Prova finale (Progetto di Reti Logiche)

Anno Accademico 2018/2019

Professore: William Fornaciari

Tutor: Davide Zoni

Federico Dei Cas [10498291] [845783]

Alessandro Contini [10538919] [870498]

Indice

1) Introduzione

1.1 Specifica

L'obiettivo del progetto è quello di descrivere e sintetizzare tramite VHDL un componente hardware che implementi la funzionalità di calcolare quali sono i centroidi a distanza minima dei punti dati rispetto a un punto di riferimento su una matrice quadrata 256x256, tramite l'algoritmo della distanza di Manhattan. Inoltre è presente una maschera di ingresso a 8 bit che specifica per quali centroidi è effettivamente necessario considerare la distanza. Tale maschera è posta a '1', a partire dal bit meno significativo, per indicare che il primo centroide va esaminato, altrimenti è posta a '0'. I dati in ingresso vengono letti da una memoria RAM con indirizzamento al byte.

L'interfaccia del componente è la seguente:

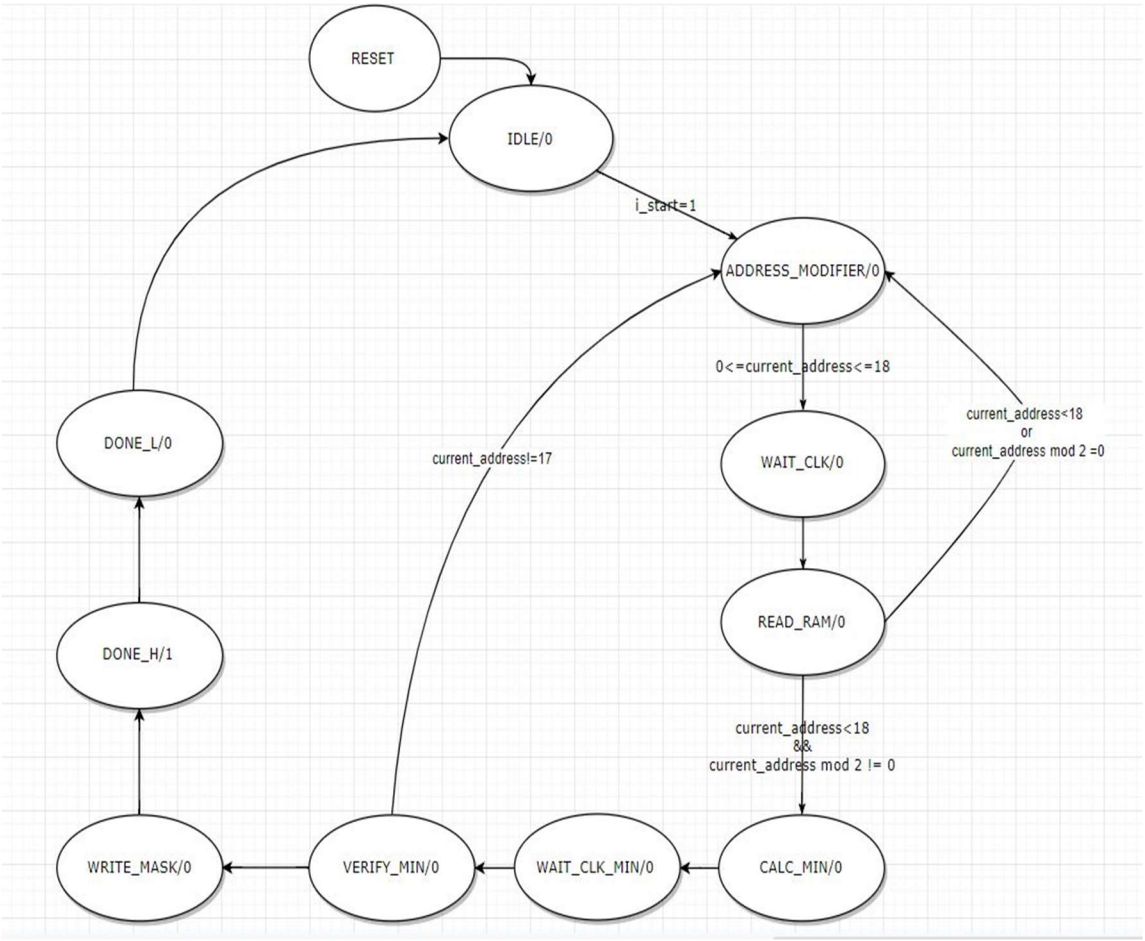
```
entity project_reti_logiche is
        port (
        i_clk : in std_logic;
        i_start : in std_logic;
        i_rst : in std_logic;
        i_data : in std_logic_vector(7 downto 0);
        o_address : out std_logic_vector(15 downto 0);
        o_done : out std_logic;
        o_en : out std_logic;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

1.2 Scelte implementative

Un'iniziale analisi del problema permette di constatare, vista anche la presenza dei due segnali in ingresso di start e reset, che l'implementazione preveda la necessità di realizzare un automa a stati finiti. La FSM volge all'utilizzo di un numero relativamente elevato di stati di attesa (stati del tipo