Prova Finale Progetto Reti Logiche

**Anno Accademico 2019-2020**

Giusti Leonardo (Codice Persona: 10633778)

La Manna Giuseppe (Codice Persona: 10608466)

**INDICE**

1. **Introduzione . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2**

1. **Architettura . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3**

**2.1. *Schema progettuale***

***2.1.1. Interfaccia del componente***

***2.1.2 Segnali utilizzati***

***2.1.3 Macchina a Stati Finiti***

**2.2. *Scelte progettuali***

1. **Risultati di test funzionali . . . . . . . . . . . . . . . . . . . . . . . . . . 4**
2. **Risultati test non funzionali . . . . . . . . . . . . . . . . . . . . . . . . . 5**

1. **Risultati della sintesi . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6**

**1. Introduzione**

**L’obbiettivo del progetto è creare il metodo di codifica Working Zone pensato per trasformare il valore di un indirizzo trasmesso, se questo appartiene a certi intervalli denominati *working-zone.***

**Il codice si occuperà di leggere l’indirizzo da codificare (ADDR) e gli 8 indirizzi base delle Working Zone, aventi dimensione di 4 indirizzi incluso quello base, confrontarli e produrre l’indirizzo opportunamente codificato.**

**Se l’indirizzo da trasmettere non appartiene a nessuna Working Zone, ai bit di indirizzamento verrà concatenato un bit addizionale WZ\_BIT messo a 0:**

**WZ(ADDR) = WZ\_BIT & ADDR**

Se l’indirizzo da trasmettere appartiene ad una Working Zone, verrà trasmesso WZ\_BIT posto a 1 concatenato a WZ\_NUM, che rappresenta la codifica in binario del numero della working-zone al quale l’indirizzo appartiene, e WZ\_OFFSET, che rappresenta lo spiazzamento in codifica one-hot:

**WZ(ADDR) = WZ\_BIT & WZ\_NUM & WZ\_OFFSET**

Nel seguente esempio viene esposto approfonditamente il metodo di codifica:

Immagine che contiene disegnando

Descrizione generata automaticamente

Notiamo che non vi è alcun ordine tra gli address di base delle working zones quindi ottimizzazioni del componente basate su questo non sono possibili.

Mostriamo di seguito la codifica di due address

* *Address1* da codificare: 68 (01000100)
* *Address2* da codificare: 15 (00001111)

Immagine che contiene screenshot

Descrizione generata automaticamente

**L’*Address 1* non appartiene ad alcuna Working Zone quindi avrà WZ\_BIT = 0 ed i successivi bit invariati**

**L’*Address 2* appartiene alla Working Zone 1 con WZ\_OFFSET = 1. WZ\_NUM corrisponde alla codifica binaria di 2 su 3 bit (001) invece il WZ\_OFFSET sarà la codifica one hot di 1 su 4 bit (0010).**

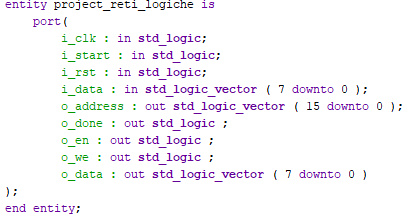
**Il valore codificato avrà come output (1 – 001 – 0010), che corrisponde al valore 146.**

