Prova Finale (Progetto di Reti Logiche)

Prof. Fabio Salice – Anno 2019/2020

Daniele Giorgianni

Indice

1 Introduzione

1.1 Scopo del progetto

Lo scopo del progetto è quello di sviluppare un componente hardware in VHDL in grado di identificare l'appartenenza di un indirizzo di memoria ad una delle aree di memoria di frequente accesso, denominate Working Zone, così da poter ridurre la potenza dissipata dalla macchina.

1.2 Specifiche generali

Le Working Zone sono un intervallo di indirizzi dalla dimensione fissata e vengono specificate all'interno della memoria tramite il loro indirizzo base.

Gli indirizzi da trattare sono composti da 8 bit di cui solo 7 bit significativi, così da poter accedere agli indirizzi della memoria che vanno dallo 0 al 127.

Nel caso l'indirizzo da codificare facesse parte di una delle Working Zone, che sono otto, verrà opportunamente codificato prima di uscire dal componente hardware sintetizzato per essere scritto in memoria, se non dovesse farne parte verrebbe scritto così come è stato letto.

La codifica dell'indirizzo prevede che il bit più significativo sia settato a 1, i tre che lo precedono indicheranno il numero della Working Zone d'appartenenza e i quattro meno significativi indicheranno l'offset rispetto all'indirizzo base della Working Zone con codifica One-Hot.

1.3 Interfaccia del componente

L'interfaccia del componente, come da specifica, è la seguente:

```
entity project_reti_logiche is
    port(
        i_clk       :   in std_logic;
        i_start     :   in std_logic;
        i_rst       :   in std_logic;
        i_data      :   in std_logic_vector(7 downto 0);
        o_address   :   out std_logic_vector(15 downto 0);
        o_done      :   out std_logic;
        o_en        :   out std_logic;
        o_we        :   out std_logic;
        o_data      :   out std_logic_vector(7 downto 0)
        );
end project_reti_logiche;
```

In particolare:

- i_clk indica il segnale di clock in ingresso;
- i_start se viene portato ad 1 dà il via all'elaborazione e dovrà rimanere alto per tutta la codifica, quando questa sarà terminata potrà essere riportato a 0 per dare la possibilità al modulo di cominciare con un'altra elaborazione;
- i_rst è il segnale di reset, che porta il modulo allo stato di partenza in qualunque situazione esso si trovi;
- i_data contiene l'indirizzo che viene letto dalla memoria
- o_address indica l'indirizzo della cella di memoria a cui punta il modulo;
- o_done se vale 1 rappresenta la terminazione della codifica, 0 quando il processo elaborativo è in corso;
- o_en è un bit che deve valere 1 per poter interagire con la memoria, 0 altrimenti;
- o_we è un bit che indica la possibilità di scrittura in memoria, deve valere 1 per scrivere e 0 per leggere;
- o_data contiene il valore che è stato elaborato dal modulo alla fine del processo.

1.4 Dati e descrizione della memoria

La memoria è così suddivisa:

Indirizzo	Contenuto
0	Indirizzo base WZ_0
1	Indirizzo base WZ_1
2	Indirizzo base WZ_2
3	Indirizzo base WZ_3
4	Indirizzo base WZ_4
5	Indirizzo base WZ_5
6	Indirizzo base WZ_6
7	Indirizzo base WZ_7
8	Indirizzo da codificare
9	Indirizzo (eventualmente) codificato

Ogni Working Zone è composta da quattro indirizzi a partire da quello base.

L'indirizzo da trattare è presente nell'indirizzo 8 della memoria, mentre l'indirizzo (eventualmente) codificato verrà scritto nell'indirizzo 9.

2 Design

Il componente hardware sviluppato è basato su una macchina a stati finiti sincrona rispetto al fronte di salita del segnale di clock.

La FSM è costituita da sei stati, il primo è quello di inizializzazione, dove si rimane fino a che il segnale i_start non viene portato a 1. Una volta terminata l'elaborazione essa si troverà nello stato finale dove alzerà a 1 il segnale di terminazione o_done e rimarrà in esso fino a che il segnale i_start non verrà portato a 0, condizione per la quale lo stato corrente tornerà ad essere il primo. La FSM si riporterà allo stato di inizializzazione nel momento in cui il segnale i_rst venisse alzato (come richiesto da specifica) e qualora il segnale i_start dovesse abbassarsi, per errore, durante la computazione.