WAIT_* nel modello di Moore sotto presentato) che non richiedono input, ma sono necessari alla corretta sincronizzazione della macchina per le operazioni di lettura. Ad ogni ciclo di clock viene inoltre controllato lo stato del segnale ***i_rst***, poiché il suo valore portato alto segnala la necessità di riportare la macchina nello stato iniziale.

2) Modello FSM

2.1 Diagramma degli stati



[Figura 1]

La rappresentazione è del tipo Moore, con l'output che rappresenta il segnale ***o_done***. In fase di progettazione abbiamo constatato la necessità di aggiungere alcuni stati, riconoscibili dal prefisso WAIT, che sono stati aggiunti alla FSM per permettere la sincronizzazione dei valori in lettura. I valori contenuti nella RAM, infatti, necessitano di un ciclo di clock da quando vengono richiesti a quando sono trasmessi nella porta di ingresso ***i_data***. Questi stati non fanno altro che riassegnare i valori dei segnali precedenti (al fine di evitare l'introduzione di Latch durante la sintesi) e aspettare il ciclo di clock necessario a ricevere i dati.

L'idea alla base del nostro algoritmo è quella di leggere in sequenza crescente i valori delle coordinate dei centroidi, partendo dall' indirizzo della **X** del punto da valutare e successivamente la sua **Y**. Dopo aver letto le coordinate del centroide da cui calcolare la distanza, l'indirizzo corrente viene posto all'indirizzo 0 della RAM, dove viene letta la maschera di ingresso con i centroidi da valutare e da scartare, per poi valutare singolarmente ogni punto dei centroidi di cui si deve trovare la distanza. Dopo aver letto la **X** del generico punto, e una volta letta la sua **Y**, si procede al calcolo della distanza di queste ultime coordinate dal punto prescelto. Infine, se necessario, viene costruita la maschera di uscita temporanea, che potrà essere successivamente modificata o rimanere invariata fino al termine delle valutazioni.

