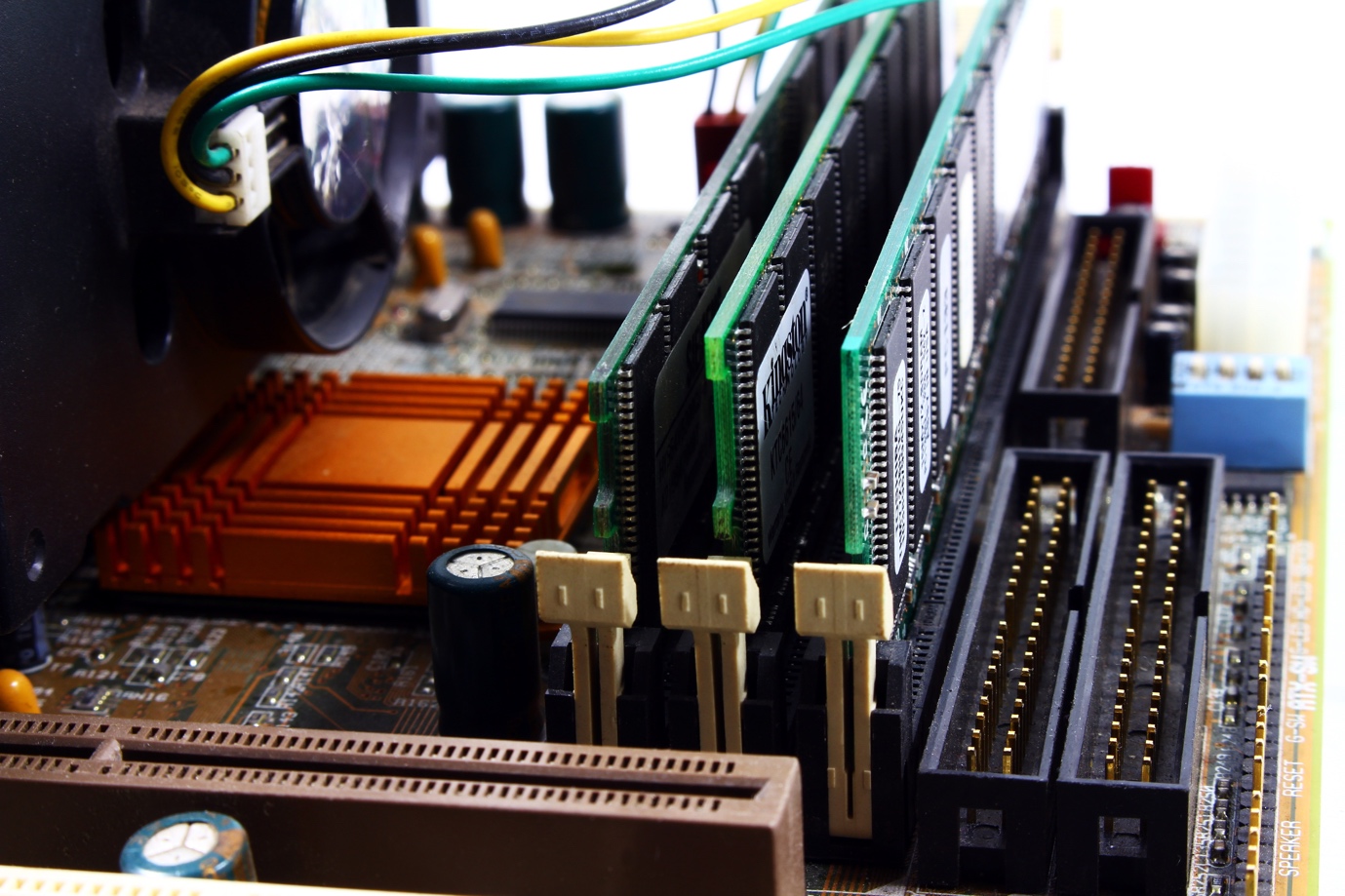
PROGETTO RETI LOGICHE

PROF. WILLIAM FORNACIARI – ANNO 2022/2023

Simone Calzolaro (Codice Persona 10710180)

Mirko Calvi (Codice Persona 10724100)



1 Introduzione

* 1. Scopo del progetto

Lo scopo del progetto è quello di sviluppare ed implementare un sistema (componente hardware), descritto in linguaggio VHDL , che si interfacci con una memoria e ne estragga i valori contenuti agli indirizzi forniti in ingresso per poi stamparli in uscita su bus dedicati.

