Prova Finale Reti Logiche

Prova Finale Reti Logiche

Prof Gianluca Palermo
Emanuele Pocelli cp: 10726303

Indice

1 Introduzione

Lo scopo del progetto è implementare un componente hardware descritto in vhdl che legga delle sequenze di bit, prenda dalla RAM il corretto valore associato a ciascuna sequenza e infine mostri tale valore. In FIGURA 1, sono riportate due sequenze di esempio. In particolare, nella prima viene modificato il valore del canale di uscita z0, mentre nella seconda quello di z2. Da notare che nella seconda sequenza si tiene ancora traccia del valore aggiornato di z0.

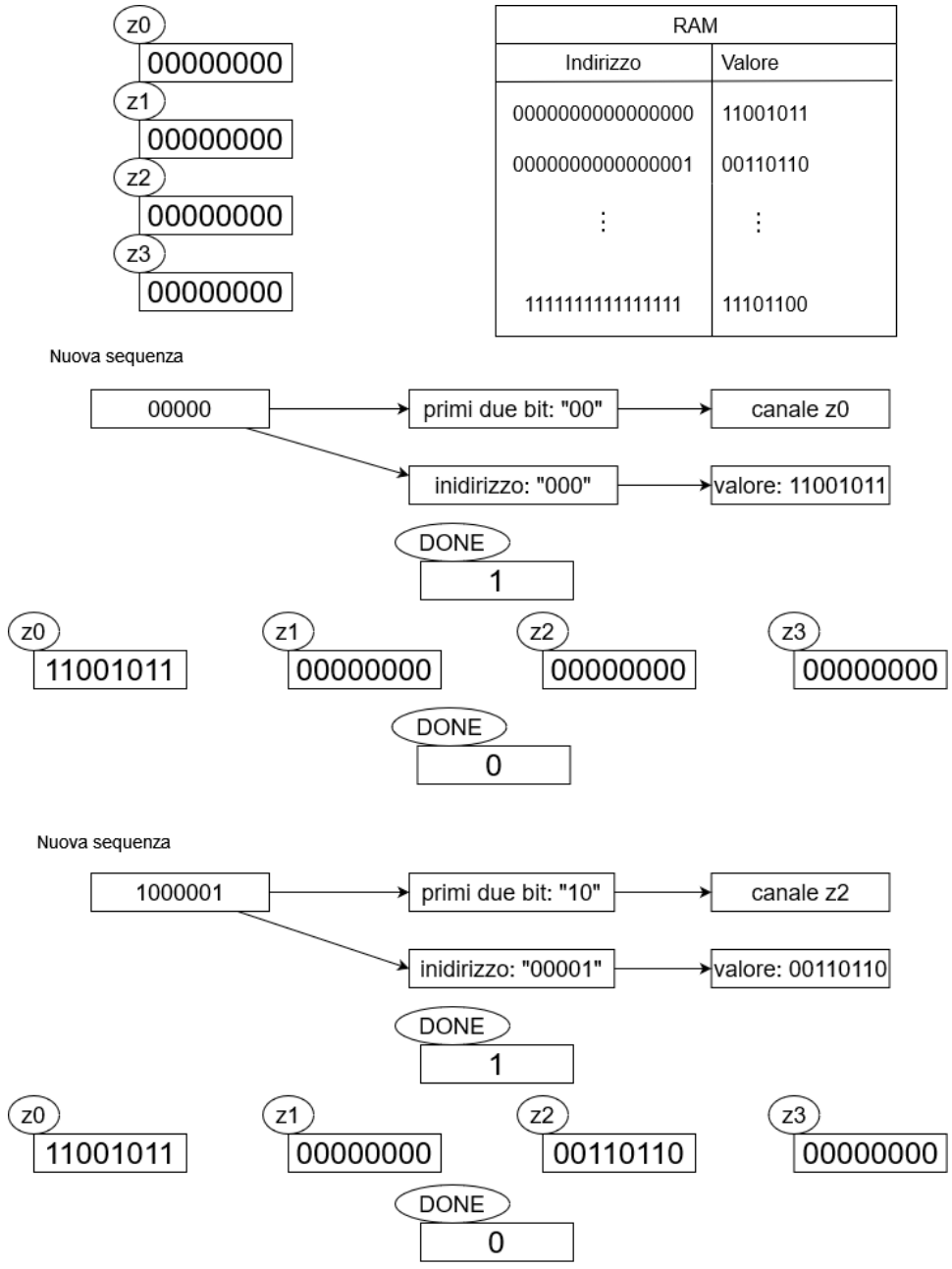


Figura 1

Dettagli

L'interfaccia del componente vhdl è la seguente:

Segnali di input:
`i_clk` - clock
`i_rst` - segnale di reset
`i_start` - segnale di inizio di una sequenza: quando va a 1 inizia una nuova sequenza, quando torna a 0 finisce
`i_w` - valore da salvare a ogni ciclo di clock in modo da formare un vettore (che corrisponde alla sequenza)

Segnali di output:
`o_z0`, `o_z1`, `o_z2`, `o_z3` - canali di uscita sul quale verranno mostrati i valori
`o_done` - segnale che si alza quando vengono mostrati i valori di uscita

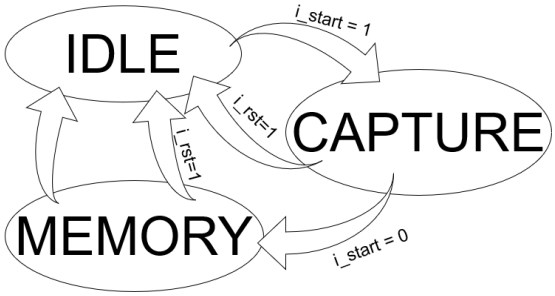
Segnali relativi alla memoria:
`o_mem_addr` - valore per comunicare alla memoria a quale indirizzo si vuole accedere
`i_mem_data` - valore relativo all'indirizzo a seguito di una richiesta
`o_mem_we` - segnale di write enable: se è 0 si vuole accedere in lettura, se è 1 in scrittura. Il componente da me descritto accede solo in lettura, quindi questo segnale sarà a 0 per tutta l'esecuzione
`o_mem_en` - segnale per comunicare con la memoria

Quando START è alto, ha inizio una nuova sequenza: a ogni ciclo di clock viene memorizzato il valore di W fino a che START non torna basso. Dopodichè, da questa sequenza memorizzata si identificano l'intestazione (quale canale interessa la sequenza appena letta) e l'indirizzo di memoria da cui prendere il valore. Una volta preso il valore, il segnale DONE sale e contemporaneamente l'uscita corrispondente mostra il valore appena salvato. Successive sequenze possono aggiornare valori su altri canali e in corrispondenza del segnale DONE alto, ogni canale mostra l'ultimo valore memorizzato a esso associato. Il segnale DONE rimane alto quando vengono mostrati i valori nelle uscite per un ciclo di clock, mentre è basso per il resto dell'esecuzione.
La lettura di W deve avvenire durante il fronte di salita del CLOCK. La lunghezza della sequenza può variare da 2 a 18 bit: i primi due bit sono di intestazione (si riferiscono al canale di uscita), mentre gli altri si riferiscono all'indirizzo di memoria. Tale indirizzo viene esteso a 16 bit.
Infine, il segnale di reset, necessario tra l'altro prima della sequenza iniziale per garantire il corretto funzionamento del programma, pone tutti i canali di uscita al valore di default "00000000" e interrompe l'eventuale sequenza in corso.

2 Architettura

Il programma è composto fondamentalmente da 2 processi: il primo gestisce il reset e l'aggiornamento delle uscite, il secondo descrive una macchina a stati. In quest'ultimo processo, vengono aggiornati dei segnali "next", uno per ogni uscita, corrispondenti al valore che l'uscita relativa dovrà avere al successivo ciclo di clock.

2.1 Stati della macchina



2.1.1 IDLE

La macchina va nello stato IDLE quando viene letto il reset o al termine di una sequenza. Si rimane in questo stato fintanto che START è basso. Quando viene letto START=1, si memorizza il primo valore di W e ci si sposta nello stato CAPTURE.