2.1 Stati della macchina

L'insieme degli stati costituenti la FSM è {S0, S1, S2, S3, S4, S5}, nel dettaglio:

- S0 è lo stato di inizializzazione, dove le variabili vengono settate ai valori di default e si attende che i_start commuti a 1. Ogni qual volta i_rst venisse alzato o i_start venisse abbassato durante l'elaborazione si ritornerà in questo stato;
- S1 è lo stato in cui i segnali vengono settati per la lettura e successivamente viene memorizzato il dato letto in una variabile;
- S2 è lo stato in cui si verifica che l'indirizzo letto appartenga ad una delle otto Working Zone, nel caso affermativo lo stato successivo sarà S3, altrimenti S4;
- S3 è lo stato in cui si codifica l'indirizzo, come da specifica, e successivamente verrà scritto in memoria;
- S4 è lo stato in cui l'indirizzo letto viene ricopiato in uscita, poiché non appartenente ad alcuna Working Zone;
- S5 è lo stato di terminazione, dove si comunica il termine dell'elaborazione alzando il segnale o_done e si attende che i_start venga abbassato.

2.2 Scelte progettuali

Il componente hardware, basato sulla FPGA xc7a200tfbg484-1, è stato descritto tramite un singolo processo, che si occupa dell'intera elaborazione.

È stato deciso di non memorizzare nel componente l'indirizzo base delle Working Zone così da ridurre gli elementi di memoria ad esso necessari.

Una volta letto l'indirizzo da codificare, questo verrà confrontato con gli indirizzi base delle WZ così da poterne stabilire l'eventuale appartenenza.

Il confronto tra l'indirizzo da codificare e l'indirizzo base della WZ avviene per sottrazione del secondo dal primo, dopo aver convertito entrambi ad interi. Se il risultato di tale operazione è compreso tra 0 e 3 allora l'indirizzo farà parte della WZ e il risultato indicherà l'offset rispetto alla base, altrimenti si passerà al confronto con l'indirizzo base della successiva.

Il confronto dell'indirizzo da codificare con quello base delle WZ comincia dalla prima WZ, specificata nella cella di memoria 0. Qualora si dovesse giungere ad una conclusione di appartenenza, le (eventuali) WZ rimaste, ovvero non confrontate, verranno ignorate e la macchina si porterà allo stato successivo per la codifica dell'indirizzo, in caso contrario, esaurite dunque le WZ da confrontare, la macchina andrà nello stato successivo dove copierà l'indirizzo da codificare in quello di uscita.

