POLITECNICO
MILANO 1863

Progetto di Reti Logiche

Gianmarco Schifone
10846556

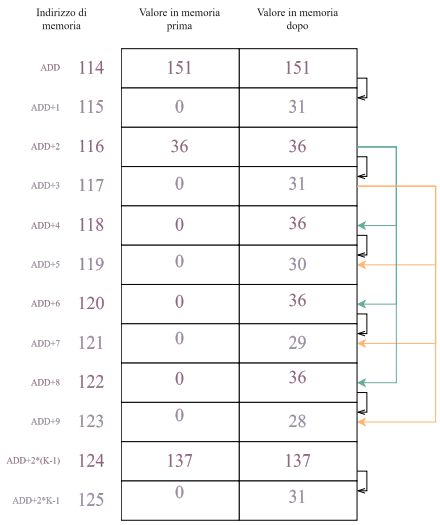
Prof. Gianluca Palermo
a.a. 2023-2024

Indice

Capitolo 1

Introduzione

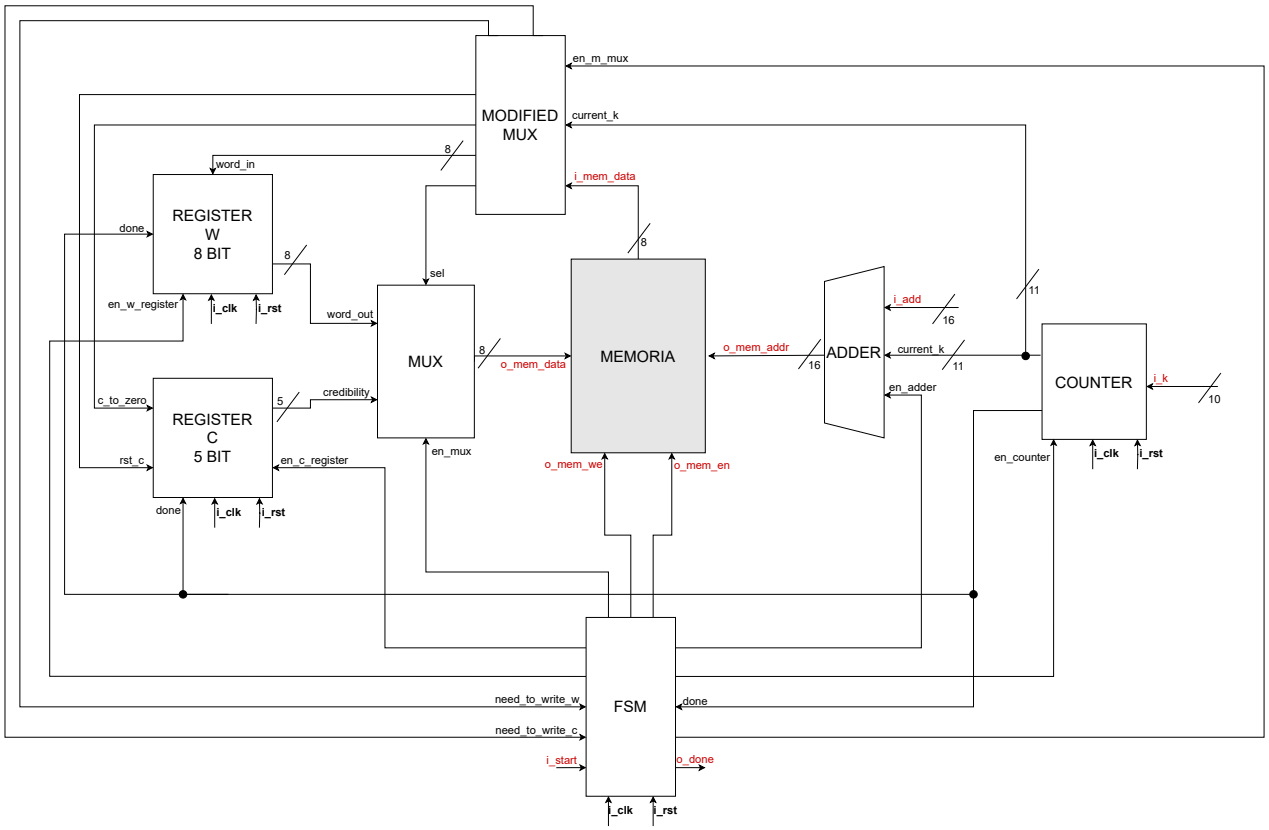
Il progetto prevede l'implementazione di un modulo hardware, descritto in VHDL, che aggiorna una sequenza di parole memorizzate in memoria, a partire da un dato indirizzo, ogni due byte. Nello specifico, il modulo sostituisce le parole il cui valore non è specificato, ovvero è 0, con l'ultima parola valida letta, ovvero maggiore di 0, e associa ad ogni elemento della sequenza, scrivendo nel byte successivo all'elemento, un valore di credibilità compreso tra 31 a 0 che decresce di 1 ad ogni sostituzione di una parola non valida con l'ultima parola valida letta e viene reimpostato a 31 ad ogni lettura di una parola valida.