2.1.2 CAPTURE

Stato in cui vengono memorizzati i valori di W. Quando viene letto START=0, gli n bit letti relativi all'indirizzo vengono shiftati a destra di (16-n) posizioni e viene posto a 1 il segnale `MEM_EN` relativo alla RAM.

2.1.3 MEMORY

Si attende la risposta della RAM e si memorizza il valore da mostrare. Infine, vengono mostrati i valori memorizzati nelle uscite Z0, Z1, Z2, Z3 e si torna nello stato IDLE.

Ovviamente, un segnale di reset alto fa tornare la macchina nello stato IDLE indipendentemente dallo stato corrente e la sequenza letta fino a quel momento viene ignorata.

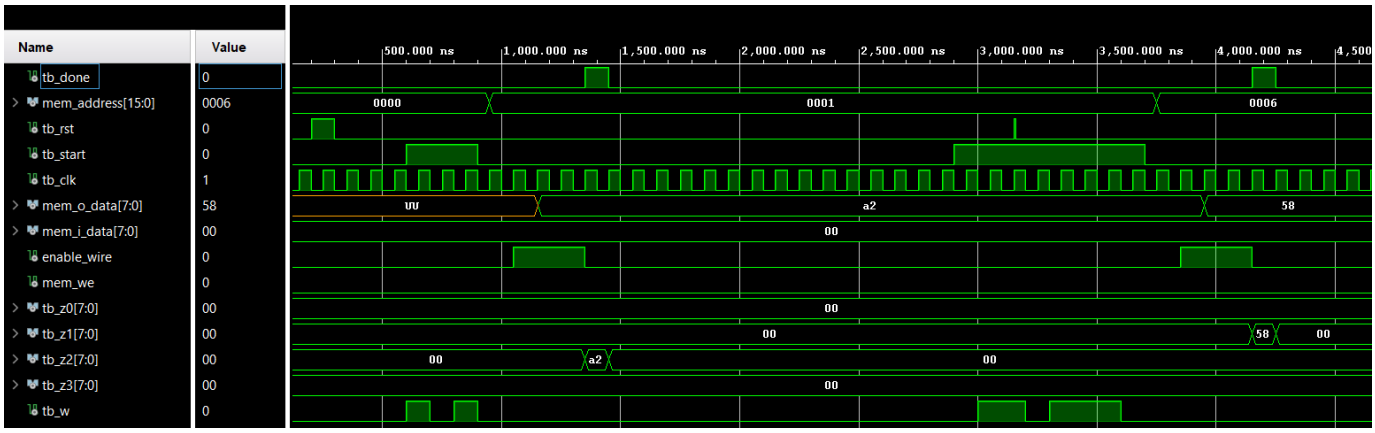
3 Funzionamento

Le uscite vengono gestite nel primo processo: a ogni ciclo viene assegnato a ciascuna di esse il valore `next` se non è presente il reset. I segnali `next`, invece, vengono modificati nel secondo processo. Sempre nel secondo processo, vengono memorizzate le uscite relative a sequenze precedenti attraverso delle variabili.
Si suppone che all'inizio di ogni esecuzione verrà sempre dato il reset. Una volta che il segnale START va a 1, il primo bit relativo all'intestazione viene salvato e ci si sposta nello stato CAPTURE. Qui, nel primo ciclo di clock viene salvato il secondo bit dell'intestazione (sempre, poichè la sequenza è garantita essere di almeno 2 bit). Successivamente, viene salvato a ogni ciclo di clock il valore di W nel vettore `address_temp` e viene tenuta traccia del numero di bit arrivati fino a quel momento: in questo modo è possibile scorrere il vettore e, quando necessario, estendere l'indirizzo sui bit più significativi tramite uno shift. Quando START torna a 0, viene appunto eseguito lo shift sulle posizioni rimanenti, viene attivato il segnale di comunicazione con la memoria e si passa allo stato MEMORY. Qui, si aspetta un ciclo di clock grazie al segnale di comando `mem_done` per attendere la risposta della RAM. Nel ciclo successivo, vengono preparati tutti i valori delle uscite e il valore DONE. Infine, si torna allo stato IDLE e si aspetta una nuova sequenza.
In presenza del reset, il comportamento è il seguente: tutte le uscite `Z` vanno al valore di default, `DONE` va a 0, si torna nello stato IDLE e viene alzato il segnale `rst_signal`. Tale segnale è necessario per comunicare al secondo processo, quello relativo alla FSM, che è arrivato il reset. Quando ciò avviene, anche i valori memorizzati delle uscite tornano al valore di default.

