Prova finale di Reti Logiche

Giorgio Colomban (matricola: 890225, codice persona: 10555274)
Stefan Djokovic (matricola: 886860, codice persona: 10610473)

Docente: William Fornaciari

13 settembre 2020

Indice

1 Specifica del progetto

La specifica del progetto [1] richiede la descrizione in VHDL di un componente hardware che possa codificare indirizzi seguendo il metodo basato sulle **Working Zone** [2] Una Working Zone è definita come un intervallo di dimensione fissa (da qui in poi *Dwz*) che parte da un indirizzo base noto. All'interno del componente saranno presenti più di una Working Zone (numero da qui poi indicato con *Nwz*). Dato un'indirizzo e un *Nwz* in input, il componente ricodificherà l'indirizzo (secondo le regole illustrate nella prossima sezione di questo documento) se questo dovesse appartenere ad una delle *Nwz* Working Zones, altrimenti lo restituirà in input senza alcuna modifica, fatta eccezione per un bit (*WZBIT*, spiegato più nel dettaglio nel prossimo paragrafo) con valore '0' concatenato all'inidirizzo iniziale.

1.1 Metodo di codifica

Dato un indirizzo a 7 bit in input (da qui in poi *INPUT*, il metodo di codifica Working Zone funziona nel seguente modo:

- Si verifica se *INPUT* appartiene ad una delle *Nwz* o no
- Nel caso *INPUT* **non appartenga** a una Working Zone, *WZBIT* verrà settato a '0' e l'output sarà **WZBIT & INPUT**, dove & è il simbolo di concatenazione
- Nel caso *INPUT* **appartenga** ad una Working Zone, e detto *WZNUM* il numero della Working Zone a cui *WZOFF* l'offset tra l'indirizzo iniziale della Working Zone *WZNUM* e *INPUT* codificato *One-Hot* (come illustrato successivamente), *WZBIT* verrà portato a '1' e l'indirizzo di output sarà **WZBIT & WZNUM & WZOFF**
- Il metodo di codifica **One-Hot**, nel nostro caso su 4 bit, dato che $Dwz = 4$, funziona nel seguente modo:
  - Se il valore da codificare è 0, allora *WZOFF = 0001*
  - Se il valore da codificare è 1, allora *WZOFF = 0010*
  - Se il valore da codificare è 2, allora *WZOFF = 0100*
  - Se il valore da codificare è 3, allora *WZOFF = 1000*

1.2 Descrizione della RAM

La RAM con cui il componente dovrà interagire e descritta nel seguente modo, e contiene i seguenti valori:

Indirizzo	**Contenuto**
0	Indirizzo di base Working Zone 0
1	Indirizzo di base Working Zone 1
2	Indirizzo di base Working Zone 2
3	Indirizzo di base Working Zone 3
4	Indirizzo di base Working Zone 4
5	Indirizzo di base Working Zone 5
6	Indirizzo di base Working Zone 6
7	Indirizzo di base Working Zone 7
8	Indirizzo (*INPUT*) da codificare
9	Indirizzo in cui caricare l'indirizzo ottenuto dalla codifica di *INPUT*

1.3 Interfaccia del componente

L'interfaccia del componente da descrivere viene fornita dalla specifica:

```
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;
```

1.4 Esempio 1

Qui segue un esempio di codifica di un *INPUT* appartenente ad una Working Zone. *WZ_n* indica l'indirizzo di base della n-esima working zone.

WZ_0	WZ_1	WZ_2	WZ_3	WZ_4	WZ_5	WZ_6	WZ_7
4	13	22	31	37	45	77	91

INPUT	38

Codifica:

WZBIT	*WZNUM*	*WZOFF*
1	100	0010

1.5 Esempio 2

Qui segue un esempio di codifica di un *INPUT* **non** appartenente ad una Working Zone. *WZ_n* indica l'indirizzo di base della n-esima working zone.

