

PROVA FINALE DI RETI LOGICHE

Aldo Plenzich (Codice persona: 10538310 – Matricola: 868058)

Mattia Sala (Codice persona: 10540582 – Matricola: 869766)

INDICE ANALITICO

1 INTRODUZIONE

- 1.1 - RICHIESTA PROGETTUALE
- 1.2 - SPECIFICHE PROGETTUALI
- 1.3 - INTERFACCIA DEL COMPONENTE
- 1.4 - ESEMPIO DI FUNZIONAMENTO

2 SCELTE PROGETTUALI

- 2.1 - FUNZIONAMENTO GENERALE
- 2.2 - LISTA DEGLI STATI
- 2.3 - MACCHINA A STATI

3 RISULTATI Sperimentali

- 3.1 - TEST BENCH
- 3.2 - SINTESI

4 CONCLUSIONI

1 - INTRODUZIONE

1.1 - RICHIESTA PROGETTUALE

Progettare e implementare un modulo in grado di leggere indirizzi dalla memoria RAM e tradurli ispirandosi al metodo di codifica basato su “Working-Zone”.

Il modulo dovrà essere in grado di leggere indirizzi direttamente dalla memoria, valutare la loro presenza all'interno delle working zone e, restituire in output un nuovo indirizzo codificato secondo le specifiche progettuali.

1.2 - SPECIFICHE PROGETTUALI

Gli indirizzi delle Working-Zone (WZ) sono memorizzati nelle celle 0-7 della memoria che rappresentano il rispettivo numero. Ciascuna Working Zone occupa l'indirizzo base salvato + 3 indirizzi successivi.

L'indirizzo da codificare verrà letto dalla cella 8 della memoria. Qui si potranno distinguere due casi:

Caso 1: L'indirizzo da codificare non appartiene a nessuna Working Zone, sarà compito del modulo trasmetterlo così com'è, mettendo il bit più significativo (MSB) a 0.

Indirizzo Memoria	Valore	Commento
0	4	// Indirizzo Base WZ 0
1	13	// Indirizzo Base WZ 1
2	22	// Indirizzo Base WZ 2
3	31	// Indirizzo Base WZ 3
4	37	// Indirizzo Base WZ 4
5	45	// Indirizzo Base WZ 5
6	77	// Indirizzo Base WZ 6
7	91	// Indirizzo Base WZ 7
8	42	// ADDR da codificare
9	42	// Valore codificato con in OUTPUT

Caso 1 con valore non presente in nessuna working-zone

Caso 2: l'indirizzo fa parte di una working zone. Il bit addizionale (MSB) è posto a 1, mentre i bit di indirizzo vengono divisi in 2 sottocampi rappresentanti:

- Il numero della working-zone al quale l'indirizzo appartiene, che sarà codificato in binario (bit 6-4).
 - L' offset rispetto all' indirizzo di base della working, codificato come one-hot (bit 3-0)
- Di seguito è riportata la codifica one-hot da utilizzare per gli offset:

0 -> 0001 1-> 0010 2 -> 0100 3 -> 1000

L'indirizzo elaborato viene infine trasmesso alla memoria e memorizzato nella cella 9.

Indirizzo Memoria	Valore	Commento
0	4	// Indirizzo Base WZ 0
1	13	// Indirizzo Base WZ 1
2	22	// Indirizzo Base WZ 2
3	31	// Indirizzo Base WZ 3
4	37	// Indirizzo Base WZ 4
5	45	// Indirizzo Base WZ 5
6	77	// Indirizzo Base WZ 6
7	91	// Indirizzo Base WZ 7
8	33	// ADDR da codificare
9	180	// Valore codificato con in OUTPUT (1 - 011 - 0100)

Caso 2 con valore presente in working-zone

1.3 - INTERFACCIA DEL COMPONENTE

L'interfaccia fornita è la seguente:

