PROVA FINALE - PROGETTO DI RETI LOGICHE

A.A. 2023/2024

Sadeesha Karagoda Gamage Ranasinghe

Matricola : 902460

Codice persona : 10464209

Docente : Fabio Salice

Introduzione

Descrizione

L'obiettivo del progetto è l'implementazione, tramite VHDL, di un componente hardware il quale legge una sequenza di parole, dati un indirizzo di memoria e un numero di celle di memoria da leggere. Il componente quindi elabora i dati in esse presenti secondo una logica ben precisa (precisata in seguito) e li scrive in memoria, avvisando la fine dell'elaborazione con il segnale DONE.

Interfaccia modulo

Interfaccia del modulo:

```
entity project_reti_logiche is
    port (
        i_clk   : in std_logic;
        i_rst   : in std_logic;
        i_start : in std_logic;
        i_add   : in std_logic_vector(15 downto 0);
        i_k     : in std_logic_vector(9 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_data : out std_logic_vector(7 downto 0);
        o_mem_we   : out std_logic;
        o_mem_en   : out std_logic
    );
end project_reti_logiche;
```

Segnali:

- i_clk : segnale di CLOCK in ingresso (generato dal testbench)
- i_rst : segnale di RESET in ingresso (generato dal testbench); è l'unico segnale asincrono ed è responsabile dell'inizializzazione e re-inizializzazione del modulo
- i_start : segnale di START in ingresso (generato dal testbench); segnala l'inizio dell'elaborazione
- i_add : segnale in ingresso (generato dal testbench) con il quale si riceve l'indirizzo ADD della prima cella di memoria dalla quale inizia l'elaborazione
- i_k : segnale in ingresso che indica quante celle di memoria è necessario leggere
- o_done : segnale di DONE in uscita; viene utilizzato per notificare la fine dell'elaborazione
- o_mem_addr : segnale in uscita che comunica alla memoria l'indirizzo della cella di memoria da cui leggere o in cui scrivere

- i_mem_data : segnale in ingresso con il quale si riceve il dato letto dalla memoria
- o_mem_data : segnale in uscita che comunica alla memoria il dato da scrivere in una determinata cella di memoria
- o_mem_we : segnale di WRITE_ENABLE in uscita verso la memoria; viene settato ad 1 per abilitare la scrittura in memoria, a 0 per la lettura da memoria
- o_mem_en : segnale di ENABLE in uscita verso la memoria; viene settato ad 1 per abilitare operazioni di lettura o scrittura con la memoria

Memoria

Il funzionamento della memoria viene descritto dal seguente codice VHDL:

```
-- 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;
```