WZ_0	WZ_1	WZ_2	WZ_3	WZ_4	WZ_5	WZ_6	WZ_7
4	13	22	31	37	45	77	91

INPUT	43

Codifica:

WZBIT	*INPUT*
0	0101011

2 Implementazione

2.1 Descrizione ad alto livello

Il componente è stato descritto in VHDL tramite tre processi, che rappresentano un circuito puramente combinatorio ed un circuito sequenziale:

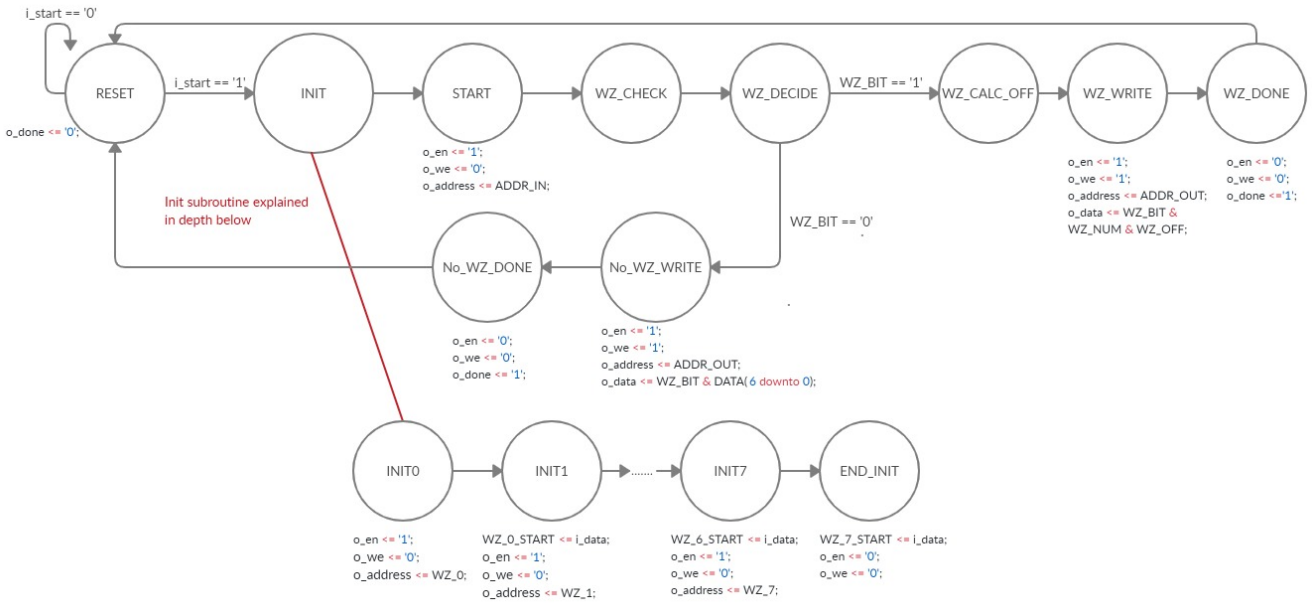
- **FSM_state_update** è un processo puramente sequenziale, che, sul fronte di salita del clock, aggiorna lo stato corrente allo stato successivo
- **FSM_state_sequence** è un processo puramente combinatorio, che, ad ogni modifica dello stato corrente da parte del circuito sequenziale, calcola quale sarà lo stato successivo
- **MAIN** è il processo principale, che sul fronte di discesa del clock effettua le computazioni relative allo stato corrente, descritte più nel dettaglio per il singoli stati nella sezione successiva.

Ad alto livello il componente, dopo aver ricevuto un segnale *i_start* alto, inizia la subroutine *INIT*, che consiste nel leggere dalla RAM e salvare in appositi segnali gli indirizzi base delle singole Working Zone. Successivamente comincia la computazione vera e propria, che consiste nel confrontare *INPUT* con tutte le Working Zone per vedere se appartiene ad una di queste. Se si, il componente procede a calcolare l'offset tra *INPUT* e l'indirizzo base della Working Zone, per poi scriverlo nella cella di RAM designata per l'output, per poi segnalare di aver finito e ritornare allo stato di *RESET*, dal quale si potrà effettuare un'altra codifica.
Se invece *INPUT* **non** appartiene ad una Working Zone, il componente procederà direttamente con la fase di scrittura in RAM dell'output e ritorno allo stato di *RESET*.

2.2 Finite State Machine

2.2.1 Grafo

In questo sottoparagrafo viene riportata una rappresentazione grafica della FSM. Sono stati omessi tutti gli archi che portano da ogni stato a *RESET* quando *i_rst* viene portato a '1' per leggibilità.



2.2.2 Descrizione degli stati

La seguente tabella contiene una breve descrizione ad alto livello del funzionamento dei singoli stati, rappresentati nella FSM nel sottoparagrafo precendente.

