Prova Finale di Reti Logiche
Politecnico di Milano

Prova Finale Di Reti Logiche
AA 2019/2020

Daniele Ferrazzo
(matricola: 892478 - codice persona: 10608810)
Andrea Figini
(matricola: 890831 - codice persona: 10574456)

Indice

1. Introduzione

1.1 Scopo del progetto

L'obiettivo del progetto è quello di implementare un componente hardware, descritto in vhdl, che realizzi una codifica degli indirizzi secondo il metodo delle working zone.
Una working zone è definita come un intervallo di indirizzi avente una dimensione fissa e accessibile tramite l'indirizzo base. Questo metotdo consiste nel determinare, dato un bus di indirizzi (working zone) e l'inidirizzo da codificare, se quest'ultimo appartenga o meno ad una working zone, e in caso positivo codificare l'indirizzo.

1.2 Codifica dell'indirizzo

La codifica dell'indirizzo avviene in due modi diversi, a seconda dell'appartenenza o meno ad una worinkg zone.
Se l'indirizzo in ingresso non appartiene a nessuna delle working zone l'indirizzo verrà trasmesso concatenato ad un bit addizionale posto a zero(WZ_BIT & ADDR)

0	ADDR
1 bit	7 bit

Se invece l'indirizzo in ingresso appartiene ad una working zone l'indirizzo verra trasmesso ponendo il primo bit a "1" (WZ_BIT) concatenato al numero della working zone a cui appartiene (WZ_NUM) e alla codifica in one-hot dell'offset rispetto all'indirizzo base della working zone (WZ_OFFSET).

1	WZ_NUM	WZ_OFFSET
1 bit	3 bit	4 bit

1.3 Interfaccia del componente

Il componente realizzato ha la seguente interfaccia:

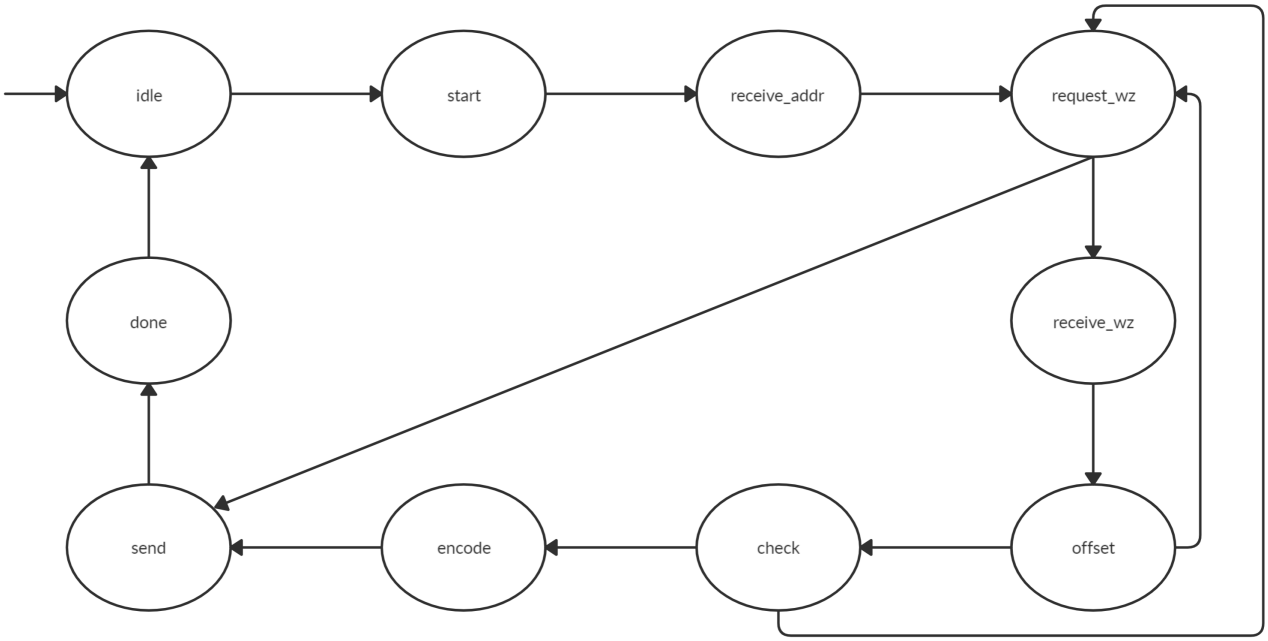
```
entity project_reti _logiche is
port (
     i_clk :  in std_logic;
     i_start :  in std_logic;
     i_rst :  in std_logic;
     i_data :  in std_logic_vector(7 downto 0);
     o_address :  out std_logic_vector(15 downto 0);
     o_done :  out std_logic;
     o_en :  out std_logic;
     o_we : 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

In particolare:

- `i_clk` è il segnale di CLOCK in ingresso generato dal Test Bench;
- `i_start` è il segnale di START generato dal Test Bench;
- `i_rst` è il segnale di RESET che inizializza la macchina pronta per ricevere il primo segnale di START;
- `i_data` è il segnale (vettore) che arriva dalla memoria in seguito ad una richiesta di lettura;
- `o_address` è il segnale (vettore) di uscita che manda l'indirizzo alla memoria;
- `o_done` è il segnale di uscita che comunica la fine dell'elaborazione e il dato di uscita scritto in memoria;
- `o_en` è il segnale di ENABLE che deve essere posto a 1 poer poter comunicare con la memoria (sia in lettura che in scrittura);
- `o_we` è il segnale di WRITE ENABLE che deve essere posto a 1 per poter scrivere nella memora. Per poter leggere da memoria deve invece essere posto a 0;
- `o_data` è il segnale (vettore) di uscita dal componente verso la memoria.

2. Architettura

2.1 Macchina a Stati Finiti



2.2 Stati della macchina

Il componente è stato descritto in VHDL tramite una macchina a stati finiti.La macchina implementata è costituita da dieci stati, come riportato nella figura sopra. Di seguito verrà fornita una breve descrizione della funzione di ciascuno degli stati e delle condizioni di transizione.

2.2.1 IDLE state

Stato di partenza della macchina, che resta in questo stato fino a che il segnale `i_start` non viene posto a 1. La macchina ritorna in questo stato di partenza se il segnale `i_rst` viene posto a 1.

2.2.2 START state

Stato in cui viene richiesto alla ram l'indirizzo da codificare.

2.2.3 RECEIVE_ADDR state

Stato in cui la macchina riceve dalla ram l'indirizzo da codificare.

2.2.4 REQUEST_WZ

Stato in cui viene richiesto alla ram l'indirizzo base delle Working Zone. La macchina si comporta in due modi diversi a seconda del numero di working zone controllate.

- Fino a che la macchina non ha controllato tutte le Working Zone il prossimo stato sarà RECEIVE_WZ.
- Se sono state controllate tutte le working zone,significa che l'indirizzo da codificare non appartiene a nessuna di queste e la macchina passa allo stato SEND.

2.2.5 RECEIVE_WZ state

Stato in cui la macchina riceve l'indirizzo base della Working Zone corrente.

2.2.6 OFFSET state

Stato in cui viene calcolato l'offset fra l'indirizzo da codificare e l'indirizzo dase della Working Zone.

- Se l'indirizzo base della Working Zone risulta maggiore dell'indirizzo da codificare, la macchina ritorna in REQUEST_WZ, in quanto l'indirizzo non appartiene sicuramente alla Working Zone corrente.
- Se l'indirizzo base della Working Zone risulta minore dell'indirizzo da codificare, viene calcolato l'offset e la macchina passa allo stato CHECK.