2.2 Analisi degli stati

Di seguito viene riportata una breve descrizione di ogni stato, per giusticare le decisioni prese in fase di design della FSM e rendere più chiaro il funzionamento del nostro modello.

2.2.1 Stato IDLE

Stato iniziale, durante il quale la macchina resta in attesa del segnale di start (i_start='1'). Una volta recepito tale input vengono inizializzati/reinizializzati i segnali ai valori iniziali. Inoltre ogni segnale di reset in arrivo al componente, in qualsiasi momento, riporta l'automa in questo stato, grazie a un controllo che viene effettuato sul fronte di salita di ogni ciclo di clock.

2.2.2 Stato ADDRESS_MODIFIER

Serve per gestire i segnali che memorizzano l'indirizzo corrente della RAM.

2.2.3 Stato WAIT_CLK

Come precedentemente riportato, questo stato permette di attendere un ciclo di clock, per far si che i segnali vengano aggiornati correttamente.

2.2.4 Stato READ_RAM

Serve per salvare i dati in arrivo dall'indirizzo attuale della RAM. Il comportamento dello stato dipende dall'indirizzo correntemente considerato (che viene gestito da ADDRESS_MODIFIER). Lo stato descritto legge quindi i valori richiesti nella RAM e li salva temporaneamente, per poi fare le dovute valutazioni sulla distanza.

2.2.5 Stato WAIT_CLK_MIN

Attende un ciclo di clock per permettere di sincronizzare i dati in arrivo dalla RAM con i segnali utilizzati, per salvare temporaneamente i valori.

2.2.6 Stato CALC_MIN