* 1. Interfaccia del componente

Text, table

Description automatically generated

In particolare:

* i\_clk: segnale di CLOCK in ingresso generato dal TestBench;
* i\_rst: segnale di RESET che inizializza/reinizializza la macchina;
* i\_start: segnale di START generato dal TestBench;
* i\_w: segnale di W in ingresso generato dal TestBench;
* o\_z0, o\_z1, o\_z2, o\_z3: canali di uscita del componente;
* o\_done: segnale di uscita che indica la fine dell’elaborazione;
* o\_mem\_addr: segnale di uscita che manda l’indirizzo alla memoria;
* i\_mem\_data: segnale che arriva dalla memoria a seguito di una richiesta di lettura;
* o\_mem\_en: segnale di ENABLE da comunicare alla memoria per poter comunicare (read e/o write);
* o\_mem\_we: segnale di WRITE ENABLE da dover mandare alla memoria per poterci scrivere (=1) o leggere (=0).
  1. Specifica generale

Diagram, schematic

Description automatically generated

Preliminarmente il sistema rimane in attesa di un segnale di RESET necessario ad inizializzare il sistema prima di qualsiasi altra operazione, successivamente esso attende l’arrivo di un segnale di START per poter cominciare l’elaborazione. A seguito dell’arrivo del segnale di START il sistema comincia a leggere sequenzialmente i segnali provenienti da i\_w (1 bit) che saranno interpretati in questo modo:

1. I primi 2 bit letti da i\_w a seguito del segnale di START saranno impiegati per indicare il canale di uscita o\_zi (con i ∈ {0,1,2,3}) sul quale si vuole che il sistema stampi il valore letto da memoria;
2. I bit seguenti saranno impiegati per il calcolo dell’indirizzo di memoria dal quale si vuole prelevare il valore.

Successivamente un segnale i\_start=0 indicherà la fine della lettura dell’indirizzo e si potrà passare alla fase di lettura della memoria.

Per permettere alla memoria di essere letta il sistema dovrà settare adeguatamente i due segnali ENABLE e WRITE ENABLE, rispettivamente ad 1 e 0. Così facendo la memoria è abilitata a leggere l’indirizzo contenuto in o\_mem\_addr del sistema per poi restituirgli il rispettivo valore sul canale i\_mem\_data.

A questo punto dell’elaborazione il sistema è pronto per aggiornare il valore del canale di uscita indicato all’inizio con il nuovo valore letto da memoria, lasciando invariati i valori caricati precedentemente sui restanti canali, e stampare in uscita i valori dei canali e il valore di o\_done=1.

NOTE:

1. o\_done = 1 per un solo ciclo di clock;
2. Il sistema stampa ‘00000000’ sempre quando o\_done = 0;
3. Il sistema stampa i valori caricati da memoria se e solo se o\_done=1.

2 Architettura

L’architettura del sistema non prevede l’utilizzo di componenti per lo svolgimento di funzioni ad-hoc, le varie funzionalità sono interamente ricoperte grazie all’utilizzo di una macchina a stati finiti e alcuni processi specifici.

Il sistema si avvale di alcuni segnali ausiliari non compresi nell’interfaccia:

SEGNALI FSM

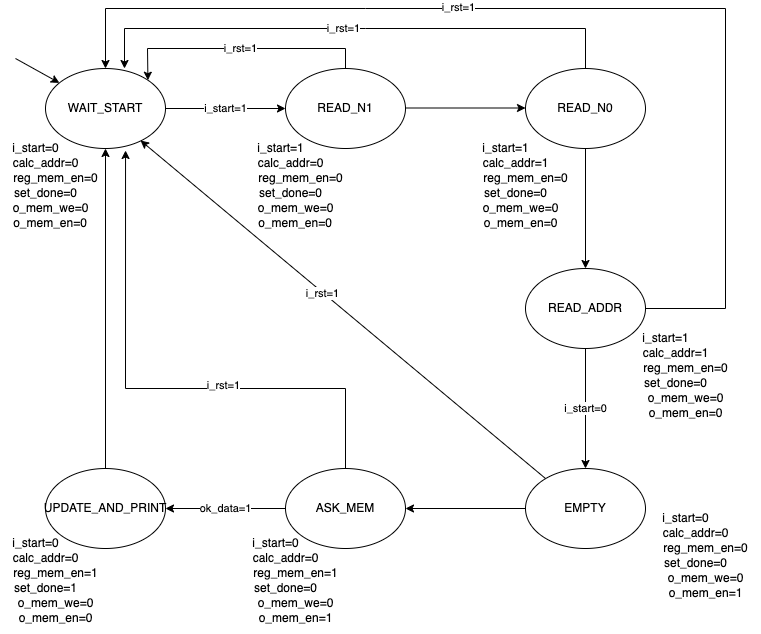
* ok\_data: segnale in input alla FSM il quale indica la fine della lettura da memoria (=1) da parte del sistema. Suddetto segnale viene settato dal processo che si occupa di lettura della memoria (get\_data\_mem);
* set\_done: segnale in output alla FSM il quale segnala al processo incaricato (update\_and\_print\_Z) che lettura e caricamento del dato da memoria sono terminati con successo, quindi può procedere a settare o\_done=1 e stampare i valori in uscita nei rispettivi canali dedicati;
* calc\_addr: segnale in output alla FSM il quale segnala al processo incaricato (mem\_addr\_calc) che la lettura del bus di uscita è terminata e può dunque partire con la lettura dell’indirizzo di memoria;
* reg\_mem\_en: segnale in output alla FSM il quale segnala al processo incaricato (get\_data\_mem) che la lettura della memoria è abilitata;
* PresState: segnale che indica alla macchina lo stato in cui attualmente si trova l’elaborazione.

SEGNALI NON FSM

* reg\_mem\_addr: registro incaricato di salvare l’indirizzo di memoria ricevuto in input per poi essere copiato nel bus dedicato o\_mem\_addr;
* reg\_data\_mem: registro incaricato di salvare il dato letto dalla memoria per poi essere copiato nel rispettivo bus;
* reg\_n: registro incaricato di salvare l’indirizzo del bus di uscita sul quale si vuole stampare il dato letto da memoria. Intuitivamente, vi vengono salvati i primi due valori di i\_w a seguito del segnale di START positivo;
* reg\_z0, reg\_z1, reg\_z2, reg\_z3: registri incaricati di salvare i valori dei rispettivi bus o\_zi (i∈ {0,1,2,3}) a modo da non perdere l’informazione quando o\_done=0 che impone a questi ultimi di essere settati a ‘00000000’;

2.1 Macchina a stati finiti

Il sistema ruota sostanzialmente attorno alla sua macchina a stati finiti, la quale si occupa di spostare l’elaborazione su diversi stati logici del processo complessivo e di assegnare segnali ausiliari ai vari processi per permettere l’esecuzione di specifiche funzionalità.



Come si evince dalla figura illustrata la macchina è composta da 7 stati: WAIT\_START, READ\_N1, READ\_N0, READ\_ADDR, EMPTY, ASK\_MEM, UPDATE\_AND\_PRINT.

2.1.1 WAIT\_START

WAIT\_START è il primo stato nel quale la macchina si trova a seguito dell’inizializzazione preliminare dovuta al primo RESET. In questo stato la macchina mantiene tutti i segnali ausiliari a zero rimanendo in attesa del segnale di START al seguito del quale la macchina sposterà l’elaborazione allo stato READ\_N1.

2.1.2 READ\_N1

READ\_N1 si occupa, come suggerisce anche il nome, di assegnare al registro reg\_n (descritto nei SEGNALI NON FSM) il primo bit i\_w letto da input. Per quanto riguarda i segnali ausiliari questi rimangono immutati rispetto allo stato precedente.

2.1.3 READ\_N0

Come lo stato precedente, READ\_N0 si occupa di assegnare a reg\_n il secondo bit i\_w letto da input ed avere così terminato il salvataggio su reg\_n dell’indirizzo del bus di uscita sul quale si vuole stampare il valore letto da memoria. Per quanto riguardo i segnali ausiliari essi rimangono immutati rispetto allo stato precedente ad eccezione di calc\_addr il quale viene settato ad 1 per segnalare al processo mem\_addr\_calc che è abilitato a leggere l’indirizzo di memoria.