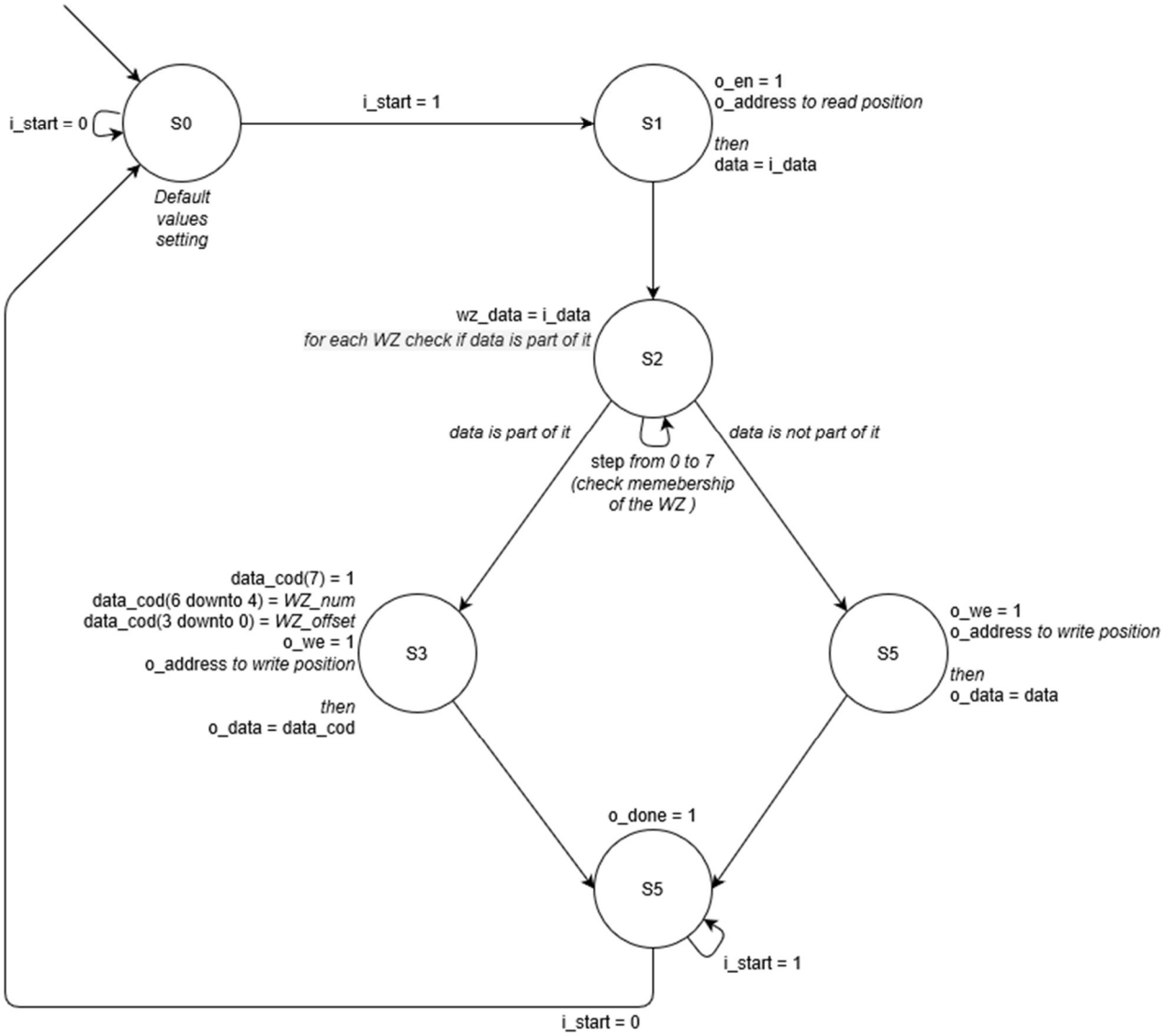
Nella pagina seguente è riportata l'immagine della FSM.

L'immagine è privata di due tipologie di collegamenti per facilitarne la lettura, i quali sono:

- I collegamenti uscenti da ogni stato diretti allo stato iniziale S0 dovuti alla presenza del segnale di reset;
- I collegamenti uscenti dagli stati {S1, S2, S3, S4, S5} diretti allo stato iniziale S0 dovuti al fatto che ogni stato proceda nella sua esecuzione solamente se i_start rimane alto, altrimenti lo stato diventerà S0 (come specificato all'inizio di questo capitolo).

Variables used by the process:

```
type state_type is (S0, S1, S2, S3, S4, S5);
signal State : state_type;
shared variable step: integer range 0 to 7;
shared variable clk_cycle: integer range -1 to 1;
shared variable data, wz_data, data_cod : std_logic_vector(7 downto 0);
```



3 Testbench

Il componente sintetizzato passa positivamente sia i test forniti dai docenti che quelli autonomamente generati per andare a coprire quante più casistiche possibili.

I test sviluppati, ovviamente sincroni col clock, possono essere raggruppati a seconda dei casi da loro coperti:

- Varie configurazioni della memoria, in modo da testare un numero elevato di indirizzi da codificare, sia nel caso di appartenenza ad una WZ sia nel caso opposto, e assegnare svariati indirizzi base alle WZ contenute in memoria (tra cui casi limite come l'indirizzo 0 e il 124);
- Start multipli, in maniera da valutare sia il corretto inizio di un'elaborazione che la sua corretta esecuzione durante l'intero processo di codifica;
- Reset multipli, così da verificare la corretta transazione allo stato iniziale da parte della macchina nel momento in cui il segnale di reset venisse alzato in uno stato qualunque tra quelli disponibili.

Queste tre tipologie sono state spesso combinate tra loro per generare test il più possibile completi.

4 Conclusioni

4.1 Risultati della sintesi

Il componente hardware implementato, come sopra descritto, supera positivamente i test sia nelle "Behavioral Simulation" che nelle "Post – Synthesis Functional Simulation", utilizzando un periodo di clock pari a 100ns.