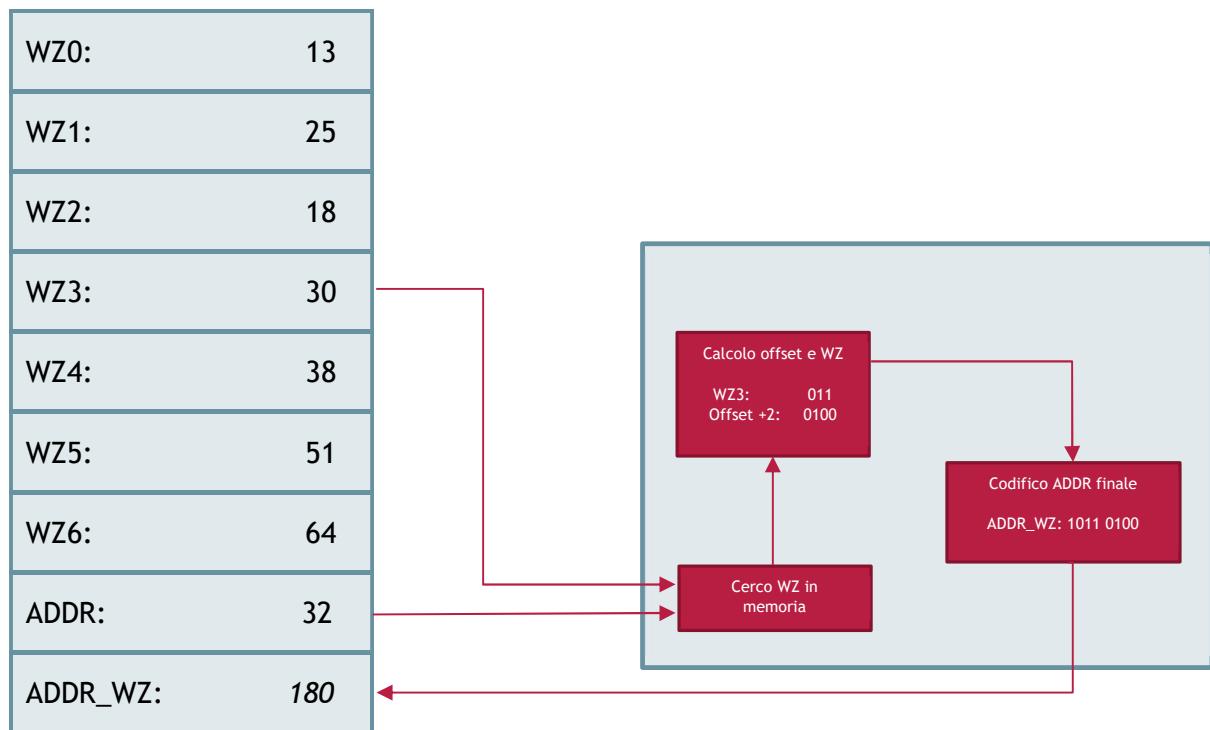
```
entity project_reti_logiche is
    port (
        i_clk : in std_logic;
        i_start : in std_logic;
        i_RST : in std_logic;
        i_data : in std_logic_vector(7 downto 0);
        o_address : out std_logic_vector(15 downto 0);
        o_done : out std_logic;
        o_en : out std_logic;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

In particolare:

- **i_clk**: 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 da dover mandare alla memoria per poter comunicare (sia in lettura che in scrittura);
- **o_we**: è il segnale di WRITE ENABLE da dover mandare alla memoria (=1) per poter scriverci. Per leggere da memoria esso deve essere 0;
- **o_data**: è il segnale (vettore) di uscita dal componente verso la memoria.

1.4 - ESEMPIO DI FUNZIONAMENTO



2- SCELTE PROGETTUALI

2.1 FUNZIONAMENTO GENERALE

Quando il segnale di **i_start** viene portato a 1, la macchina dallo stato iniziale INIT inizia l'elaborazione spostandosi sullo stato START. Qui richiede la lettura alla memoria, della cella numero 8 (ovvero l'indirizzo da codificare) e ne attende la ricezione. Dopo aver ricevuto l'indirizzo da codificare, la macchina entra nello stato CHECK_POS in cui controlla se la prossima cella di memoria da leggere appartiene all'insieme 0-7; ovvero quelle aree di memoria in cui sono salvati gli indirizzi base delle varie working zone. Questo stato controlla se il modulo ha letto tutte le celle contenenti gli indirizzi delle working zone. Nel caso in cui si è iterato su ognuna di esse, senza aver trovato una Working Zone associata all'indirizzo da codificare, si entrerà nello stato MISS_WZ; altrimenti se ci sono altre celle da iterare, il prossimo stato sarà SUB_POS. Qui viene sottratto 1 dall'indirizzo della cella corrente e impostato come prossimo stato: "READ_NEW_POS", che si occuperà di leggere la cella al nuovo indirizzo appena calcolato. Dopo aver atteso la risposta della memoria, lo stato CHECK_WZ controlla se l'indirizzo da codificare appartiene alla working zone appena ricevuta in input dal modulo. In caso negativo la macchina si sposterà sullo stato CHECK_POS per una nuova iterazione, altrimenti lo stato HIT_WZ si occuperà di codificare l'indirizzo - secondo le specifiche di progetto - che dallo stato SEND_ADDRESS verrà inviato alla cella 9 della memoria.

Nel componente inoltre è presente un ingresso di reset **i_rst** che funziona in modo asincrono. Qualora esso venga portato ad 1, la macchina dallo stato corrente passa immediatamente allo stato INIT, interrompendo l'elaborazione e pronta a ricevere un nuovo segnale di **i_start**.

2.2 LISTA DEGLI STATI