2.2.7 CHECK state

Stato in cui viene controllato che l'offset calcolato sia minore della dimensione delle Working Zone.

- In caso positivo la macchina passa allo stato ENCODE.
- In caso negativo la macchina ritorna allo stato REQUEST_WZ.

2.2.8 ENCODE state

Stato in cui l'indirizzo appartenente alla Working Zone viene codificato secondo la specifica.

2.2.9 SEND state

Stato in cui viene trasmesso alla ram l'indirizzo codificato.

2.2.10 DONE state

Stato in cui la macchina pone a 1 il segnale `o_done` fino a che il segnale `i_start` non viene posto a 0, a quel punto il segnale `o_done` viene riportato a 0 e la macchina passa allo stato IDLE.

2.3 Process

Il componente è stato descritto facendo uso di quattro process:

- il primo si occupa della corretta inizializzazione e della gestione dei registri utilizzati.
- il secondo si occupa del funzionamento della macchina a stati, genstendo le transizioni da uno stato all'altro.
- il terzo si occupa della codifica dell'indirizzo, sia nel caso in cui questo appartenga ad una working zone sia nel caso contrario.
- il quarto si occupa di calcolare, e eventualmente resettare in caso di multistart, l'indice della working zone corrente.

3. Sintesi

3.1 Synthesis report

Analizzando il "Vivado Synthesis Report" possiamo notare che sono stati sintetizzati i 5 registri usati nel codice (per un totale di 46 Flip Flop utilizzati);

Num. bit	**Num. registri**	**Contenuto del registro**
8	4	Address da codificare; Address Working Zone in input; Codifica temporanea; Offset.
4	1	Indice della Working Zone corrente (Utilizza 4 bit perchè ci sono solamente 8 Working Zone da analizare).

3.2 Report utilization

Analizzando il "Vivado Report Utilization" possiamo vedere l'area occupata dal design sintetizzato. L'obiettivo del gruppo non era quello di ottimizzare l'area occupata del componente quindi questo aspetto è stato lasciato al tool di sintesi.

Risorsa	**Utilizzo**	**Disponibilità**	**Utilizzo in %**
Look Up Table	46	134600	0.03%
Flip Flop	46	269600	0.02%

Nonostante l'obiettivo non fosse quello di ottimizzare l'area del componente i valori di utilizzo sono, in ordini di grandezza, molto minori rispetto alla disponibilità della FPGA.

3.3 Warning

Nella versione finale del componente tutti i warning generati dal tool di sintesi durante lo sviluppo sono stati risolti. Non sono quindi presenti warning "comuni" come warning per inferred latch o warning per segnali non inseriti nella sensitivity list.

4. Test

Per testare il componente realizzato sono state utilizzate varie tipologie di test. Per prima cosa sono stati testati i due testbench forniti insieme alle specifiche del progetto, poi sono stati generati casualemente più di un milione di casi di test, infine sono stati realizzati dei test specifici atti a verificare il corretto funzionamento del componente anche nei casi limite.

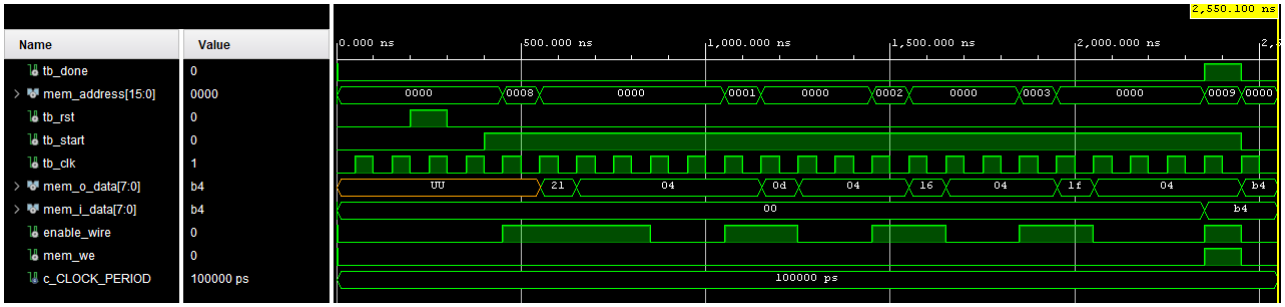
4.1 Test Bench 1 (fornito da specifica)

In questo test viene testato un normale caso di codifica (nessuna condizione limite).