2.1.4 READ\_ADDR

Questo stato è responsabile della lettura dell’indirizzo di memoria da input, la quale è effettuata dal processo mem\_addr\_calc. Lo stato mantiene il valore calc\_addr=1 per tutta la durata della lettura dell’indirizzo. La fine della lettura è indicata dal segnale i\_start=0 il quale forza il cambiamento di stato.

2.1.5 EMPTY

Questo stato di elaborazione è stato progettato con l’obbiettivo di settare la macchina in vista della fase di lettura della memoria, esso infatti setta calc\_addr=0 indicando appunto la fine della lettura dell’indirizzo, ed o\_mem\_en=1 il quale abilita la memoria alla lettura. Il secondo e non meno importante obbiettivo è quello di “ritardare l’andamento della macchina” per aspettare la memoria. Questa scelta è dovuta al fatto che le varie simulazioni effettuate suggerivano che la macchina da noi progettata anticipava la lettura della memoria di 1 ciclo di clock, rendendo così necessaria l’aggiunta di un nuovo stato atto a ritardare la vecchia esecuzione di 1 ciclo di clock , da cui il simbolico nome “EMPTY”.

2.1.6 ASK\_MEM

ASK\_MEM è impiegato nella gestione della lettura del dato da memoria, infatti esso mantiene il segnale o\_mem\_en=1 settando inoltre reg\_mem\_en=1 il quale provvede a segnalare al processo get\_data\_mem l’abilitazione a leggere il valore dalla memoria. Sempre da quest’ultimo processo lo stato attende l’arrivo del segnale ok\_data=1 il quale indica che la lettura è andata a buon fine e dunque può cambiare stato di elaborazione.

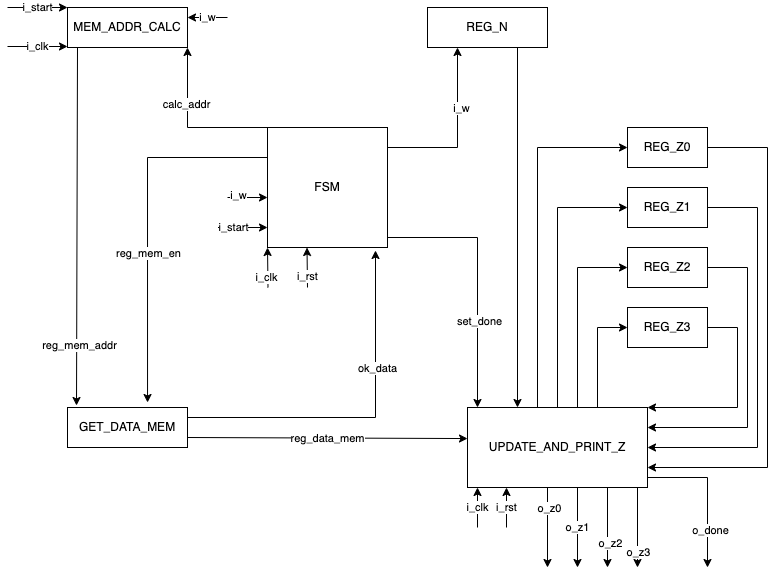
2.1.5 UPDATE\_AND\_PRINT

Come suggerisce il nome quest’ultimo stato si occupa di gestire l’aggiornamento devi valori sui bus di uscita e di stamparne i valori grazie al settaggio di o\_done=1. Il tutto è possibile in quanto lo stato, settando il segnale set\_done=1, autorizza il processo update\_and\_print\_Z ad eseguire stampa e aggiornamento. Il tutto avviene in 1 ciclo di clock, dopodiché la macchina torna allo stato iniziale WAIT\_START.

2.2 Processi del sistema

La macchina a stati finiti si occupa di gestire lo stato di elaborazione del sistema e di settare il valore di alcuni segnali ausiliari, ma come già si è visto nella sezione precedente, le funzionalità principali sono svolte da processi separati.

Questi processi sono indipendenti tra loro e la loro gestione e coordinazione è gestita dalla FSM.