Utilizzando il pulsante "Schematic" presente nel menù sotto la voce "Synthesized Design" è possibile osservare il numero delle varie risorse utilizzate dal componente, che sono:

- 176 Cells;
- 213 Nets;
- 81 LUT (corrispondente allo 0.06% della quantità disponibile);
- 47 FF (corrispondente allo 0.02% della quantità disponibile).

4.2 Ottimizzazioni

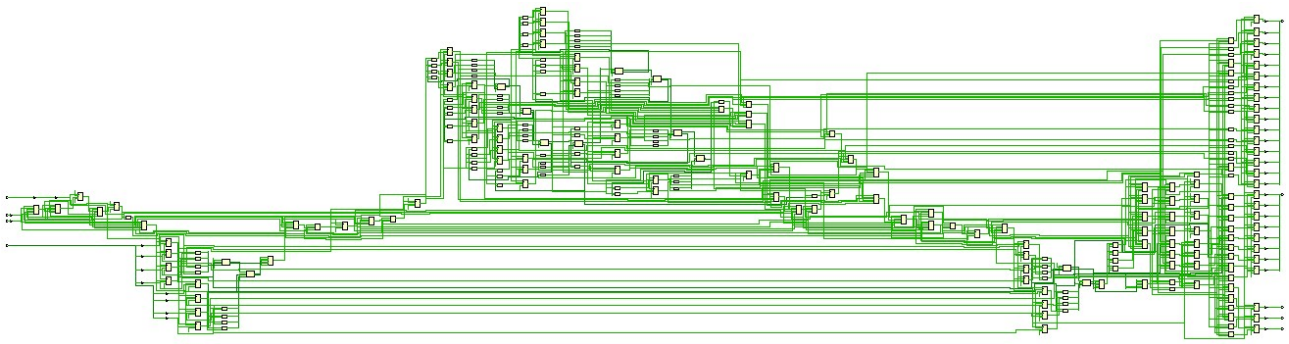
La FSM implementata nella sua prima versione utilizzava le risorse come segue:

- 321 Cells;
- 411 Nets;
- 175 LUT;
- 92 FF.

A seguito di una attenta analisi del codice scritto è stato possibile ridurre il numero delle variabili utilizzate, eliminare assegnamenti superflui ed ottimizzare lo stesso andando ad inserire opportune condizioni “if – then – else”. Questo ha permesso di arrivare ad avere:

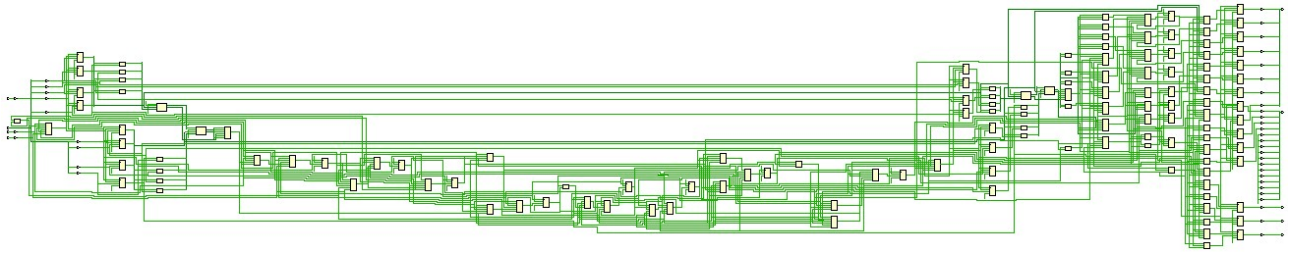
- 299 Cells;
- 389 Nets;
- 157 LUT;
- 88 FF.

Con uno schema come segue:



Una svolta al numero di risorse utilizzate è stata permessa, oltre ad una ulteriore riduzione delle variabili impiegate (che ha portato alla riscrittura di alcune parti del codice), dall’inserimento di un range per le variabili di tipo intero utilizzate. Permettendo di avere i valori indicati nel capitolo 4.1.

Lo schema di quest’ultima ottimizzazione è riportato di seguito al fine di confrontarlo col precedente in termini di area utilizzata.



Nella pagina seguente è possibile osservare lo schema nella sua interezza.