Esempio

Capitolo 2

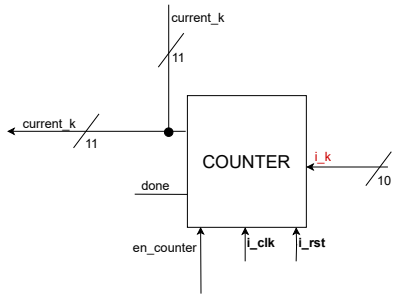
Architettura



Datapath

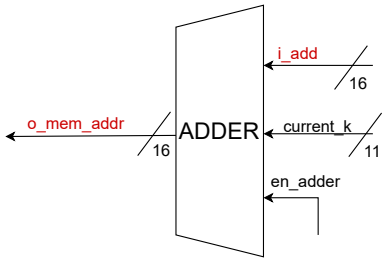
2.1 Contatore

Un contatore che riceve in ingresso il numero i_k di parole della sequenza da elaborare e incrementa il segnale memorizzato, e trasferito nel segnale d'uscita $current\_k$, da 0 a $2(i\_k - 1) + 1$, valore dopo il quale l'elaborazione deve terminare abilitando un segnale *done*.



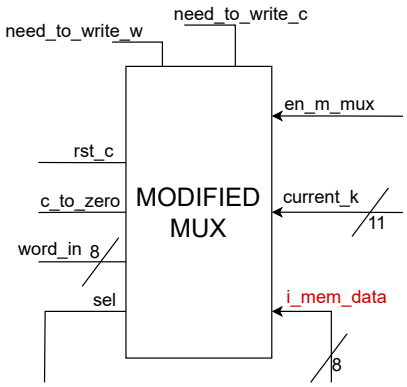
2.2 Addizionatore

Un addizionatore che riceve in ingresso l'indirizzo i_add a partire dal quale è memorizzata la sequenza di parole, e il segnale $current\_k$. Il segnale d'uscita o_mem_addr è la somma dei due, si interfaccia con la memoria e contiene l'indirizzo attualmente in esame.



2.3 Componente basato su MUX

Un componente, basato su un multiplexer, che riceve in ingresso la parola i_mem_data, letta in memoria, e $current\_k$. I casi possibili sono tre:



2.3.1 *current_k* dispari

Ovvero il suo LSB è uguale a 1, sull'indirizzo attualmente in esame andrà scritto il valore di credibilità associato alla parola precedente. I segnali d'uscita *need_to_write_c* e sel conterranno il valore 1.

2.3.2 *current_k* pari

Ovvero il suo LSB è uguale a 0.

$i\_mem\_data > 0$

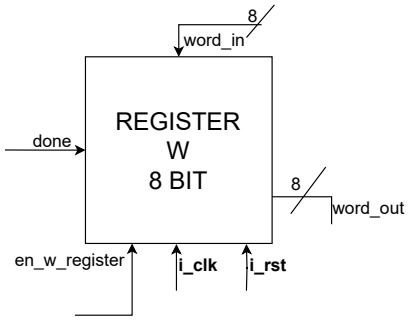
La parola letta contiene un valore valido della sequenza, è necessario memorizzarla in modo da poterla eventualmente scrivere successivamente in memoria. Il segnale d'uscita *word_in* conterrà il valore di *i_mem_data* e il segnale d'uscita *rst_c* conterrà il valore 1.