INIT – Stato di inizializzazione, imposta a 0 i segnali enable e write enable, inoltre attende l'arrivo del segnale di start per cominciare il processo.

START – È il primo stato del processo in cui viene inviata alla memoria la richiesta di lettura della cella 8, ovvero l'indirizzo da codificare. Inoltre, viene impostato **o_en** a **1** e mantenuto **o_we** a **0** per informare che si vuole leggere.

WAIT_READ - Questo stato viene utilizzato dal modulo per attendere la risposta della memoria.

READ_DATA - Il segnale in ingresso nel modulo, contenente l'indirizzo da codificare viene associato ad un segnale interno (address) e memorizzato per gli stati successivi.

CHECK_POS – Questo stato controlla se l'indirizzo da leggere della prossima cella di memoria è valido, ovvero se è compreso tra la cella 0 e la 7. Se l'iterazione di questo stato porterà a considerare una cella di memoria fuori dall'insieme sopra citato, allora non esiste una working zone associata all'indirizzo da codificare, e quindi verrà considerato come prossimo stato MISS_WZ.

SUB_POS – Viene sottratto 1 all’ultimo indirizzo utilizzato per leggere la cella di memoria.

READ_NEW_POS – Viene letto l’indirizzo della Working zone presente nella cella all’indirizzo calcolato dallo stato precedente.

WAIT_MEM_2 – Questo stato viene utilizzato dal modulo per attendere la risposta della memoria.

CHECK_WZ - Viene effettuato un controllo, se l’indirizzo da codificare è associato alla Working zone appena letta, si entrerà nello stato HIT_WZ, altrimenti il modulo reitererà sulla memoria tornando allo stato CHECK_POS.

HIT_WZ – Il modulo calcola l’indirizzo finale da trasmettere alla memoria rispettando le richieste della specifica nel caso in cui esso sia associato ad una working zone.

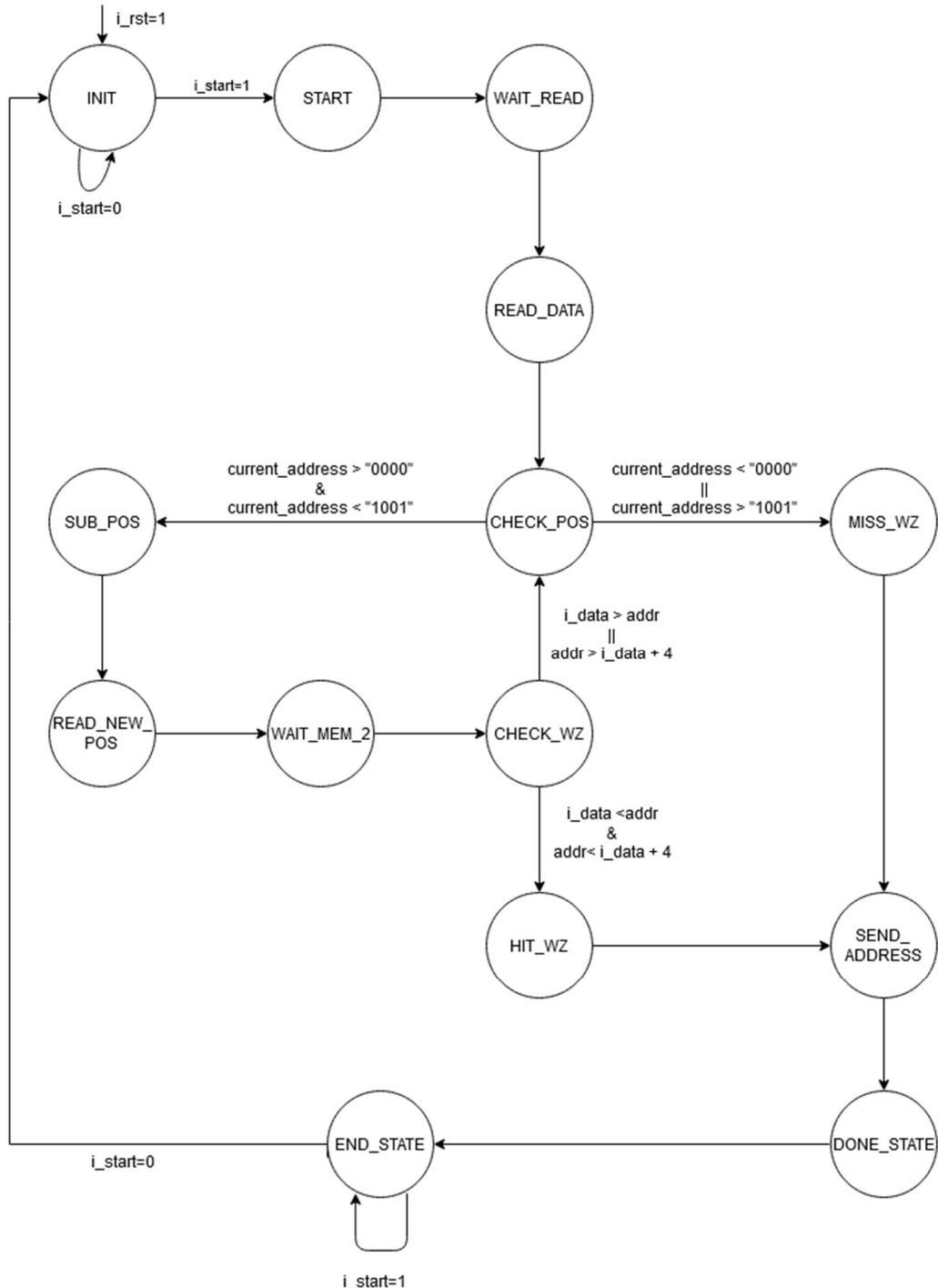
MISS_WZ - Il modulo calcola l’indirizzo finale da trasmettere alla memoria rispettando le richieste della specifica nel caso in cui esso non sia associato ad una working zone.

SEND_ADDRESS – Viene inviato alla memoria l’indirizzo codificato e salvato nella cella 9.

DONE_STATE – Stato in cui viene impostato il segnale **N_o_done** a 1.

END_STATE – Stato finale in cui si attende che il segnale **i_start** venga posto a 0. Dopodiché, si tornerà allo stato iniziale (INIT).

2.3 MACCHINA A STATI E FUNZIONAMENTO



Quando il segnale di start viene portato a 1, la macchina dallo stato iniziale INIT inizia l'elaborazione spostandosi sullo stato START. Qui richiede la lettura, alla memoria, della cella numero 8 (ovvero l'indirizzo da codificare) e ne attende la ricezione. Dopo aver ricevuto l'indirizzo da codificare la macchina entra nello stato CHECK_POS in cui controlla se la prossima cella di memoria da leggere appartiene all'insieme 0-7, ovvero quelle aree di memoria in cui sono salvati gli indirizzi base delle varie working zone. Questo stato controlla se il modulo ha letto tutte le celle contenenti gli indirizzi delle