4 TestBench

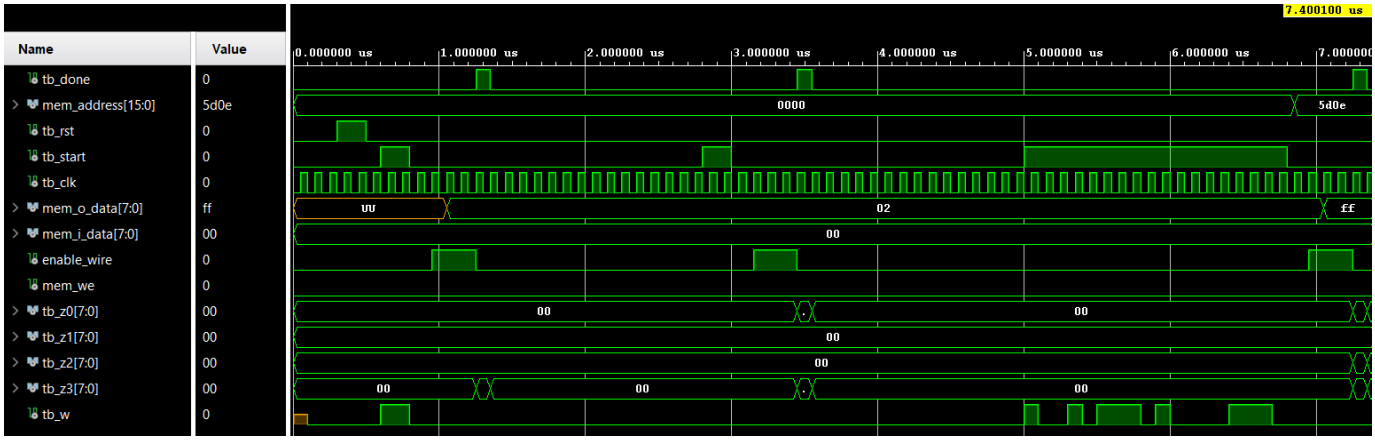
Ho creato alcuni TestBench per testare il comportamento nei casi limite che ho individuato.

Reset asincrono



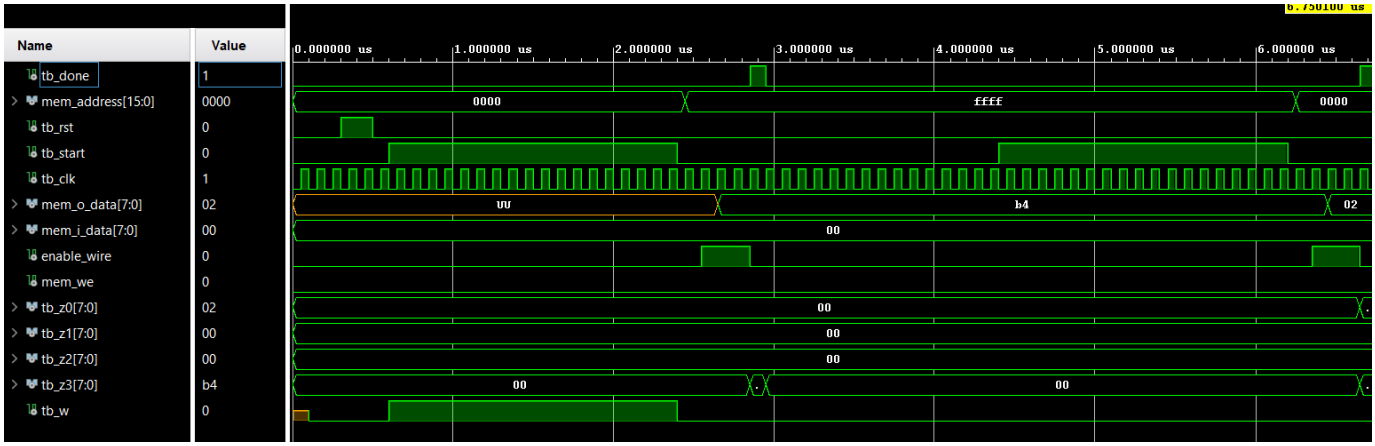
Il segnale di reset diventa alto in mezzo a una sequenza. Il programma scarta i valori letti precedentemente e al ciclo successivo ha subito inizio la nuova sequenza.