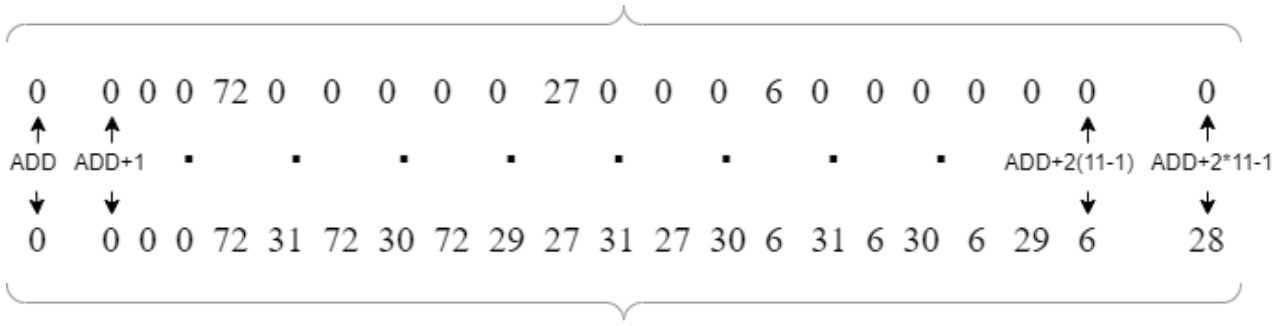
Funzionamento

All’istante iniziale il modulo viene inizializzato mettendo RESET ad 1 e tutte le sue uscite valgono 0.
Quando il segnale in ingresso START sale ad 1, dopo che RESET è sceso a 0, inizia l’elaborazione, durante la quale START rimane ad 1: il componente riceve in ingresso dal segnale i_add un indirizzo di memoria ADD e, a partire da esso, legge una sequenza di K valori (dove K è ricevuto da i_k), ciascuno memorizzato ogni due byte (primo valore all’indirizzo ADD, secondo valore all’indirizzo ADD+2, …, ultimo valore all’indirizzo ADD+2(K-1)). Ogni valore è compreso tra 0 e 255: se esso è diverso da 0, rimane intatto, altrimenti viene sovrascritto con il primo valore diverso da 0 che lo precede nella sequenza (se presente, altrimenti resta 0). In ciascun byte successivo ad uno appartenente alla sequenza (ADD+1, ADD+3, ..., ADD+2K-1) viene scritto un valore chiamato di credibilità: se il valore della sequenza appena letto è diverso da 0, il suo valore di credibilità è pari a 31, altrimenti è pari al valore di credibilità precedente decrementato di 1 (purché esso sia presente e maggiore di zero, se no il nuovo valore di credibilità vale 0).
Il componente notifica la conclusione dell’elaborazione settando il segnale DONE ad 1. Nel ciclo di clock successivo a ciò, START scende a 0, e in quello ancora successivo DONE fa lo stesso. Il componente può ora eseguire nuovamente l’intero processo appena descritto, senza il bisogno di re-inizializzarlo con RESET.

Esempio di diagramma temporale del funzionamento

Esempio di funzionamento dell’elaborazione con K = 11

Sequenza iniziale (prima che START valga 1)

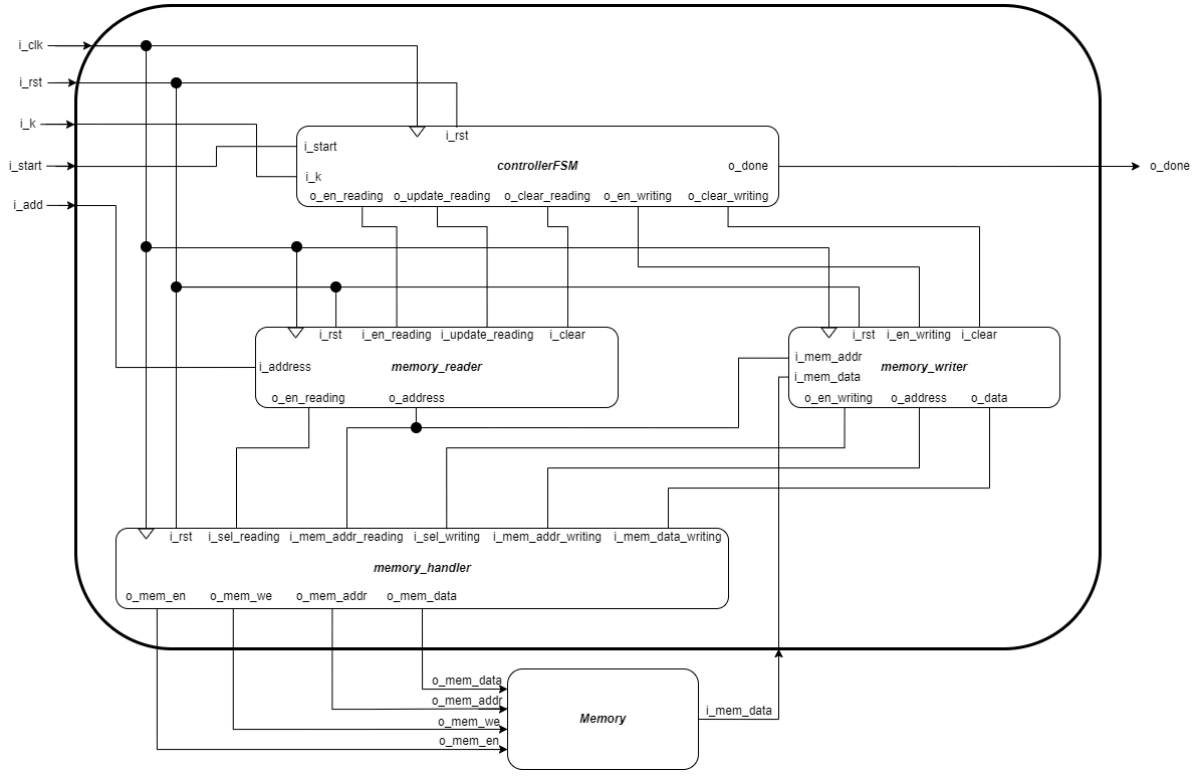


Sequenza finale (dopo l’elaborazione)

Architettura

Il modulo è costituito da 4 sottomoduli diversi, ognuno responsabile di una specifica funzione. Tutti i segnali sono sincroni e vengono interpretati sul fronte di salita del clock, eccetto il segnale RESET.