L'indirizzo da codificare è "00100001" (dec: 33) e appartiene alla Working Zone numero 4 (con indirizzo base 31). La maschera di output sarà "10110100" (dec: 180).

4.1.1 Contenuto della RAM

Contenuto	Valore	indirizzo	Comprende Address
WZ 1	4	0	no
WZ 2	13	1	no
WZ 3	22	2	no
WZ 4	31	3	si
WZ 5	37	4	no
WZ 6	45	5	no
WZ 7	77	6	no
WZ 8	91	7	no
ADDRESS da codificare	33	8	no
ADDRESS codificato	180	9	-

4.1.2 Waveform Post-Synthesis Functional Simulation



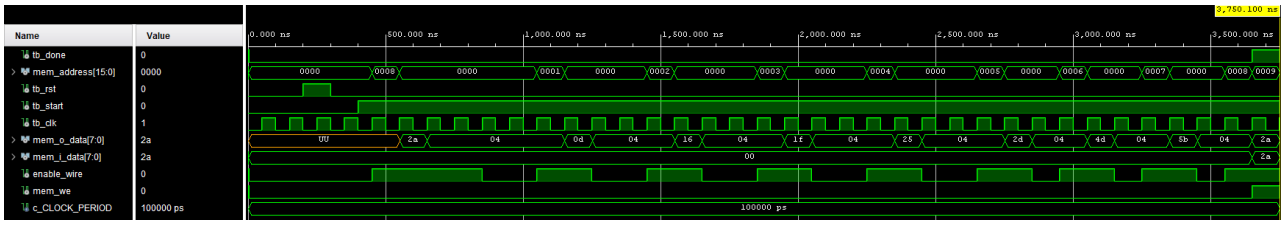
4.2 Test Bench 2 (fornito da specifica)

In questo test viene testato un normale caso di codifca (nessuna condizione limite). L'indirizzo da codifcare è "00101010" (dec: 42) e non appartiene a nessuna working zone. La maschera di output sarà "00101010" (dec: 42).

4.2.1 Contenuto della RAM

Contenuto	Valore	indirizzo	Comprende Address
WZ 1	4	0	no
WZ 2	13	1	no
WZ 3	22	2	no
WZ 4	31	3	no
WZ 5	37	4	no
WZ 6	45	5	no
WZ 7	77	6	no
WZ 8	91	7	no
ADDRESS da codificare	42	8	no
ADDRESS codificato	42	9	-

4.2.2 Waveform Post-Synthesis Functional Simulation



4.3 Random Test

Per verificare la robustezza e affidabilità del componente sono stati generati in modo casuale un numero molto elevato di casi di test, comprendenti casi di multistart e di reset sincroni. L'elevato numero di casi di test generati ci ha permesso di testare il componente nella maggior parte dei casi limite in maniera automatica, permettendoci di scovare alcuni malfunzionamenti in determinati casi particolari, poi testati più approfonditamente e in seguito risolti.
Tutti i test effettuati in questo modo sono stati superati sia in pre-sintesi che in post-sintesi.

4.4 Test mirati

Nonstante l'utilizzo dei test random abbiamo deciso di controllare manualmente alcuni casi limite. Tra i quali:

- indirizzo da codificare `input_addr` ="00000000" ovvero il minimo indirizzo che il componente può codificare.
- indirizzo da codificare `input_addr` ="01111111" ovvero il massimo indirizzo che il componente può codificare.
- indirizzo da codificare con valore pari all'indirizzo base della working zone.
- indirizzo da codificare con valore appena precedente all'indirizzo base di una working zone
- indirizzo da codificare con valore appena successivo all'ultimo indirizzo disposinbile di una working zone.

5. Conclusioni

Il componente realizzato si comporta come da specifica sia in simulazione Behavioral che in simulazione Post-Synthesis-Functional, con un periodo di clock di 100ns.
Inoltre il codice scritto permette di configurare a proprio piacimento le dimensioni delle Working zone, il numero di working zone e la dimensione degli indirizzi attraverso un package di costanti inserito nel codice VHDL.

I tempi di simulazione del caso pessimo e ottimo sono i seguenti:

- Caso pessimo: l'indirizzo da codificare non appariene a nessuna working zone, il componente quindi dovrà effettuare nove letture dalla RAM (l'indirizzo da codificare e le otto working zone).
  tempo = **4050ns**.
- Caso ottimo: l'indirirzzo da codificare appartiene alla prima working zone, il compoente quindi dovrà effettuare solamente due letture dalla RAM (l'indirirzzo da codificare e la prima working zone).
  Tempo = **1350ns**.