Sequenza da due 1 e due 0



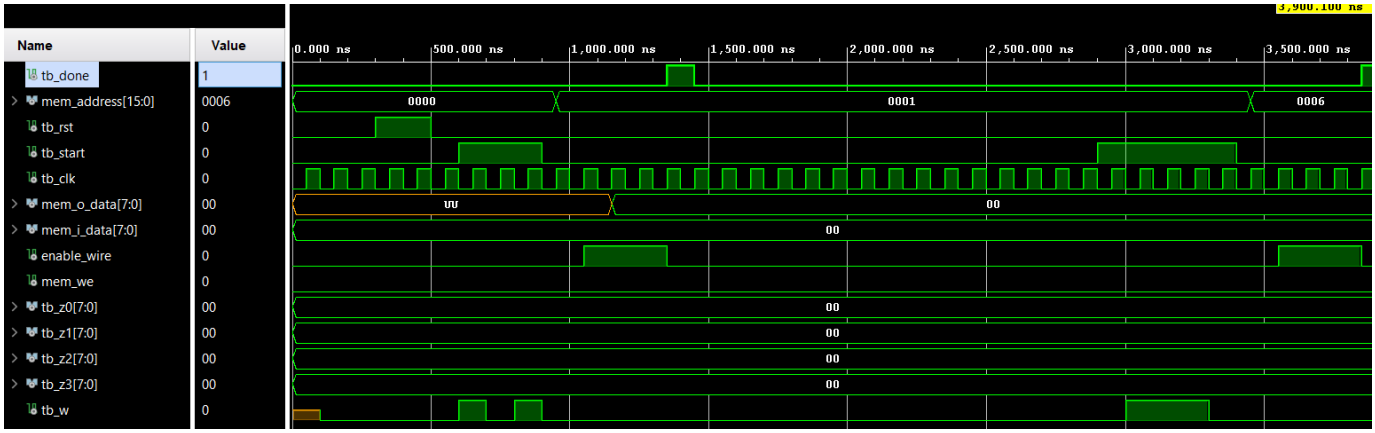
Per entrambi i registri Z0 e Z2 viene mostrato il valore memorizzato nella cella di memoria "0000000000000000".

Sequenza di 18 ’1’ e 18 ’0’



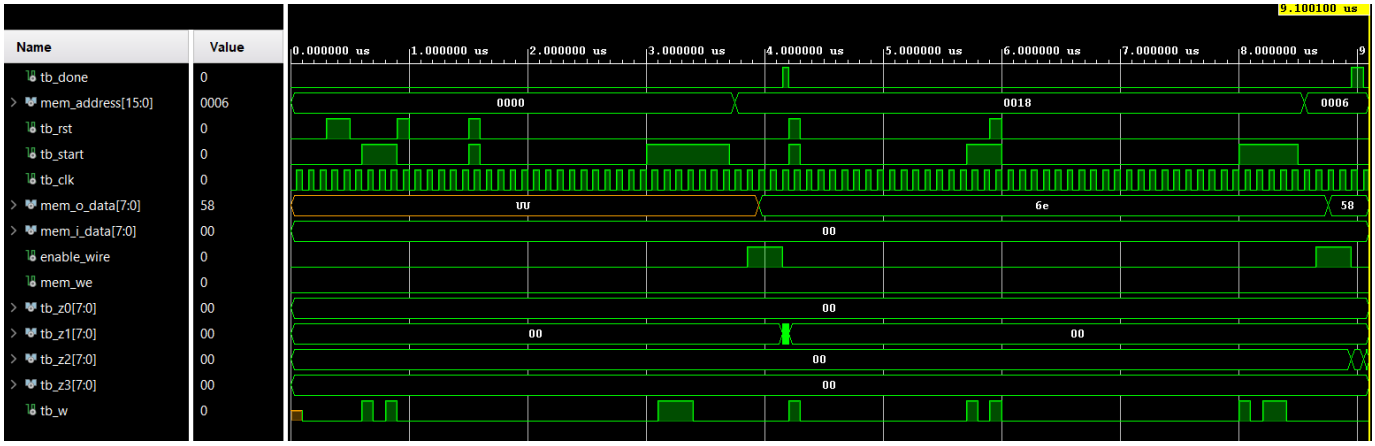
Vengono presi tutti i 18 bit.