Calcola la distanza di Manhattan tra i punti attuali e il punto di riferimento.

2.2.7 Stato VERIFY_MIN

Aggiorna il valore della distanza minima nel caso in cui l'elemento sia da considerare.
Innanzitutto analizza i valori della maschera in ingresso, se l'elemento della maschera che indica questo punto è 1 valida la distanza . Se tale distanza è inferiore della minore trovata finora, azzera la maschera d'uscita, se è maggiore non altera il risultato.

2.2.8 Stato WRITE_MASK

Scrive nell'indirizzo di memoria 19 della RAM il risultato della computazione ed aspetta un ciclo per aver il valore della maschera disponibile alla lettura del testbench.

2.2.9 Stati DONE_H e DONE_L

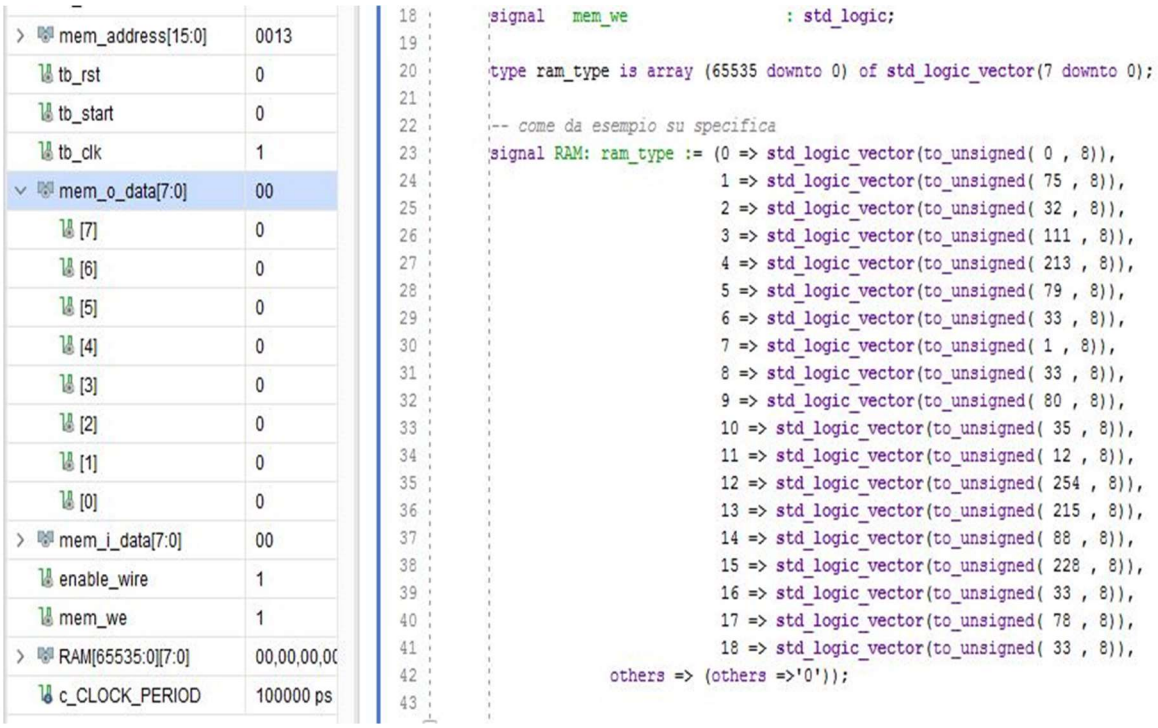
Il primo porta ad HIGH il segnale di ***o_done***, segnalando la fine dell'elaborazione.
Il secondo lo riporta ad un livello logico basso e permette alla macchina di tornare allo stato iniziale, in una configurazione di attesa, pronta per una nuova elaborazione.

3) Casi di test

Per verificare l'effettivo funzionamento del componente, abbiamo generato dei casi di test che riteniamo significativi e che permettano di valutare il comportamento atteso della nostra implementazione nella maggior parte dei casi possibili. Tra i testbench generati, alcuni di essi meritano di essere riportati e descritti in modo più esaustivo.

3.1 Maschera di ingresso posta a "00000000"

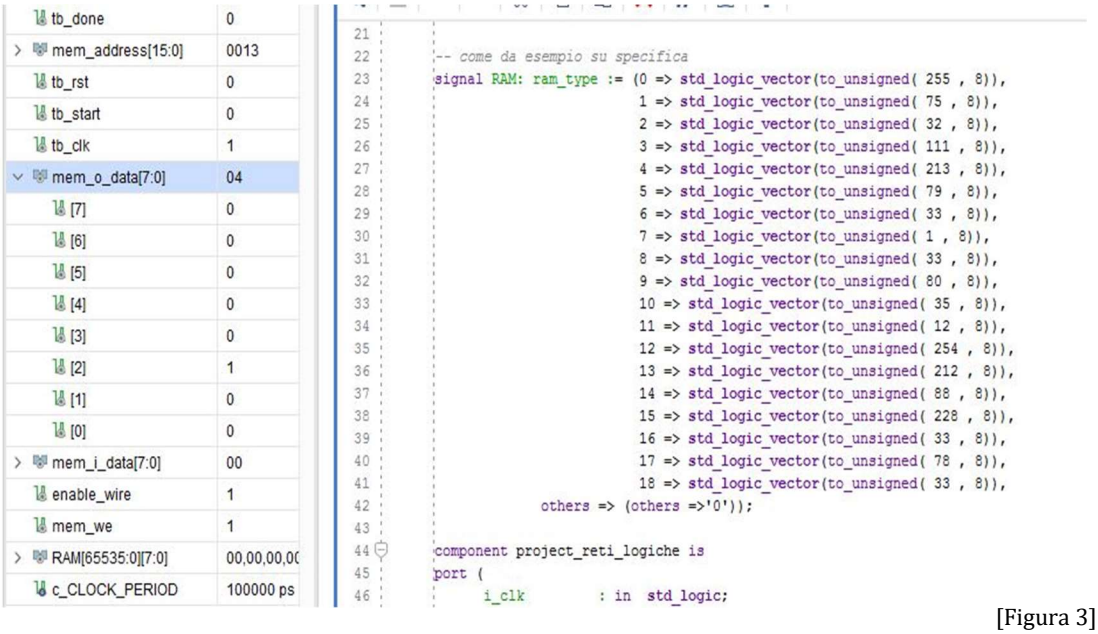
In questo caso ci aspettiamo che la maschera d'uscita sia "00000000", in quanto nessun valore deve essere tenuto in considerazione.
Si osservi come l'uscita all'interno di ***mem_o_data*** sia "00000000". Abbiamo ritenuto questo caso di test significativo perché permette di verificare che i controlli sui bit della maschera di ingresso funzionino e che tutti i segnali vengano inizializzati correttamente.