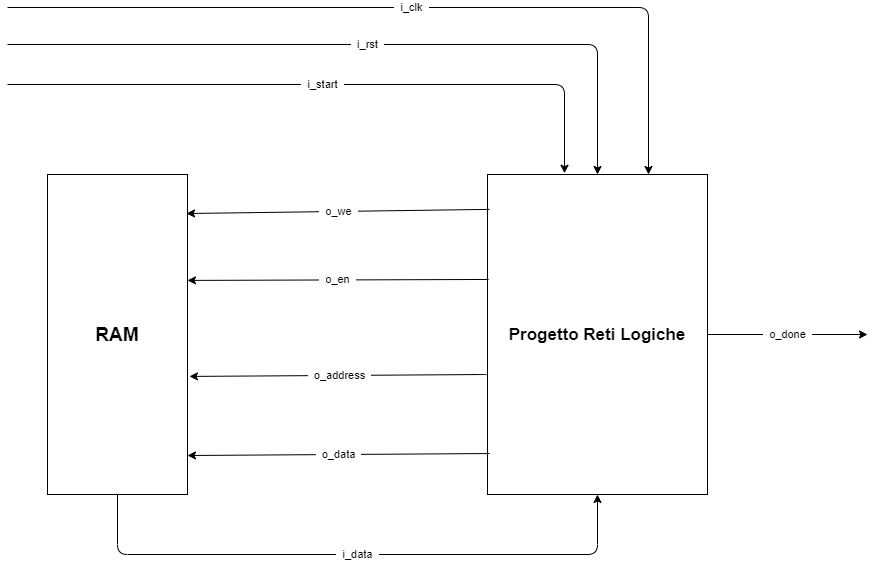
**2. Architettura**

**2.2 Schema progettuale**

**2.2.1. Interfaccia del componente**

**Il componente è collegato ad una memoria RAM tramite la seguente interfaccia:**

****



Schema dell'interfaccia del componente con la memoria RAM

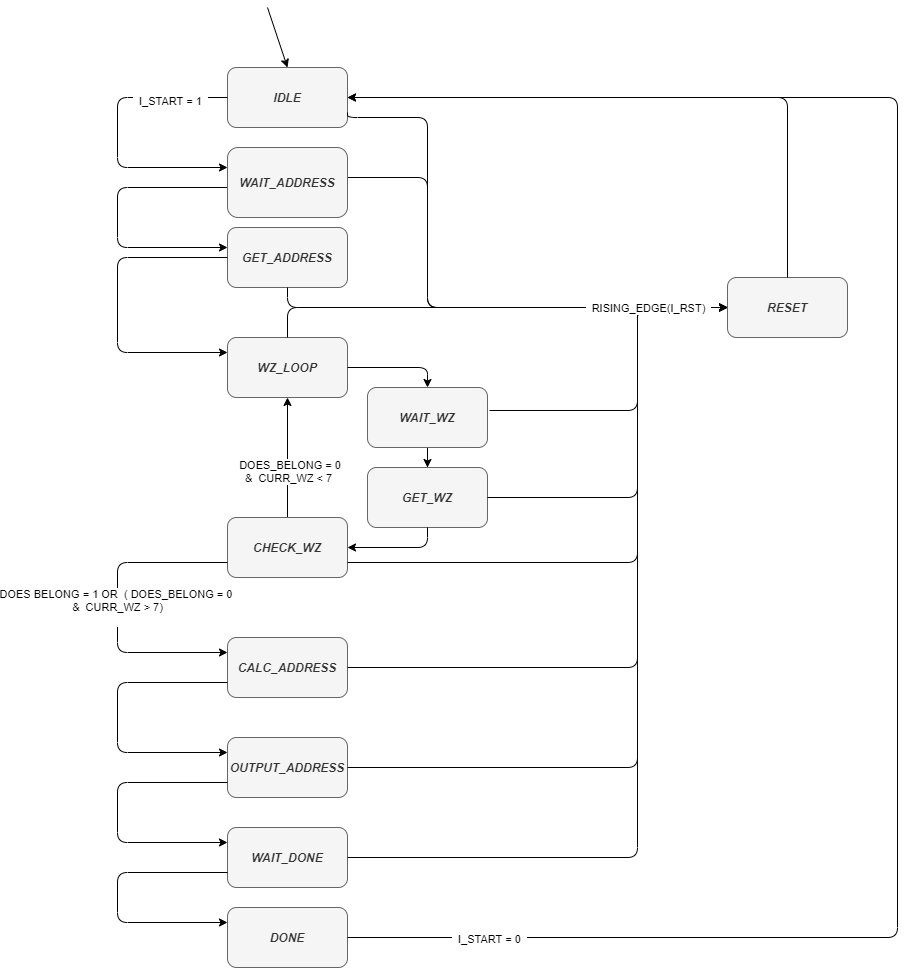
**2.2.2. Segnali utilizzati**

**La seguente tabella riassuntiva rappresenta i segnali introdotti per la realizzazione del modulo gestiti all’interno degli stati della FSM. Per ogni segnale presente in tabella ne è stato inserito uno aggiuntivo (signal\_name\_next) nel quale viene memorizzato il valore del segnale al ciclo di clock successivo.**