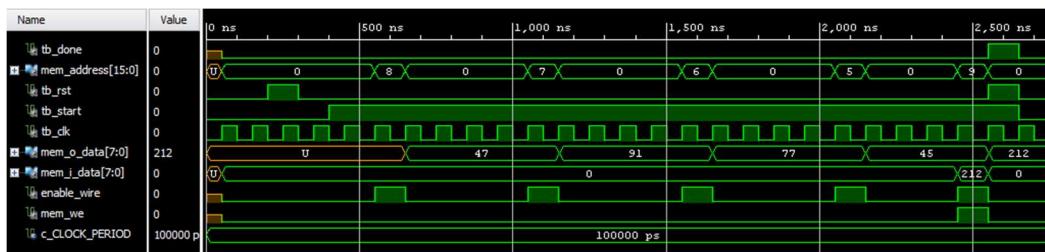
working zone. Nel caso in cui si è iterato su ognuna di esse, senza aver trovato una Working Zone associata all'indirizzo da codificare, si entrerà nello stato MISS_WZ. Altrimenti se ci sono altre celle da iterare, il prossimo stato sarà SUB_POS. Qui viene sottratto 1 dall'indirizzo della cella corrente e impostato next state a READ_NEW_POS che si occuperà di leggere la cella al nuovo indirizzo appena calcolato. Dopo aver atteso la risposta della memoria lo stato CHECK_WZ controlla se l'indirizzo da codificare appartiene alla working zone appena ricevuta in input dal modulo. In caso negativo la macchina si sposterà sullo stato CHECK_POS per una nuova iterazione, altrimenti lo stato HIT_WZ si occuperà di codificare l'indirizzo secondo le specifiche di progetto che dallo stato SEND_ADDRESS verrà inviato alla cella 9 della memoria. Infine, gli ultimi due stati, si occupano di settare il segnale **o_done=1** e aspettare che **i_start=0**, per riportare la macchina allo stato iniziale.

3 - RISULTATI Sperimentali

3.1 Test Bench

Durante l'esecuzione di alcuni test bench, si sono riscontrati i seguenti problemi:

- 1) Test bench 1: “IN WZ” – In questo caso di test, l'indirizzo da codificare appartiene ad una working zone.



- 2) Test bench 2: “NO WZ” – In questo caso di test viene testato cosa accade nel caso l'indirizzo da codificare non appartenga a nessuna working zone.
- 3) Test bench 3: “RESET ASINCRONO” – Test bench in cui il modulo riceve un reset asincrono durante l'esecuzione, che ne interrompe il processo e riporta allo stato INIT.
- 4) Test bench 4: “MULTI-RESET” – Come il test bench 3, ma il modulo riceve più di un reset in fasi diverse del processo.
- 5) Test bench 5: “LIMIT-ADDR” – Vengono testati indirizzi “limite” da codificare tra quelli consentiti con valori appartenenti e non alle working zone. (Indirizzi da 0 a 127)
- 6) Test bench 6: “MULTI-START” – Test bench in cui vengono eseguiti più cicli della macchina a stati.

Problema riscontrato:

Durante l'esecuzione del Test bench 6, si è riscontrato che veniva letto un valore errato dalla memoria e, inoltre, la richiesta di lettura della cella ‘8’ veniva ignorata. Il motivo per cui ciò avveniva, era che la macchina iniziava un nuovo ciclo, dopo aver impostato **o_done=1**, senza attendere che il segnale di start venisse prima impostato a 0 e poi a 1.



La soluzione adottata è stata di utilizzare un nuovo stato **END_STATE**, in cui si aspetta che il segnale **i_start** venga posto a 0, prima di tornare allo stato iniziale (**INIT**).