Rappresentazione dell'architettura del modulo



MEMORY_READER

Viene abilitato o disabilitato dal controllerFSM mediante il segnale (in ingresso) i_en_reading. In ogni istante dell'elaborazione, calcola l'indirizzo di memoria del byte da leggere e lo invia al memory_handler e al memory_writer attraverso il segnale o_address. Inoltre attiva l'operazione di lettura del memory_handler al momento opportuno, mediante il segnale o_en_reading.

MEMORY_WRITER

Viene abilitato o disabilitato dal controllerFSM mediante il segnale (in ingresso) i_en_writing. In ogni dato istante dell'elaborazione, riceve l'indirizzo di memoria del byte considerato da memory_reader, e il valore in esso contenuto dalla memoria,

mediante il segnale i_mem_data; dunque usa questi dati per il calcolo dell'indirizzo del byte in cui scrivere e il valore da scriverci, e lo invia al memory_handler rispettivamente attraverso i segnali o_address e o_data. Inoltre attiva l'operazione di scrittura del memory_handler al momento opportuno, mediante il segnale o_en_writing.

MEMORY_HANDLER

Questo sottomodulo centralizza il controllo delle operazioni di lettura e scrittura con la memoria, di cui è responsabile. Controlla quindi tutti i segnali in uscita verso la memoria (o_mem_en, o_mem_we, o_mem_addr e o_mem_data), i quali sarebbero altrimenti stati gestiti o contemporaneamente da memory_reader e memory_writer, con conseguente rischio di conflitti sui valori in uscita e difficoltà di coordinamento, o da controllerFSM, al costo di minore modularizzazione e maggiore complessità. La sua operazione di lettura è attivata dal sottomodulo memory_reader alzando il segnale i_sel_reading ad 1, mentre quella di scrittura è attivata da memory_writer alzando il segnale i_sel_writing ad 1 (questi non saranno mai contemporaneamente settati ad 1).