|  |  |
| --- | --- |
| **Nome Segnale** | **Contenuto** |
| **current\_state** | **Memorizza lo stato corrente della FSM** |
| **current\_wz** | **Memorizza l’indirizzo base della Working Zone corrente** |
| **read\_address** | Memorizza il valore dell’indirizzo da codificare ricevuto dalla RAM |
| **does\_belong** | **Specifica se l’indirizzo da codificare appartiene o meno alla Working Zone corrente** |
| **wz\_num** | **Numero intero che indica quale tra le 8 working zone sia caricata in quel momento** |
| **coded\_address** | Contiene l’indirizzo finale da restituire |
| **need\_rst** | Booleano che indica se il componente deve essere inizializzato |

**2.2.3. Macchina a Stati Finiti**

Il funzionamento alla base del componente è stato implementato attraverso una FSM che usa *i\_start* come segnale di avvio della codifica e *i\_rst* come segnale per l’inizializzazione della macchina. Il passaggio tra gli stati successivi sarà approfondito in seguito alla figura dello schema.



* **idle: s**tato iniziale in cui la macchina attende un segnale di *i\_start* per iniziare la sua computazione: una volta ricevuto la macchina richiede l'*address* da codificare alla RAM;
* **wait\_address:** stato in cui attendo che la RAM presenti in *i\_data* l'*address* da codificare;
* **wz\_loop:** stato in cui richiedo alla RAM l'address base della Working Zone corrente, inizialmente carico WZ 0;
* **get\_wz:** stato in cui salvo l'address base della Working Zone corrente presentatomi dalla RAM su *i\_data;*
* **check\_wz:** stato in cui verifico se l'*address* da codificare appartiene alla Working Zone corrente o meno: se appartiene mi preparo a codificarlo, in caso contrario passo alla Working Zone successiva; se sono all'ultima Working Zone allora l'*address* non appartiene ad alcuna di queste e mi preparo a codificarlo di conseguenza;
* **calc\_address:** stato in cui calcolo l'*address* codificato tenendo in considerazione il numero della Working Zone di appartenenza e l'offset dell'*address* da codificare;
* **output\_address:** stato in cui richiedo alla RAM di poter scrivere l'*address* codificato;
* **wait\_done:** stato in cui attendo che l’*address* codificato sia stato effettivamente scritto in memoria;
* **done:** stato di completamento in cui attendo che il segnale di *i\_start* scenda a 0 per tornare in Idle e poter continuare a codificare l'*address* successivo;
* **Reset:** stato in cui azzero il valore di ogni segnale del componente in seguito ad una richiesta di *i\_rst*, per tornare poi allo stato di Idle e aspettare la codifica successiva;

**3. Scelte progettuali**

Per il design del componente si è deciso di usare un modello behavioural con tre processi :

* **reset\_change** : processo sensibile al segnale di i\_rst, è incaricato di rilevare ogni istanza di reset che arriva in qualsiasi momento al sistema
* **state\_change** : processo sensibile al segnale di i\_clk, è incaricato di aggiornare i valori dei registri usati dal componente ad ogni ciclo di clock; in presenza di una richiesta di reset si occupa di portare la FSM allo stato di reset
* **lambda** : processo sensibile ai segnali del componente, contiene le operazioni che vengono eseguite al fine di codificare correttamente un address

**Un’altra decisione critica per il design del componente riguarda la modalità di caricamento degli address base delle working zone: invece di caricare tutti gli address all’inizio della computazione e di doverli ricaricare in seguito ad ogni reset si è deciso di caricare individualmente le working zones in base alle nostre esigenze.**

**Questo approccio permette di caricare solo le working zones necessarie: nella computazione di un address appartenente alla prima WZ sarà caricata solamente la quest’ultima, la FSM andrà direttamente allo stato di calc\_address permettendo quindi una conversione veloce ; al contrario nel caso di un address che non appartiene ad alcuna WZ sarà necessario caricare e confrontare l’address con ogni address base di WZ con una conversione più lenta.**

**Inoltre non è necessaria alcuna operazione addizionale nel momento in cui viene effettuato il reset del componente**

**4. Risultati di test funzionali**

**5. Risultati di test non funzionali**

**6. Risultati della sintesi**