I processi coinvolti nel sistema sono 3:

* MEM\_ADDR\_CALC: processo incaricato della lettura dell’indirizzo di memoria da input. Esso è sincrono al clock e le sue funzionalità vengono abilitate dalla FSM nello stato READ\_N0 il quale setta il segnale calc\_addr=1 permettendo l’inizio del salvataggio dei bit di indirizzo nel registro reg\_mem\_addr. Il segnale calc\_addr viene resettato in ASK\_MEM, e per tutto l’intervallo di tempo per cui si manterrà nullo il processo forzerà reg\_mem\_addr a ‘0000000000000000’.
* GET\_DATA\_MEM: processo incaricato della lettura della memoria. Esso è un processo asincrono, infatti l’inizio della lettura è dettata del settaggio del segnale reg\_mem\_en=1 da parte dello stato EMPTY della FSM il quale abilita il processo a leggere la memoria. In particolare il processo si occupa di salvare il segnale i\_mem\_data proveniente dalla memoria nel registro ausiliario reg\_data\_mem.

Si noti che il processo non è completamente passivo alla FSM ma assume un ruolo attivo per le sorti della macchina a stati stessa, in quanto si occupa di settare il segnale ok\_data=1, il quale segnala la corretta terminazione della lettura da memoria e permette alla macchina di cambiare stato di elaborazione quando essa si trova in ASK\_MEM.

* UPDATE\_AND\_PRINT\_Z: processo incaricato di aggiornare e stampare i dati sui canali di uscita dedicati. Queste funzionalità sono svolte in maniera sincrona rispetto al CLOCK. Il processo è incaricato di mantenere i valori sui registri reg\_zi (i∈{0,1,2,3}) e di stampare sui canali di uscita o\_zi un vettore di zeri quando il segnale set\_done=0, il quale indica a sua volta di stampare o\_done=0. Il vero ruolo del processo viene fuori quando la macchina a stati setta il segnale set\_done=1 grazie allo stato UPDATE\_AND\_PRINT, il quale forzerà il processo ad eseguire:

l’aggiornamento del canale scelto, la stampa in output dei canali di uscita o\_zi , la stampa in uscita del canale o\_done settato a 1.

Il processo inoltre può gestire il caso di RESET ricevendo un segnale completamente asincrono da input e forzando i canali di uscita e i registri dedicati a zero.

Il processo viene eseguito in un ciclo di clock come governato dalla FSM.

1. Risultati sperimentali
   1. Sintesi

Immagine che contiene testo, ricevuta, schermata

Descrizione generata automaticamente

Immagine che contiene tavolo

Descrizione generata automaticamente

3.2 Simulazioni

Nel seguente paragrafo verranno spiegate le simulazioni di diversi casi limite

3.2.1 Casi particolari di reset

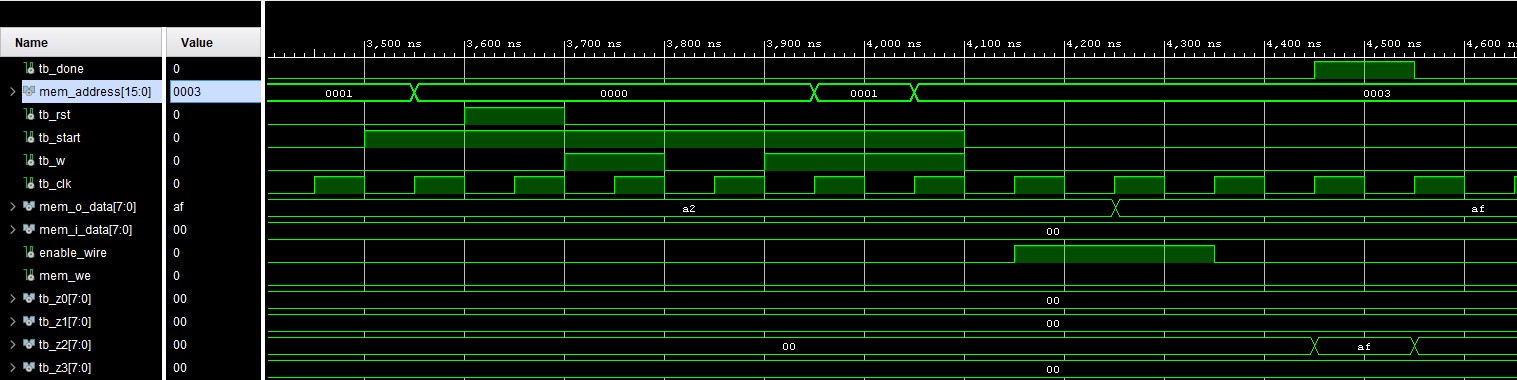
Abbiamo individuato 8 diversi casi limite in può essere richiesto il reset della rete :