Valore ”00000000” in memoria



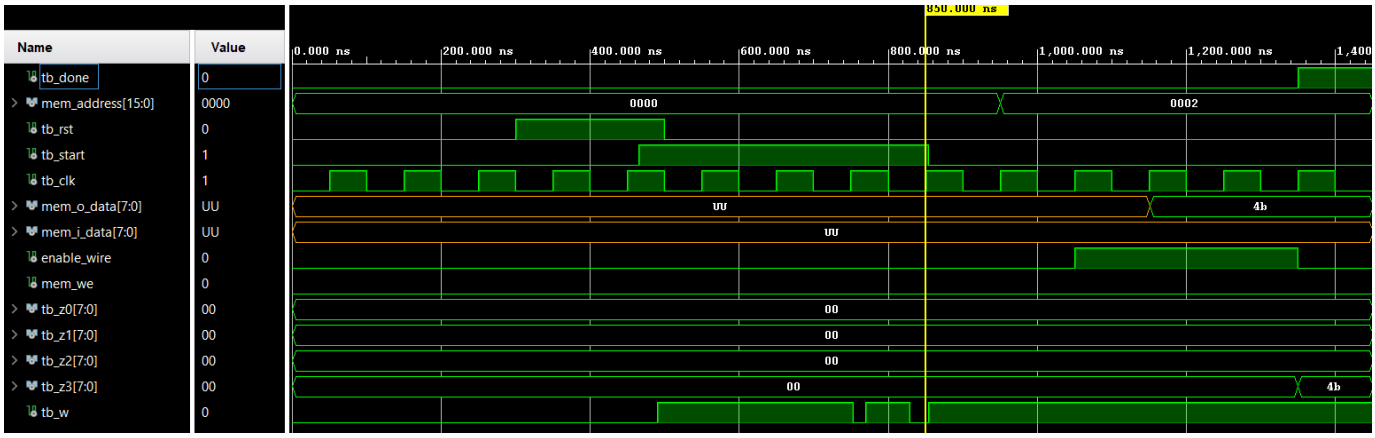
Viene mostrato il valore e DONE correttamente va a 1.

Multipli reset



Testa il corretto funzionamento nel momento in cui viene dato reset più volte di fila.

START e W non sincronizzati con il CLOCK



Correttamente viene preso il valore che W ha durante la fase di salita del CLOCK.

Conclusioni

Il componente sintetizzato supera correttamente tutti i test in Behavioral e Post-Synthesis Functional. Le ottimizzazioni che ho fatto riguardano la minimizzazione del numero di stati. Per attendere la risposta della RAM, invece che creare uno stato a parte, ho optato per un segnale intermedio che viene poi resettato in IDLE. Questo perchè si interroga la RAM una sola volta a ogni sequenza e in un solo uno stato. La fsm opera solo nel fronte di salita del clock: così posso salvare il valore che W ha appunto nel fronte di salita. È stato necessario cambiare lo stato e l'indice `num` nel falling edge per averlo poi aggiornato durante il rising edge.
Il tempo minimo di clock per il corretto funzionamento del programma è di 1ns per via del ritardo della RAM.