CONTROLLERFSM

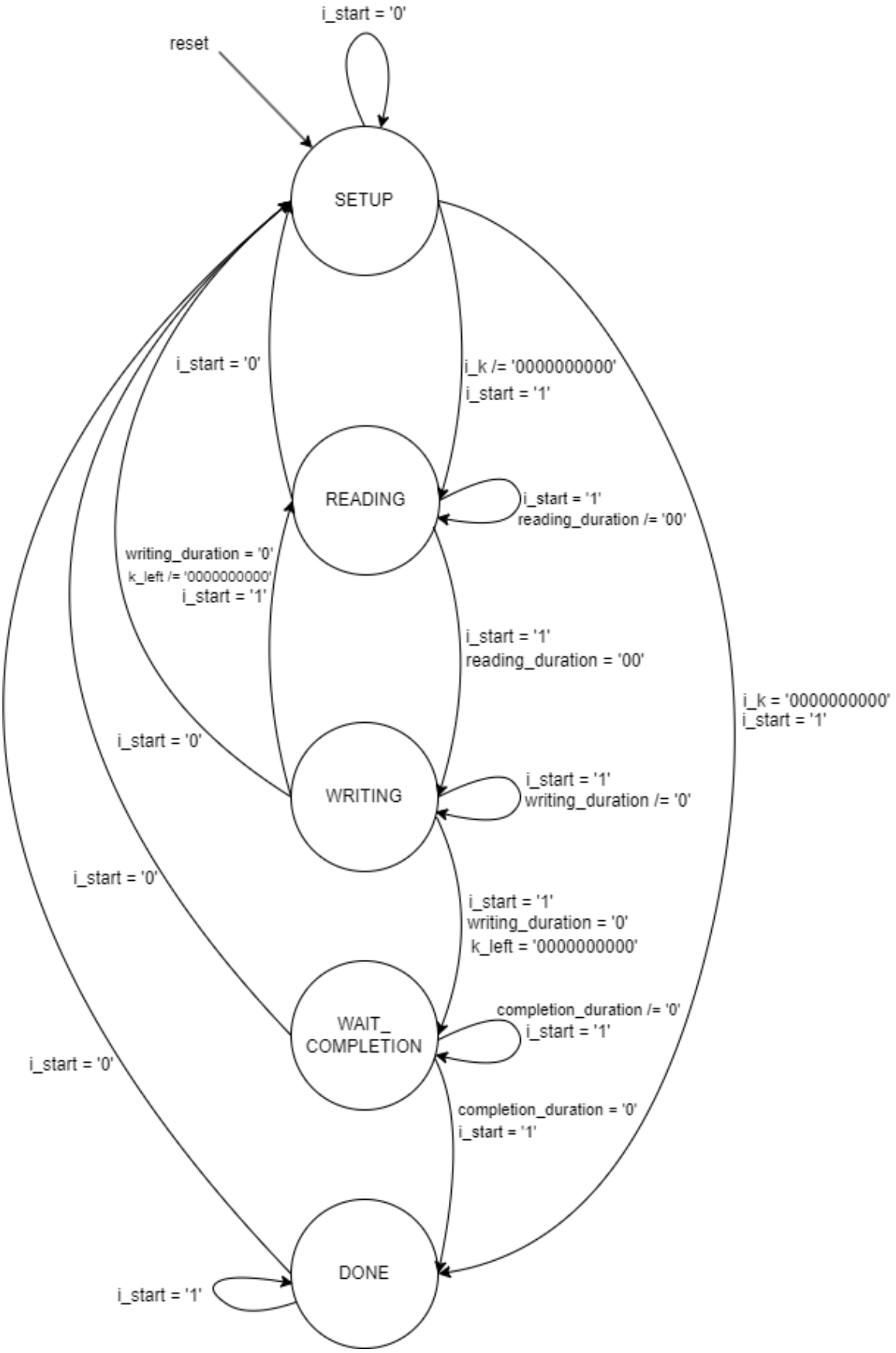
Sottomodulo che agisce da macchina a stati finiti. Ha lo scopo di coordinare il funzionamento dei vari sottomoduli e notificare la fine dell'elaborazione alzando il segnale DONE ad 1. In base a i segnali START, RESET e i_k e quanti cicli di clock sono trascorsi dall'ultima transizione, si può spostare in 5 stati diversi: SETUP, READING, WRITING, WAIT_COMPLETION e DONE.
Lo stato SETUP è quello iniziale, nel quale la macchina attende che START valga 1 per iniziare l'elaborazione. È anche lo stato a cui la macchina ritorna e rimane (a prescindere da qualunque stato in cui essa sia o a cui debba andare) ogni volta che RESET è attivo.
Dallo stato SETUP, si passa generalmente allo stato READING; l'unico caso in cui ciò non accade è quando i_k vale 0, in tal caso la fase di elaborazione non è necessaria, quindi si passa direttamente allo stato DONE.
Gli stati che si presentano durante l'elaborazione sono quelli di READING, WRITING e WAIT_COMPLETION.
Lo stato READING abilita il memory_reader e disabilita il memory_writer, mentre lo stato WRITING fa il contrario; questi due stati vengono utilizzati per coordinare il funzionamento dei due sottomoduli che controllano. Ogni volta che si passa allo stato READING, il segnale o_update_reading viene settato ad 1, per poi essere settato a 0 nel ciclo di clock successivo.
Per ognuno dei K valori della sequenza, è necessario l'utilizzo del memory_reader seguito da quello del memory_writer, quindi dallo stato READING si passa a quello