* Caso 1: reset subito dopo il reset di inizializzazione della rete
* Caso 2: reset durante la lettura del primo bit per l’assegnamento del canale z per il futuro output
* Caso 3: reset durante la lettura del secondo bit per l’assegnamento del canale z per il futuro output
* Caso 4: reset durante la lettura dell’indirizzo di memoria
* Caso 5: reset immediatamente dopo la lettura dell’indirizzo di memoria
* Caso 6: reset durante la lettura del dato dalla memoria
* Caso 7: reset immediatamente dopo la lettura del dato dalla memoria
* Caso 8: reset al termine dell’output dei dati sui registri Z, tale caso non verrà analizzato in quanto analogo al reset di inizializzazione e al Caso 1:

caso_0_1_2
 CASI 1 e 2

Come si può notare dall’immagine i caso 1 lascia invariato il sistema mentre il caso 2 porterà il sistema ad ignorare il primo bit. Reg\_n avrà quindi valore “11”, infatti l’uscita che è stata modificata quando o\_done=’1’ è la quarta (tb\_z3).

CASO 3



Il reset avviene durante la lettura del secondo bit del canale. Dopo il reset infatti reg\_mem=”10” quindi l’output avverà su z2.

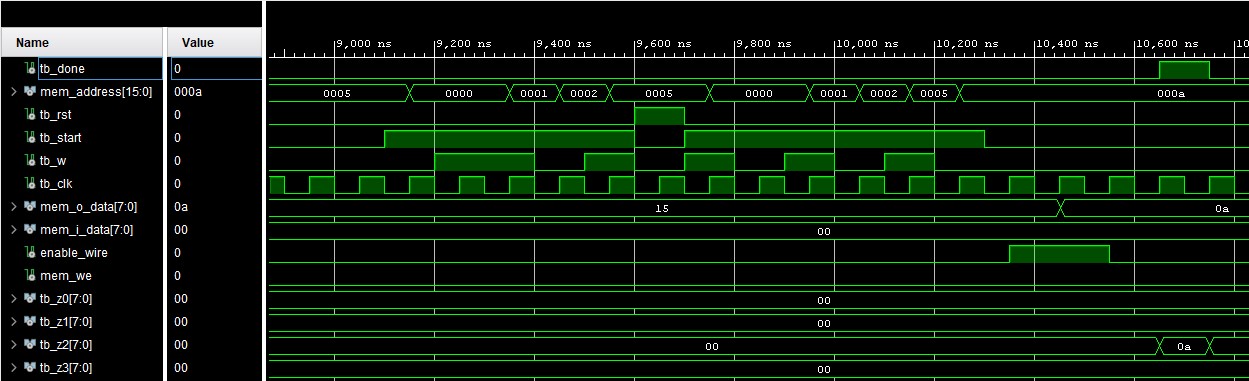
CASO 4

Immagine che contiene testo, elettronica

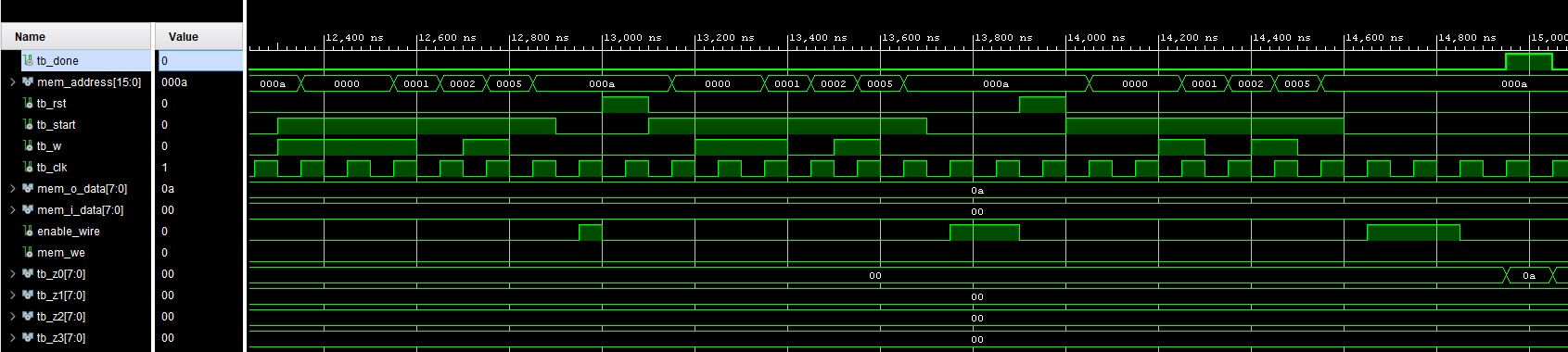
Descrizione generata automaticamente

