Prova Finale di Reti Logiche

Andrea Sanvito

15 Maggio 2024

Matricola: 983819
Codice Persona: 10814394

1 Requisiti del Progetto

1.1 Overview

Il progetto riguarda l'implementazione in VHDL di un modulo in grado di modificare una memoria RAM, seguendo determinate specifiche.

In particolare, viene dato in input al modulo un indirizzo da 16 bit della memoria addr e un valore intero k da 10 bit. Compito del modulo è quello di:

- accedere alla RAM all'indirizzo fornito in input;
- considerare la memoria come una sequenza di "coppie" di numeri, dove il primo rappresenta un dato, e il secondo un valore di credibilità del dato stesso.
- scorrere le k coppie in memoria una ad una, e ad ogni passo:
  - se il dato è specificato (ossia diverso da zero), lo si mantiene in memoria e si imposta il valore di credibilità seguente al massimo;
  - altrimenti, si aggiorna il dato in memoria con l'ultimo dato valido letto, e si decrementa il valore di credibilità;

L'implementazione deve essere completamente sincrona con un segnale di Clock fornito esternamente, eccetto per il segnale di Reset, che è invece asincrono. E' riportato di seguito un esempio.

1.2 Esempio

Consideriamo l'indirizzo di memoria iniziale addr pari a 128, mentre k pari a 16.

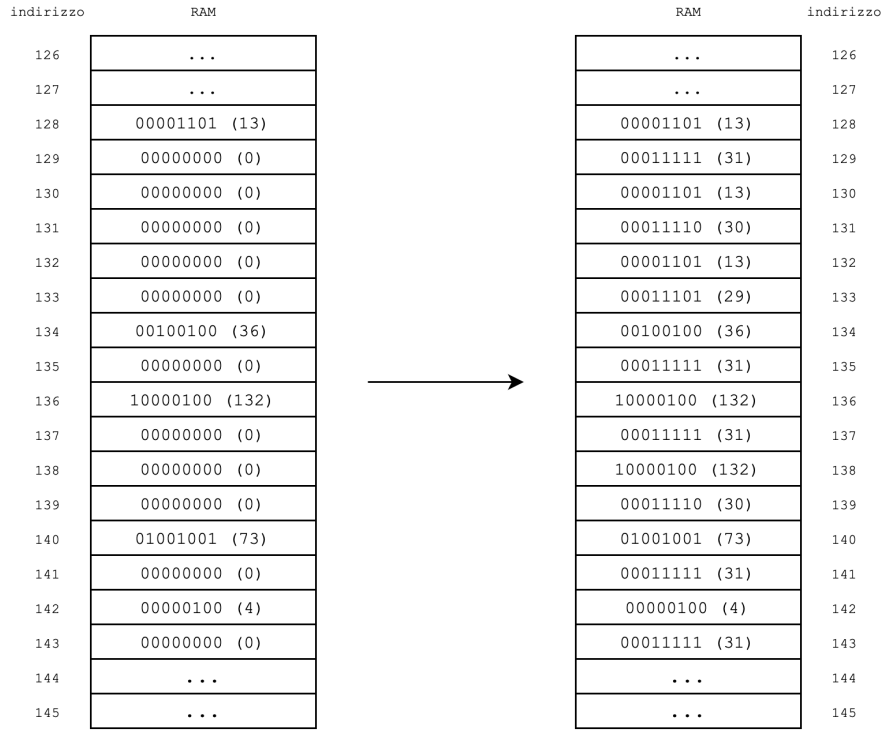

Figure 1: memoria RAM prima e dopo le operazioni svolte dal modulo

Il nostro modulo accede alla RAM all'indirizzo specificato e legge il dato: essendo questo diverso da zero, lo salva come "ultimo valore valido" e porta il valore di credibilità al massimo (da specifica, il massimo equivale a 31) e lo scrive nella cella seguente.
La cella successiva a quella appena scritta contiene uno zero, ossia un valore non specificato: il modulo lo sostituisce inserendo l'ultimo valore valido, e decrementa la credibilità, scrivendola nella cella successiva. Così, fino alla cella all'indirizzo 134, dove troviamo un dato diverso da zero: il modulo procede a sostituire l'ultimo valore valido letto, e a riportare la credibilità a 31, scrivendola nella cella successiva. Queste azioni vengono ripetute fino a quando k, il quale viene decrementato ogni volta che si legge un dato, arriva a zero.