$i\_mem\_data = 0$

La parola letta contiene un valore non valido della sequenza, è necessario sostituirla scrivendo in memoria l'ultimo valore valido letto. Il segnale d'uscita sel conterrà il valore 0 e il segnale d'uscita *need_to_write_w* conterrà il valore 1. Se non si dispone di un valore valido da scrivere, si lascia invariato il contenuto dell'indirizzo in esame e il segnale d'uscita *c_to_zero* conterrà il valore 1.

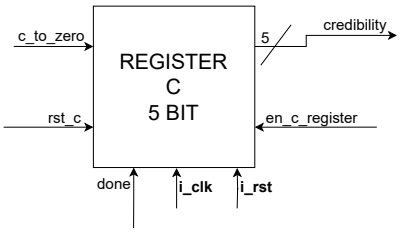
2.4 Registro di parola

Un registro che riceve in ingresso la parola *word_in* da memorizzare al suo interno, riportandola sul segnale d'uscita *word_out*. Dispone inoltre di un segnale d'ingresso *done* che resetta il registro.



2.5 Registro di credibilità

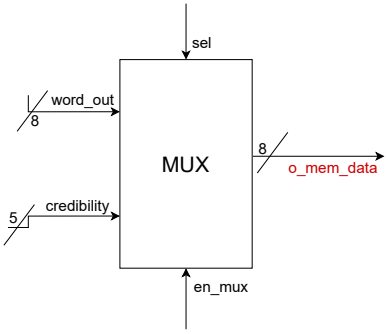
Un registro che memorizza il valore credibility di credibilità attuale, decrementandolo ad ogni scrittura di una parola in memoria. Dispone di un segnale d'ingresso *done* che resetta il registro riportando il valore di credibilità a 31, di un segnale d'ingresso *rst_c* che resetta il registro riportando il valore di credibilità a 31, di un segnale d'ingresso *c_to_zero* che imposta il valore di credibilità a 0.



2.6 Multiplexer

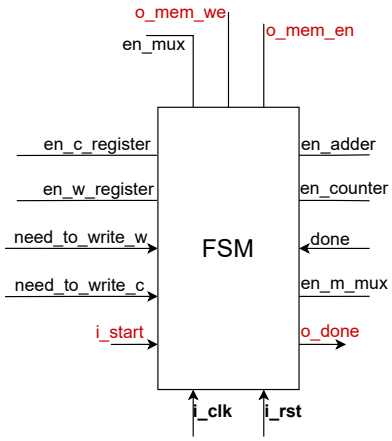
Un multiplexer che trasferisce sul segnale d'uscita *o_mem_data*, che si interfaccia con la memoria, la parola *word_out* da scrivere nella sequenza o il valore di credibilità *credibility* da associare alla parola memorizzata