Stato	**Funzionamento**
RESET	Ogni volta che *i_rst* viene portato a '1', questo diventa lo stato attuale e *o_done* verrà portato a '0'. Passerà alla subroutine di INIT quando *i_start* verrà portato a '1'
INIT0	Legge l'indirizzo base della Working Zone numero 0 dalla RAM e lo salva in *WZ_0_START*
INIT1	Legge l'indirizzo base della Working Zone numero 1 dalla RAM e lo salva in *WZ_1_START*
INIT2	Legge l'indirizzo base della Working Zone numero 2 dalla RAM e lo salva in *WZ_2_START*
INIT3	Legge l'indirizzo base della Working Zone numero 3 dalla RAM e lo salva in *WZ_3_START*
INIT4	Legge l'indirizzo base della Working Zone numero 4 dalla RAM e lo salva in *WZ_4_START*
INIT5	Legge l'indirizzo base della Working Zone numero 5 dalla RAM e lo salva in *WZ_5_START*
INIT6	Legge l'indirizzo base della Working Zone numero 6 dalla RAM e lo salva in *WZ_6_START*
INIT7	Legge l'indirizzo base della Working Zone numero 7 dalla RAM e lo salva in *WZ_7_START*
END_INIT	Completa il processo di inizializzazione, porta *o_en* a '0' e va nello stato di *RESET*
START	Legge *INPUT* dalla RAM
WZ_CHECK	Controlla se *INPUT* appartiene ad una delle Working Zone. In caso appartenga a una Working Zone, il segnale WZ_BIT viene portato a '1', altrimenti rimane a '0'

WZ_DECIDE	Se *WZ_BIT* è a '0', setta il prossimo stato a NO_WZ_WRITE, altrimenti a WZ_CALC_OFF
WZ_CALC_OFF	Calcola l'offset in codifica *One-Hot* tra la Working Zone individuata in WZ_CHECK e *INPUT*
WZ_WRITE	Scrive su *o_address* l'indirizzo codificato, quindi WZ_BIT
WZ_NUM	WZ_OFF
WZ_DONE	Porta il segnale *o_done* a 1 per permettere alla FSM di tornare nello stato di *RESET*
NO_WZ_WRITE	Scrive su *o_address* l'indirizzo originale con l'aggiunta di WZ_BIT, quindi WZ_BIT
INPUT	
NO_WZ_DONE	Porta il segnale *o_done* a 1 per permettere alla FSM di tornare nello stato di *RESET*

3 Risultati sperimentali

3.1 Report di sintesi

Il componente risulta correttamente sintetizzabile, senza avere nessun inferred latch, con il seguente uso di componenti sull'FPGA target (`xc7a200tfbg484-1`)

- **LUT**: 119 (0.09% del totale)
- **Flip Flops**:120 (0.04% del totale)

3.2 Test benches

Il componente è stato testato sia tramite i test benches forniti, sia tramite diversi test benches creati ad-hoc per verificare il corretto funzionamento del componente in alcuni casi limite.
E' ragionevole ritenere il componente ragionevolmente corretto in behavioural pre-sintesi ed in functional e timing post-sintesi.
I casi di test più notevoli a cui è stato sottoposto il componente sono i seguenti:

- Indirizzo all'interno di una Working Zone
- Indirizzo non all'interno di una Working Zone
- Working Zone limitrofe a partire dall'indirizzo 0
- Impulso di reset asincrono (durata di 1ns), dove il componente registra correttamente il segnale di reset e quindi ritorna allo stato *RESET*
- 15 milioni di test casuali (e conformi alla specifica) generati tramite un semplice script in Python, che hanno permesso di verificare la correttezza del componente oltre ogni ragionevole dubbio.

I test effettuati hanno evidenziato diverse criticità durante la fase di sviluppo, aiutando a raffinare il componente.

4 Conclusioni

Il componente è considerabile corretto alla luce dei test effettuati, risultando correttamente sintetizzabile e comportandosi nel modo aspettato sia nei test pre-sintesi che nei test timing e functional post-sintesi, rispettando dunque la specifica e gli obbietivi fissati all'inzio della fase progettuale.

Riferimenti bibliografici

[1] Specifica progetto reti logiche 2019/2020. https://drive.google.com/file/d/1qPpIJY_B2mjlEAKcyJTtj9rzhGN9Qyb5/view?usp=sharing.

[2] T. Lang E. Musoll and J. Cortadella. Working-zone encoding for reducing the energy in microprocessor address busses, 1998.