di WRITING. Il numero di transizioni ancora da svolgere verso READING, partendo da stati diversi da READING, è memorizzato nel segnale interno k_left. Il segnale interno reading_duration tiene traccia dei cicli di clock rimanenti da trascorrere nello stato READING, che ha una durata totale di 3 cicli di clock.
Dallo stato WRITING si può passare a due possibili stati: se rimangono valori della sequenza da leggere (cioè se k_left è maggiore di zero) si ritorna allo stato READING, altrimenti si passa allo stato WAIT_COMPLETION. Il segnale interno writing_duration tiene traccia dei cicli di clock rimanenti da trascorrere nello stato WRITING, che ha una durata totale di 2 cicli di clock.
Lo stato WAIT_COMPLETION si presenta quindi solo immediatamente prima della fine dell'elaborazione: esso ha lo scopo di attendere l'esecuzione dell'ultima operazione di scrittura, dopo la quale si passa allo stato DONE. Il segnale interno completion_duration tiene traccia dei cicli di clock rimanenti da trascorrere nello stato WAIT_COMPLETION, che ha una durata totale di 2 cicli di clock.
Lo stato DONE setta il segnale DONE ad 1 per due cicli di clock (cioè fino a quando rileva che START vale 1), per poi spostarsi nuovamente allo stato SETUP. Inoltre, mediante i segnali o_clear_reading e o_clear_writing, riporta rispettivamente memory_reader e memory_writer alle loro condizioni iniziali (cioè quelle prima dell'inizio dell'elaborazione).
Nel complesso la macchina si comporta come una macchina di Moore, eccetto per il segnale o_update_reading, che è attivo solo durante il primo dei tre cicli di clock passati nello stato READING (quindi è un'uscita che non dipende solo dallo stato corrente).
Un possibile approccio alternativo consisteva nell'inserire uno stato aggiuntivo che precedesse sempre READING, il quale avrebbe impostato o_update_reading a 1, lasciando poi allo stato READING il compito di azzerarlo: in tal modo la macchina sarebbe stata completamente di Moore. Si è preferito raggrupparli in un unico stato poiché, eccetto per o_update_reading, i due stati sarebbero stati identici.

Tabella degli stati e le loro rispettive uscite (eccetto o_update_reading)

Stato	Uscite				
	o_en_reading	o_en_writing	o_done	o_clear_reading	o_clear_writing
SETUP	0	0	0	0	0
READING	1	0	0	0	0
WRITING	0	1	0	0	0
WAIT_COMPLETION	0	0	0	0	0
DONE	0	0	1	1	1

Rappresentazione della macchina a stati (ogni volta che RESET sale a 1 si passa allo stato SETUP, che è anche lo stato iniziale)



Report sperimentali

Sintesi

È stata utilizzata una FPGA Artix-7 xc7a200tfbg484-1 per la sintesi del componente. Vengono utilizzati solamente flip-flop al fine di evitare complicazioni derivanti dall'eventuale uso di latches, come si può notare dal seguente utilization report:

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	215	0	134600	0.16
LUT as Logic	215	0	134600	0.16
LUT as Memory	0	0	46200	0.00
Slice Registers	119	0	269200	0.04
Register as Flip Flop	119	0	269200	0.04
Register as Latch	0	0	269200	0.00
F7 Muxes	0	0	67300	0.00
F8 Muxes	0	0	33650	0.00

Inoltre il componente opera entro i limiti imposti dal clock (avente periodo di 20 ns), infatti ha uno slack positivo di circa 16 ns, come si può notare dal seguente timing report:

