria sia congelato ed alza pre_done. Mantiene en_in_0 e mem_addr_rst a 0, e pone en_in_1 a 0. Si noti che l'uscita del modulo $done$ è alta nel primo ciclo di clock dello stato S0. Lo stato S3 legge sempre un bit non

di interesse memorizzandolo nel registro utilizzato per l'indirizzo della cella di memoria, il che, tuttavia, non comporta alcuna situazione patologica in quanto il valore letto dalla memoria viene pubblicato durante la suddetta lettura, di fatto ignorandola.



2.1.6 Casi limite per la macchina a stati

Se il segnale di ingresso start resta alto per due soli cicli di clock, come affermato in precedenza, lo stato S2 transita direttamente allo stato S3 senza iterare. Il modulo risponde correttamente al suddetto caso limite, in quanto il registro che contiene l'indirizzo della cella di memoria viene azzerato prima della lettura. Il bit superfluo letto dallo stato S3 non è dunque patologico (come spiegato nel precedente paragrafo).

Se il segnale di ingresso start resta alto per 18 cicli di clock, tutti i bit dell'indirizzo della cella di memoria vengono salvati nell'apposito registro, il quale è in grado di memorizzare esattamente 16 bit. Il bit superfluo letto dallo stato S3 non è patologico (come spiegato nel precedente paragrafo).

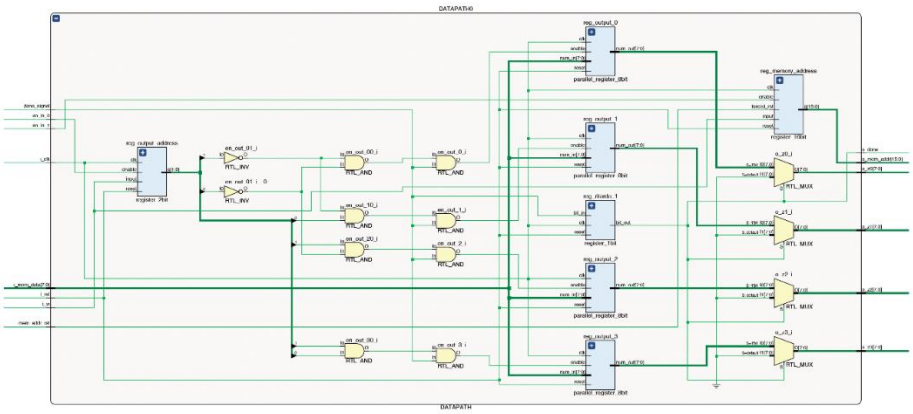
La macchina a stati è stata progettata affinché il modulo funzioni sia per una singola iterazione del processo, che per un numero indefinito di iterazioni, tramite un idoneo ripristino dei segnali di uscita definito sulle assunzioni garantite dalle specifiche. Questo varrebbe anche se il segnale di start si dovesse alzare appena dopo la disattivazione del segnale di uscita *done* del modulo.

2.1.7 Minimizzazione

La macchina a stati proposta è evidentemente minima, in quanto è una macchina di Moore in cui tutti gli stati hanno uscite diverse da ogni altro.

2.2 Datapath

Il *Datapath* è costituito da 2 registri a scorrimento serie-parallelo, di cui il primo da 2 bit (*reg_output_address*) ed il secondo da 16 bit (*reg_memory_address*), un flip flop per ritardare di un ciclo il segnale *pre_done* (*reg_ritardo*_1) ed un registro per ogni uscita del modulo (*reg_output*_1/4), a cui si aggiunge della logica combinatoria e vari segnali di ingresso ed uscita.



2.2.1 Segnali di ingresso ed uscita

NOME	TIPO	I/O	DESCRIZIONE
done_signal	std_logic	input	uscita *pre_done* della macchina a stati
en_in_0	std_logic	input	uscita *en_in*_0 della macchina a stati
en_in_1	std_logic	input	uscita *en_in*_1 della macchina a stati
i_clk	std_logic	input	segnale di clock del modulo
i_mem_data[7:0]	std_logic_vector	input	valore letto dalla memoria
i_rst	std_logic	input	segnale di reset del modulo
i_w	std_logic	input	ingresso *i_w* del modulo
mem_addr_rst	std_logic	input	uscita *mem_addr_rst* della macchina a stati
o_mem_addr[15:0]	std_logic_vector	output	indirizzo della cella di memoria da leggere
o_done	std_logic	output	uscita *o_done* del modulo
o_z0[7:0]	std_logic_vector	output	uscita 0_*z*0 del modulo
o_z1[7:0]	std_logic_vector	output	uscita *o_z*1 del modulo
o_z2[7:0]	std_logic_vector	output	uscita *o_z*2 del modulo
o_z3[7:0]	std_logic_vector	output	uscita *o_z*3 del modulo

2.2.2 Registro reg_memory_address

Registro serie-parallelo da 16 bit sincrono con segnale di enable e segnale asincrono di reset, utilizzato per memorizzare l'indirizzo della cella di memoria da cui il modulo deve leggere.