[Figura 2]

3.2 Maschera di ingresso posta a “111111111”

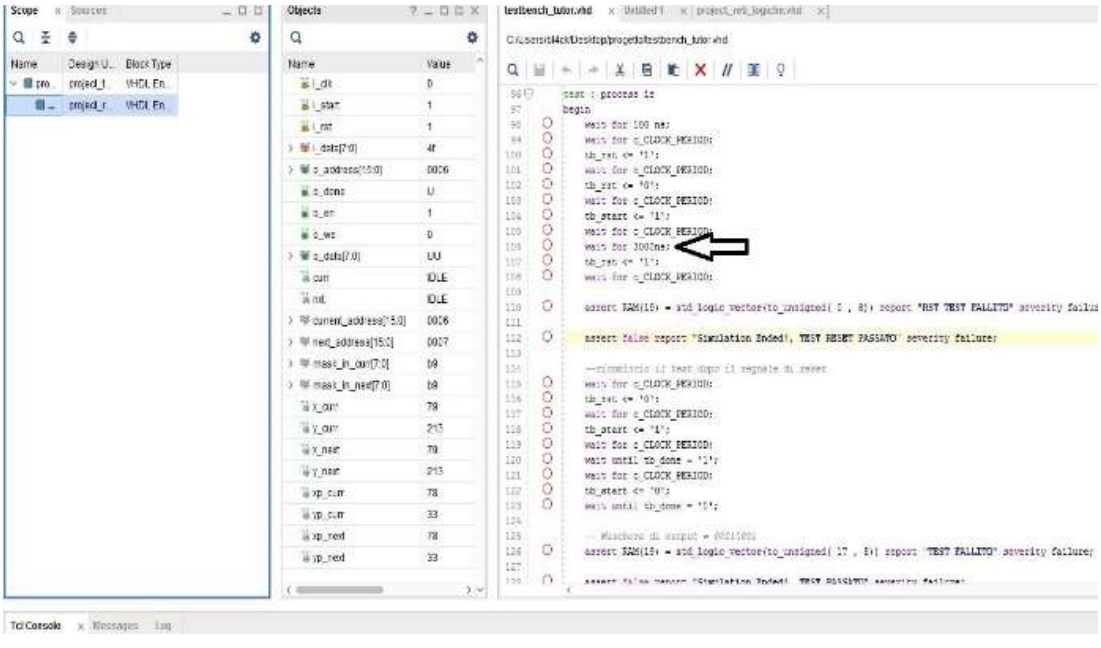
Con lo stesso input del test precedente vediamo il caso in cui vengono accettati tutti gli input possibili. Saranno i punti più vicini in assoluto a portare in alto la maschera di uscita. Si osserva come il terzo punto (indirizzi 5 e 6) sia il più vicino al punto di riferimento (indirizzi 17 e 18) in quanto la loro distanza di Manhattan è unitaria .



[Figura 3]

3.3 Ricezione di un segnale di reset improvviso

Un altro caso che merita di essere riportato, è la ricezione di un improvviso segnale di reset (i_rst = ‘1’) che può essere ricevuto in qualsiasi momento e deve riportare la macchina allo stato IDLE, ripristinando le condizioni iniziali. Si osserva che al ciclo successivo di clock i segnali **xp_curr** e **yp_curr**, cioè le coordinate del punto dal quale si deve calcolare la distanza, vengono reinizializzate dopo che **i_rst** è stato posto a ‘1’ e la macchina torna nello stato IDLE, pronta per una nuova computazione.



[Figura 4]

4) Ottimizzazioni

Una possibile ottimizzazione potrebbe essere quella di gestire il calcolo della distanza con un if-case, anziché utilizzare la funzione *abs* fornita da VHDL per calcolare la distanza del punto attualmente considerato rispetto a quello scelto come riferimento. L'if case, grazie a un controllo sul segno delle coordinate del centroide attualmente valutato e quello di riferimento, potrebbe permettere di calcolare le distanze senza l'ausilio della funzione *abs*, riducendo i ritardi introdotti da questa funzione.