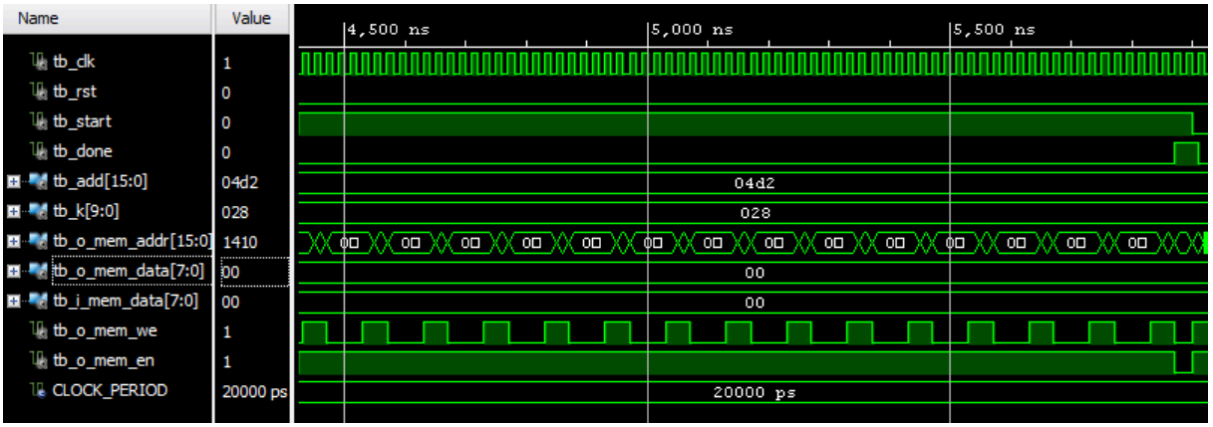
```
Timing Report

Slack (MET) :             16.111ns  (required time - arrival time)
  Source:                 memory_reader_1/curr_address_reg[5]/C
                            (rising edge-triggered cell FDCE clocked by clock  {rise@0.000ns fall@10.000ns period=20.000ns})
  Destination:            memory_reader_1/curr_address_reg[0]/CE
                            (rising edge-triggered cell FDCE clocked by clock  {rise@0.000ns fall@10.000ns period=20.000ns})
  Path Group:             clock
  Path Type:              Setup (Max at Slow Process Corner)
  Requirement:            20.000ns  (clock rise@20.000ns - clock rise@0.000ns)
  Data Path Delay:        3.507ns  (logic 1.811ns (51.640%)  route 1.696ns (48.360%))
  Logic Levels:           4  (CARRY4=2 LUT3=1 LUT5=1)
  Clock Path Skew:        -0.145ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD):    2.100ns = ( 22.100 - 20.000 )
    Source Clock Delay      (SCD):    2.424ns
    Clock Pessimism Removal (CPR):    0.178ns
  Clock Uncertainty:      0.035ns  ((TSJ^2 + TIJ^2)^1/2 + DJ) / 2 + PE
    Total System Jitter     (TSJ):    0.071ns
    Total Input Jitter      (TIJ):    0.000ns
    Discrete Jitter          (DJ):    0.000ns
    Phase Error              (PE):    0.000ns
```

Simulazioni

Oltre al testbench e gli esempi forniti, sono stati ideati altri testbench per verificare il corretto funzionamento del componente, con una particolare attenzione ai casi limite. Ogni testbench controlla l'intera memoria per verificare che le modifiche siano state apportate solo alle celle previste.

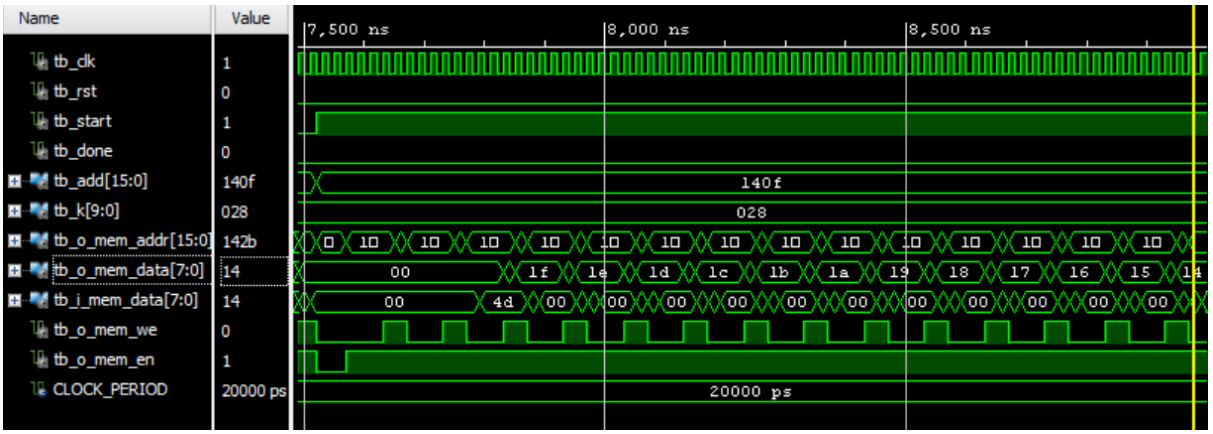
- **Intera sequenza di zero**

Verifica che quando tutti i valori della sequenza sono pari a zero, il componente si comporti in maniera corretta.