```
when END_STATE =>
    if i_start = '0' then
        NS <= INIT;
    else
        NS <= END_STATE;
    end if;
```

3.2 - SINTESI

Il codice viene sintetizzato senza errori, permettendo di eseguire timing e functional simulation correttamente. Dai report si osserva l'utilizzo da parte del componente di 35 FF e 65 LUTs.

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	65	0	134600	0.05
LUT as Logic	65	0	134600	0.05
LUT as Memory	0	0	46200	0.00
Slice Registers	35	0	269200	0.01
Register as Flip Flop	35	0	269200	0.01
Register as Latch	0	0	269200	0.00
F7 Muxes	0	0	67300	0.00
F8 Muxes	0	0	33650	0.00

Tempi d'esecuzione:

Tempo esecuzione casi particolari (*post synthesys functional simulation*):

- Caso indirizzo non appartenente a nessuna working zone: 5250 ns
- Caso indirizzo appartenente a ultima cella working zone: 5350 ns

Tempo esecuzione casi particolari (*post synthesys timing simulation*):

- Caso indirizzo non appartenente a nessuna working zone: 5253737 ps
- Caso indirizzo appartenente a ultima cella working zone: 5353737 ps

4 - CONCLUSIONI

In conclusione, l'obiettivo posto dalla specifica era quello di realizzare un componente che simulasse la codifica di un indirizzo col il metodo delle “Working Zone”. Il tutto è stato ottenuto progettando una macchina a stati, in grado di individuare la presenza o meno dell'indirizzo da codificare, e di ricalcolarlo secondo le specifiche fornite.

La macchina a stati è stata poi tradotta in codice VHDL e rielaborata qualora fosse stato necessario. Il codice è poi stato testato, rielaborato e ottimizzato per la sintesi. Infine, sono state effettuate le simulazioni post-sintesi, tramite una batteria di test bench, da cui se n'è ricavata un'ottimizzazione generale del componente.