Questo caso particolare è volto a dimostrare il corretto funzionamento del sistema a seguito di un reset in un momento qualsiasi della lettura dell’indirizzo di memoria. Come mostra la figura durante il reset mem\_address viene posto a “00..00”, avvenuta la lettura di indirizzo e canale di output viene stampato un valore su tb\_z1, da notare che il valore del caso 3 su tb\_z2 non è più presente.

CASO 5



A seguito del passaggio a ‘0’ di tb\_start viene chiamato un reset che riporta il sistema allo stato iniziale e annulla tutti i registri tb\_z.

 CASO 6 e 7

Il caso 6 è degno di nota in quanto è ben visibile dal grafico che quando viene chiamato il reset a 13,00 ns il valore enable\_wire passa subito da ‘1’ a ‘0’ interrompendo all’istante la lettura con la memoria.

Il reset del caso 7 invece attende che la lettura dalla memoria venga completata per poi chiamare il reset e iniziare subito dopo un nuovo ciclo di lettura.

3.2.2 Casi limite di lettura dell’indirizzo di memoria

Sono stati individuati 2 casi limite di lettura dell’indirizzo di memoria:

* Indirizzo di memoria lungo 0 bit
* Indirizzo di memoria lungo 16 bit

Immagine che contiene testo, elettronica

Descrizione generata automaticamente

Figura , 0 bit length

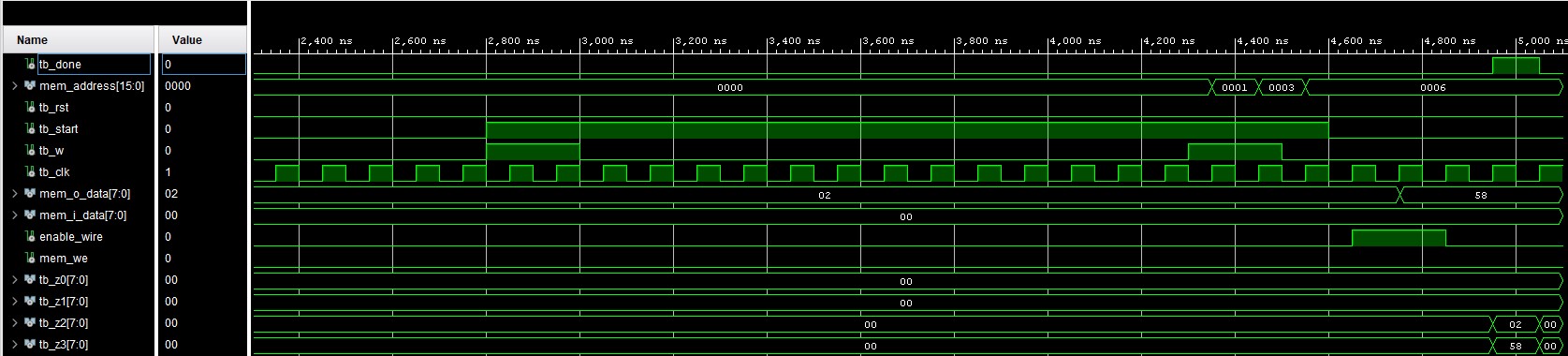


Figura , 16 bit length

In figura 1 e figura 2 viene dimostrato il corretto funzionamento della rete nei due casi limite, da notare che nel secondo caso viene anche dimostrata la persistenza dei segnali sui canali Z in quanto su tb\_z2 viene riportato lo stesso valore acquisito durante la lettura di figura 1.