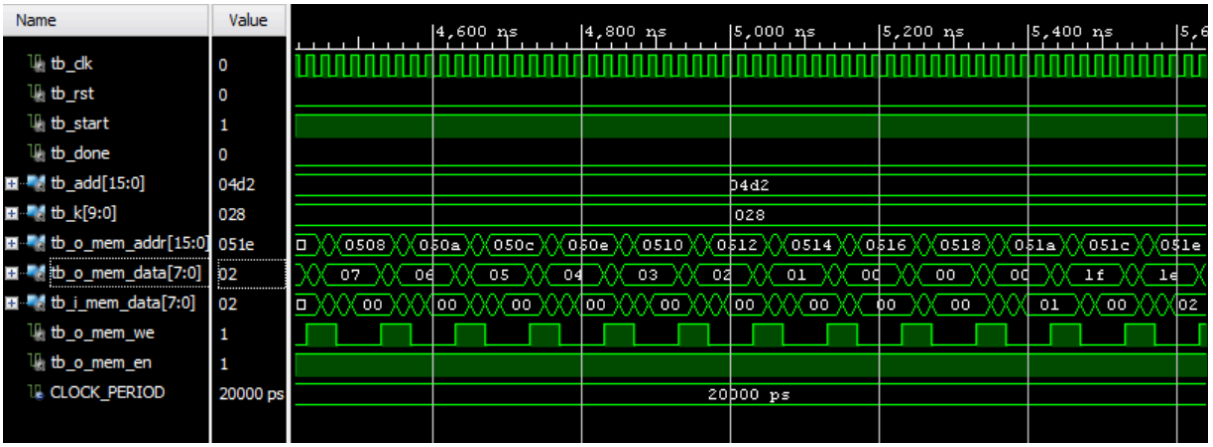
- **Sequenza di valori comincia con zero**

Verifica che quando i valori iniziali della sequenza sono pari a zero, il componente si comporti in maniera corretta.



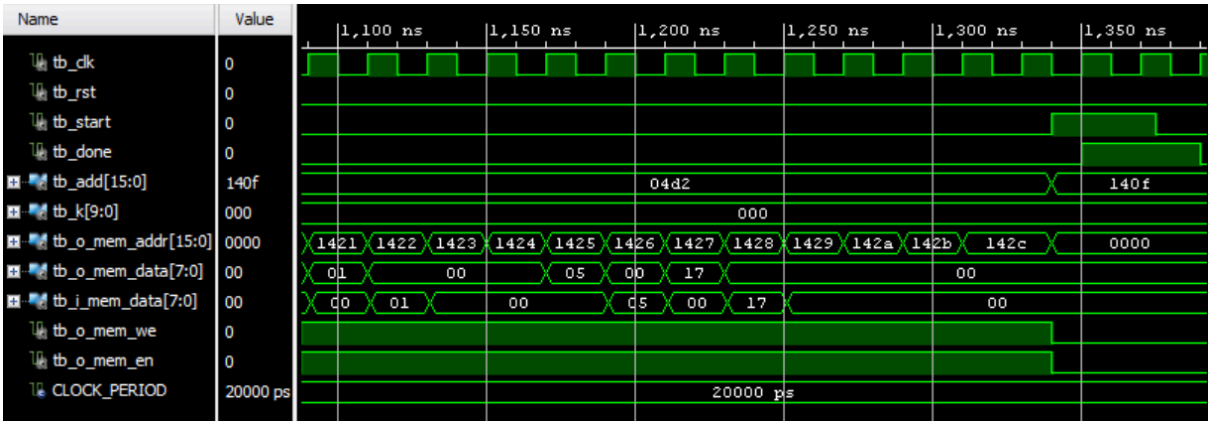
- **Credibilità sempre maggiore di 0**

Verifica che la credibilità non scenda sotto lo zero.