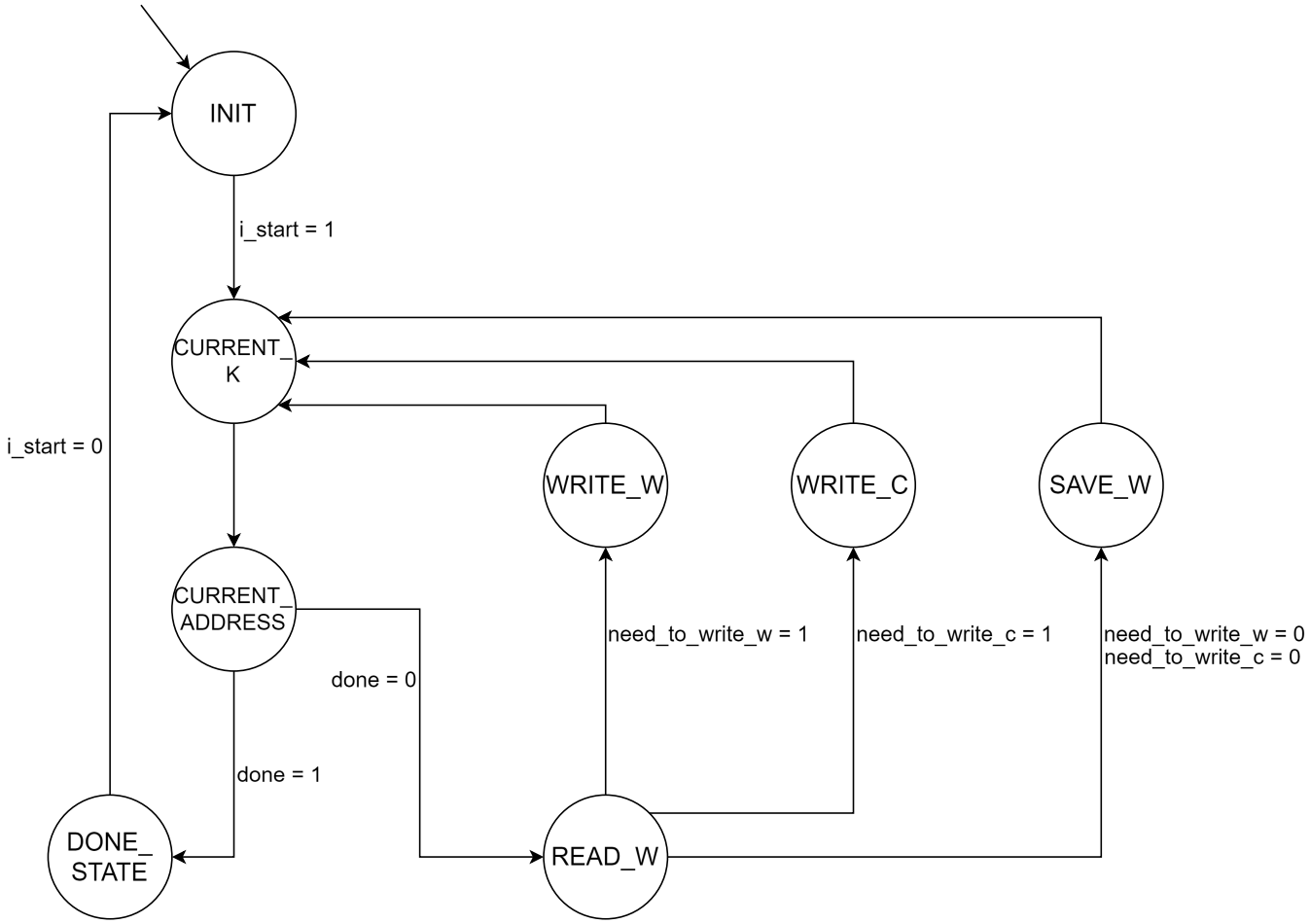
nell'indirizzo subito prima, in base al segnale selettore sel ricevuto in ingresso



2.7 FSM

Una FSM che gestisce l'attivazione e disattivazione dei componenti in base allo stato in cui si trova il sistema. Inoltre, si occupa di abilitare la scrittura in memoria in base ai segnali d'ingresso *need_to_write_c* e *need_to_write_w*. Infine, si occupa di avviare l'elaborazione a seguito di un segnale d'ingresso *i_start*, e si occupa di terminare l'elaborazione, la cui fine è segnalata dal segnale d'ingresso *done*, portando a 1 il valore del segnale d'uscita *o_done*.





Stati della FSM

2.7.1 INIT

Stato di reset, in cui il sistema attende che il segnale *i_start* vada a 1.
I segnali d'uscita non specificati sono uguali a 0.

2.7.2 CURRENT_K

Stato in cui viene calcolato l'offset dell'indirizzo di memoria da esaminare.
$en\_counter = 1$
I segnali d'uscita non specificati sono uguali a 0.

2.7.3 CURRENT_ADDRESS

Stato in cui viene calcolato l'indirizzo di memoria da esaminare e richiesta la lettura in memoria.
$en\_adder = 1$
$o\_mem\_en = 1$
I segnali d'uscita non specificati sono uguali a 0.

2.7.4 READ_W

Stato in cui viene letto il valore contenuto nell'indirizzo di memoria in esame.
$en\_adder = 1$
$o\_mem\_en = 1$
$en\_m\_mux = 1$
I segnali d'uscita non specificati sono uguali a 0.