1.3 Ipotesi Progettuali

Il progetto è stato svolto con determinate ipotesi:

- l'indirizzo addr deve essere valido, ossia non deve essere oltre l'ultimo indirizzo di memoria della RAM;
- la somma tra addr e k deve essere valida, ossia non deve andare oltre l'indirizzo finale della RAM;

Il comportamento del modulo implementato non è specificato per i casi sopra elencati.

2 Architettura

2.1 Overview

L'implementazione è stata ottenuta attraverso un singolo modulo, una FSM, con due processi. Essa comunica con la memoria, esegue computazione, e asserisce gli output.

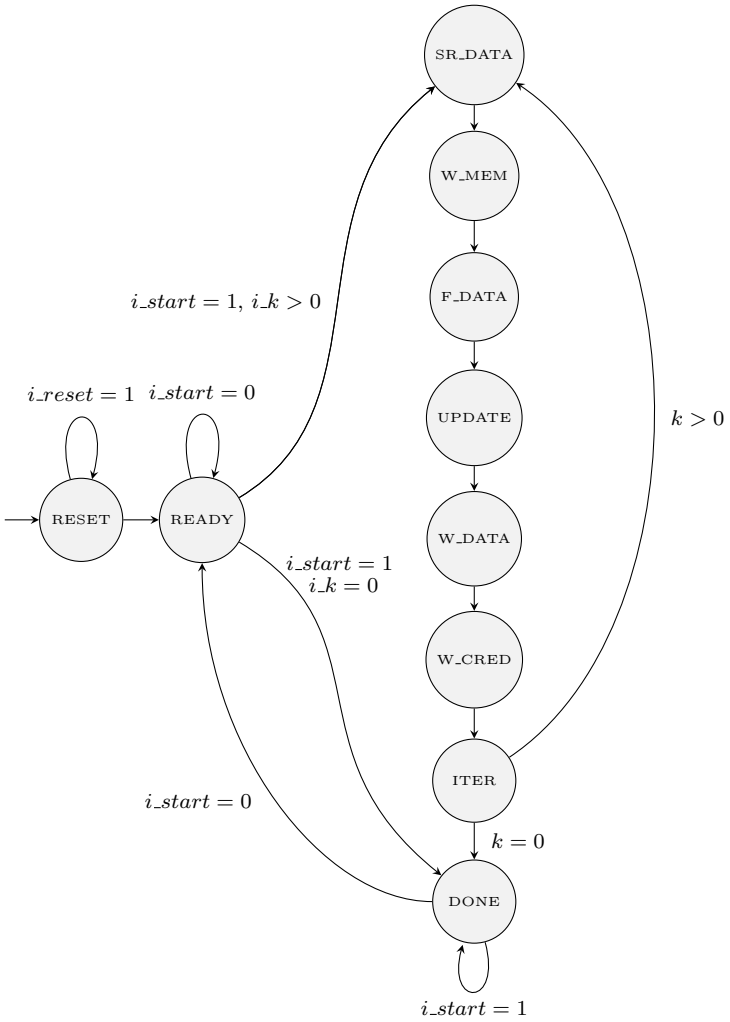

Figure 2: schema della FSM del modulo

2.2 Processi

Il modulo è basato su due processi:

- STATE_REG: è il processo che permette di aggiornare segnali e stato della FSM. Nella lista di sensibilità ha segnale di reset e segnale di clock. Per come è implementato, viene controllato se il processo è stato risvegliato dal segnale di reset, e in tal caso resetta il modulo, oppure se è stato risvegliato dalla variazione del clock. In caso il segnale sia passato da basso ad alto, vengono aggiornati i segnali ai valori "successivi", compreso il segnale di stato, permettendo così il funzionamento della FSM.