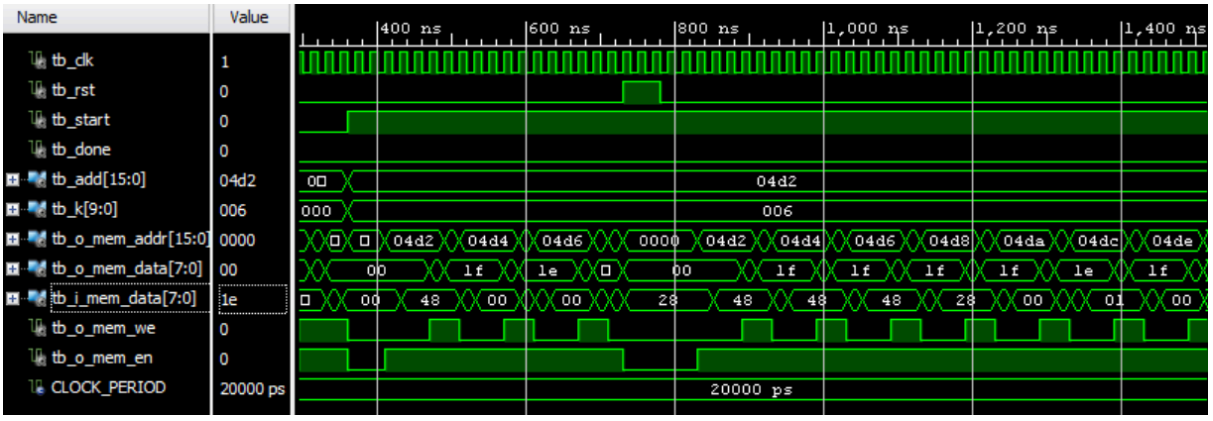
- **i_k = 0**

Verifica che, se i_k = 0, il componente non interagisca con la memoria ed alzi DONE ad 1 non appena rileva che START vale 1.



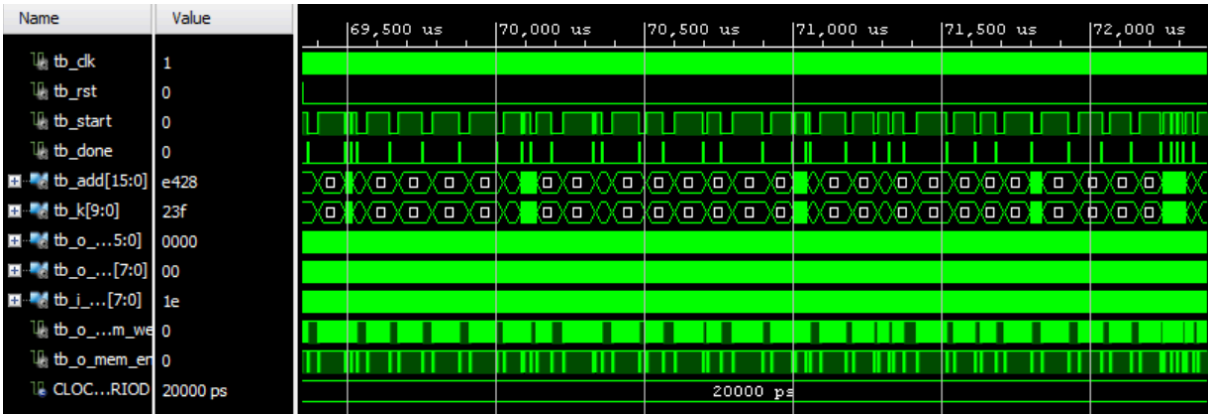
- **Reset prima della fine del processo**

Verifica il corretto funzionamento del componente dopo l'attivazione del segnale RESET (anche molteplici volte) durante il processo.



- **1000 sequenze di seguito**

Verifica il corretto funzionamento durante 1000 sequenze testate di seguito. Fra queste sono inclusi alcuni nuovi possibili casi limite quali l'utilizzo dell'indirizzo della prima cella di memoria come indirizzo iniziale della sequenza, i_k = 1, e una sequenza i cui numeri sono tutti diversi da zero e fra di loro.
Il reset non viene attivato perché si considera già testato sufficientemente nel suo apposito testbench.



Conclusione

Il componente progettato supera tutti i testbench considerati e non genera latch indesiderati. Esso è composto interamente da sottomoduli utili ed organizzati, infatti ciascuno svolge una propria specifica funzione ed è coordinato con gli altri. Come analizzato, il componente opera correttamente, rispettando tutte le richieste del progetto.