La scelta di questa tipologia di registro è motivata dalla semplificazione che introduce rispetto al processo di lettura e memorizzazione dell'indirizzo di memoria, poiché fornisce in uscita

l'indirizzo su 16 bit in un solo ciclo di clock, indipendentemente dal numero di bit ricevuti in ingresso.

2.2.3 Flip-flop reg_ritardo_1

Flip-flop D con segnale asincrono di reset, utilizzato per ritardare l'uscita *pre_done* della macchina a stati, in modo da allineare tale segnale all'uscita nel momento in cui quest'ultima è pronta per essere pubblicata sulle uscite *z*. L'uscita del registro è, infatti, l'uscita *o_done* del modulo.

2.2.4 Registro reg_output_address_x

Quattro registri parallelo-parallelo da 8 bit sincroni con segnale di enable e segnale asincrono di reset, utilizzati per memorizzare l'ultimo valore mostrato sull'uscita x.

Il ritardo di un ciclo di clock dovuto a questi registri motiva l'utilizzo del flip-flop.

2.2.5 Scelte per la selezione dell'uscita

L'utilizzo dei registri descritti nel paragrafo precedente permettono al modulo di "ricordare" l'ultimo valore mostrato su ogni uscita. Attraverso l'utilizzo di quattro multiplexer controllati dal segnale *o_done*, le quattro uscite mostrano costantemente 0, tranne quando tale segnale è alto.

3. Risultati sperimentali

La progettazione è stata supportata dalla fase di sperimentazione, testando il componente con gli opportuni test bench, sia in casi standard che – soprattutto - nei casi limite. Il componente supera tutti i test a cui è stato sottoposto, sia in simulazione comportamentale che post-sintesi (e post-implementazione). Nella fase di test non sono stati presi in considerazione i ritardi dovuti alla propagazione dei segnali.

3.1 Report di sintesi

Come già affermato nel paragrafo precedente, il componente finale risulta essere correttamente sintetizzabile. In particolare, relativamente alla FPGA target, dal report di sintesi emerge che:

- L'utilizzo percentuale della LUT ammonta allo 0.02%;
- L'utilizzo percentuale dei flip-flop ammonta allo 0.02%, pari ad un numero di 55 flip-flop;
- Il Worst Negative Slack ammonta a 97.09 nano secondi.

3.2 Test bench a clock molto bassi

Il modulo risponde correttamente anche a segnali di clock con periodo molto breve. In particolare, nella maggior parte dei casi, risulta essere funzionante anche a periodi inferiori a 10ns.

3.3 Test bench con reset asincrono

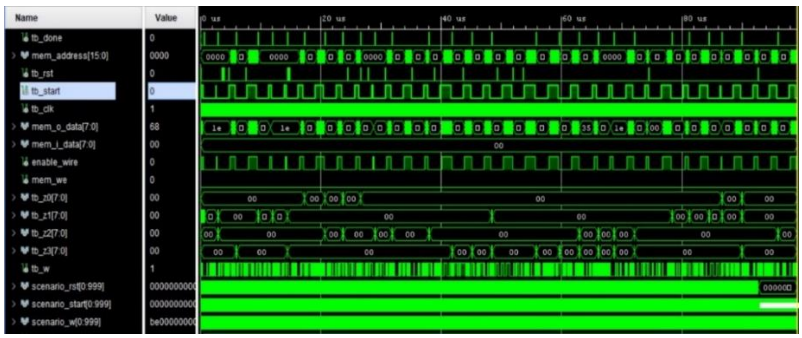
Il modulo risponde correttamente anche nel caso in cui sia attivato un segnale di reset asincrono.

3.4 Test bench con indirizzo di lunghezza limite

Il modulo risponde correttamente, come descritto precedentemente nel paragrafo 2.1, anche con indirizzi in ingresso da 0 e da 16 bit ricevuti in ingresso.

3.5 Test bench con sequenza multipla

Infine, con un breve script in Python, abbiamo generato un test bench di grandi dimensioni e verificato che il modulo risponda correttamente anche se *start* viene attivato molte volte:



Il test bench generato attiva il segnale *start* per mille volte.

4. Conclusioni e possibili ottimizzazioni

Il componente HW descritto soddisfa le specifiche grazie ad un'attenta analisi preliminare e ad un efficace progettazione, in due soli cicli di clock dal momento in cui si abbassa il segnale di start.

4.1 Ottimizzazione

È possibile, d'altronde, ottimizzare ulteriormente il modulo, affinché impieghi un solo ciclo di clock per leggere e canalizzare il valore da mostrare in uscita.

Dal fronte di discesa in corrispondenza del quale si abbassa il segnale di start, un ciclo di clock è comunque necessario affinché il valore letto in memoria venga pubblicato sulla rispettiva uscita.

Attualmente, in aggiunta, il nostro modulo ritarda di un ciclo di clock il segnale alto *pre_done*, affinché il valore letto in memoria venga prima canalizzato e memorizzato nel registro dell'idonea uscita, e poi mostrato al fronte di salita successivo.

Invece di attendere questo ulteriore ciclo di clock, si potrebbe mostrare in uscita il valore letto anche mentre viene memorizzato nel registro, al costo di dover aggiungere quattro ulteriori

multiplexer (uno per ogni uscita), controllati dal medesimo segnale dei già presenti quattro multiplexer.