- LAMBDA_DELTA: è il processo che contiene le azioni da eseguire per ogni stato, in termine di update sia dei segnali interni che degli output.

2.3 Segnali Interni

All'interno del modulo sono stati utilizzati diversi segnali per la gestione dell'elaborazione. In particolare:

- k: nella fase iniziale dell'elaborazione, viene posto pari a i_k; decrementato ogni volta che una parola viene elaborata.

- addr: rappresenta l'indirizzo corrente che il modulo sta elaborando. Inizialmente è posto pari a i_addr, e incrementato a ogni lettura/scrittura sulla RAM;

- data: rappresenta il dato letto all'iterazione corrente;

- last_valid_data: usato per scrivere in memoria nelle celle in cui il dato non è specificato, rappresenta l'ultimo dato letto diverso da zero;

- credibility: rappresenta il valore di credibilità dell'ultimo dato valido letto;

Per ogni segnale interno e per ogni segnale di output, sono anche inizializzati dei segnali di "next". Questi servono a tenere traccia del valore che i segnali stessi devono avere al ciclo di clock successivo. Più in dettaglio, questi segnali di "next" vengono posti uguali ai loro corrispondenti segnali "correnti" ad ogni clock. Essendo che ad ogni transizione i segnali "correnti" vengono posti uguali ai segnali "next", se durante l'elaborazione, in un generico stato, un segnale di next viene cambiato, il corrispettivo segnale "corrente", al ciclo di clock seguente, verrà modificato.
L'implementazione di questi segnali è utile per evitare latch.

2.4 Stati

Le operazioni compiute dalla FSM dipendono dallo stato in cui essa si trova. Gli stati implementati sono:

- RESET: è lo stato iniziale della FSM e lo stato in cui la FSM si troverà dopo un segnale di reset in ingresso. Da specifica, la FSM rimarrà in questo stato fino a quando reset non verrà portato a 0 e start non verrà portato ad 1;

- READY: è lo stato di idling della FSM. La FSM rimane in attesa di un segnale di start;
- SET_READ_DATA: stato in cui la FSM comunica alla RAM la volontà di accedere ad essa, ad un determinato indirizzo addr;
- WAIT_MEM: la memoria RAM richiede un ciclo di clock in più per asserire in output i dati richiesti dalla FSM. Di conseguenza, il nostro modulo deve rimanere in attesa per un ciclo di clock;
- FETCH_DATA: stato in cui la FSM recupera il dato dalla RAM;
- UPDATE: la FSM esegue le operazioni a seconda dei dati letti.
- WRITE_DATA: la FSM scrive il dato nella RAM;
- WRITE_CREDIBILITY: la FSM scrive aggiorna il valore di credibilità nella RAM;
- ITERATE: step con cui i segnali interni alla FSM vengono aggiornati e "preparati" per la prossima iterazione;
- DONE: stato che sancisce la fine dell'elaborazione. La FSM esce da questo stato solamente quando il segnale in ingresso di start viene portato a 0. Dopodichè, la FSM tornerà allo stato READY portando il segnale di done a 0.

3 Test Benches

I test svolti sul modulo sono stati pensati appositamente per toccare i punti critici del modulo stesso, andando così a coprire tutti i possibili casi limite. In particolare, sono stati eseguiti test per le seguenti situazioni:

- segnale di start dato prima del segnale di reset nella fase iniziale del TB;
- K in ingresso pari a zero;
- Il primo indirizzo nella memoria della sequenza di parole di interesse contenente zero;
- condizioni tali da portare il valore di credibilità a zero durante l'elaborazione;
- test contenenti più segnali di reset e start, ergo condizioni in cui il modulo deve compiere più di una elaborazione;
- memoria contenente solo zeri.

Il modulo implementato è riuscito ovviamente a eseguire tutti i test bench senza problemi.

4 Risultati Sperimentali

5 Conclusioni