2.7.5 SAVE_W

Stato in cui viene memorizzato in un registro il valore valido, appartenente alla sequenza, letto nell'indirizzo di memoria in esame.
$en\_adder = 1$
$o\_mem\_en = 1$
$en\_m\_mux = 1$
$en\_w\_register = 1$
$en\_c\_register = 1$
I segnali d'uscita non specificati sono uguali a 0.

2.7.6 WRITE_C

Stato in cui viene scritto, nell'indirizzo di memoria in esame, il valore di credibilità associato all'ultima parola letta nella sequenza.
$en\_adder = 1$
$o\_mem\_en = 1$
$en\_m\_mux = 1$
$en\_c\_register = 1$
$en_mux = 1$
$o\_mem\_we = 1$
I segnali d'uscita non specificati sono uguali a 0.

2.7.7 WRITE_W

Stato in cui viene sostituito il valore non valido, presente nell'indirizzo di memoria in esame, con l'ultimo valore valido letto.
$en\_adder = 1$
$o\_mem\_en = 1$
$en\_m\_mux = 1$
$en_mux = 1$
$o\_mem\_we = 1$
I segnali d'uscita non specificati sono uguali a 0.

2.7.8 DONE_STATE

Stato di fine elaborazione della sequenza.
$o\_done = 1$
I segnali d'uscita non specificati sono uguali a 0.

Capitolo 3

Risultati sperimentali

La sintesi del modulo HW viene eseguita senza generare errori.

3.1 Utilization Report

Inferred latch: 0

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	102	0	41000	0.25
LUT as Logic	102	0	41000	0.25
LUT as Memory	0	0	13400	0.00
Slice Registers	42	0	82000	0.05
Register as Flip Flop	42	0	82000	0.05
Register as Latch	0	0	82000	0.00
F7 Muxes	0	0	20500	0.00
F8 Muxes	0	0	10250	0.00

3.2 Timing Report

Required time: 20ns

```
Slack (MET) :             16.797ns  (required time - arrival time)
```

La Behavioral Simulation e la Post-Synthesis Functional Simulation eseguono correttamente tutti i test creati. Riporto di seguito la descrizione dei test più rilevanti effettuati.

- Test bench standard che verifica la corretta elaborazione di una sequenza.
- Test bench che verifica la corretta elaborazione di una sequenza nella quale un valore di credibilità raggiunge lo 0 senza decrementarlo ulteriormente. E' stato scritto tramite uno scenario di input che contiene per almeno 31 volte di seguito il valore 0 all'interno della sequenza.
- Test bench che verifica la corretta elaborazione di una sequenza la cui parola iniziale è uguale a 0.
- Test bench che verifica la corretta elaborazione di più sequenze.
- Test bench che verifica il corretto funzionamento del sistema a seguito di un segnale d'ingresso $i\_k = 0$.
- Test bench che verifica il corretto funzionamento del sistema a seguito di un segnale d'ingresso $i\_k = 1023$.
- Test bench che verifica il corretto funzionamento del sistema a seguito di un reset avvenuto durante l'elaborazione di una sequenza.

Capitolo 4

Conclusioni

Il modulo HW progettato soddisfa le esigenze funzionali descritte in precedenza, con la corretta gestione dei casi limite. Inizialmente, il processo di progettazione si è concentrato sulla costruzione del datapath, che svolge un ruolo fondamentale nella definizione del flusso di dati all'interno del modulo hardware. Successivamente, è stata sviluppata la FSM responsabile della gestione del datapath, necessaria per garantire un controllo coerente e affidabile delle operazioni. Una volta completata la progettazione concettuale, si è proceduto alla realizzazione pratica attraverso la descrizione in VHDL dei componenti. Durante questa fase, sono stati eseguiti test approfonditi sui singoli componenti al fine di garantire il corretto funzionamento e l'aderenza agli obiettivi prestabiliti. Infine, con il completamento dei test e l'assicurazione della funzionalità di ciascun componente, sono stati collegati tra loro i moduli per creare un sistema integrato e coerente. Eventuali discrepanze o inefficienze individuate sono state affrontate con modifiche mirate al datapath e alla FSM, assicurando così un'implementazione ottimale.