POLITECNICO DI MILANO

SCUOLA DI INGEGNERIA INDUSTRIALE E DELL'INFORMAZIONE
Corso di Laurea in Ingegneria Informatica

Prova Finale di Reti Logiche

Anno Accademico 2018–2019

Professore:
William Fornaciari

Tutor:
Davide Zoni

Canditati:

Roberto Buratti	Matricola: 869112	Codice Persona: 10577247
Alessio Bersani	Matricola: 867660	Codice Persona: 10520128

Documentazione

Si vuole implementare un componente hardware descritto in VHDL in grado di valutare la minima distanza (Manhattan distance) tra un punto ed N centroidi tutti definiti in uno spazio quadrato bidimensionale 256 x 256. Degli N centriodi, K (con K minore o uguale ad N) sono quelli di cui valutare la distanza del punto dato e vengono definiti dalla maschera di ingresso ad N bit: il bit a 1 indica che il centroide è valido, in caso contrario non deve essere esaminato. La maschera di uscita è sempre a N bit e che i bit a 1 saranno non più di K.
Le coordinate dei vari centroidi sono salvate in memoria e la vicinanza viene espressa tramite una maschera di bit (maschera di uscita) ognuno dei quali corrisponde ad un centroide. Il bit viene posto a 1 se il centroide corrispondente è il più vicino al punto fornito, 0 altrimenti; nel caso in cui il punto considerato risulti equidistante da più centroidi, i bit della maschera d'uscita relativi ad essi saranno tutti impostati ad 1.

Per il suo completamento si è deciso di utilizzare la FSM rappresentata in Figura 1.

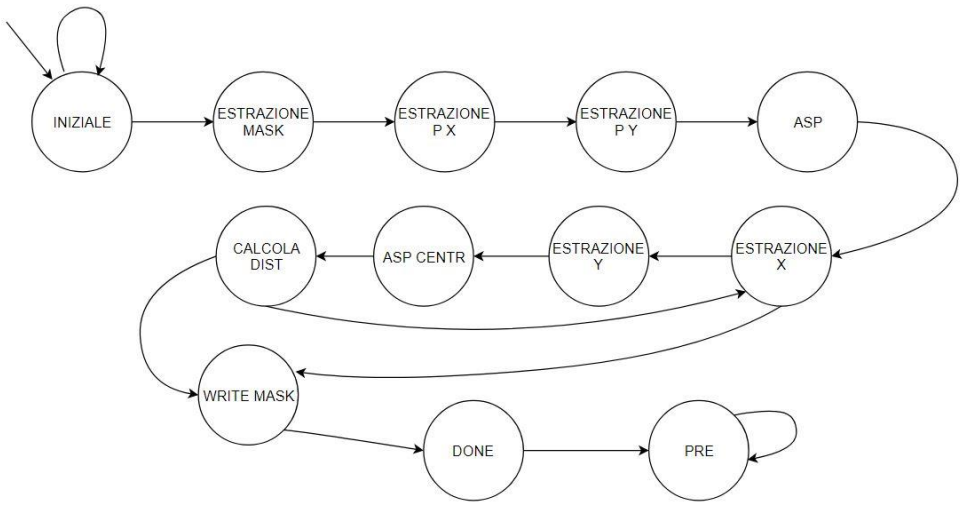

Figura 1: Diagramma degli stati della macchina a stati finiti implementata.

Descrizione dei vari stati:

INIZIALE: è lo stato iniziale della macchina. Quando al componente arriva il segnale i_start pari a 1 la macchina passa allo stato ESTRAZIONE_MASK.

ESTRAZIONE_MASK: il componente invia la richiesta di lettura della maschera in input alla RAM. La macchina si sposta successivamente nello stato ESTRAZIONE_P_X.

ESTRAZIONE_P_X: la maschera ricevuta in input viene salvata in un registro. Viene inviata alla RAM la richiesta per l'estrazione della coordinata X del punto da valutare. Si passa allo stato ESTRAZIONE_P_Y.

ESTRAZIONE_P_Y: la coordinata X ricevuta viene salvata in un ulteriore registro. Si entra nello stato ASP.

ASP: si aspetta un ciclo di clock in modo da garantire al componente il tempo per salvare in un altro registro la coordinata Y del punto. In seguito inizia lo stato ESTRAZIONE_X.

ESTRAZIONE_X: si occupa dell'estrazione della coordinata X degli i-esimi centroidi definiti dalla maschera ricevuta in ingresso. Se il valore dell'i-esimo bit della maschera è pari a 0 e se 'i' è pari a 7, allora si passa allo stato di WRITE_MASK, altrimenti 'i' viene incrementato. Invece nel caso in cui l'i-esimo bit della maschera è pari a 1, viene richiesta alla RAM la coordinata X del punto preso in esame. Si passa successivamente allo stato ESTRAZIONE_Y.

ESTRAZIONE_Y: viene salvato nel registro designato la coordinata X dell'i-esimo centroide e viene richiesta la coordinata Y del punto alla RAM. La macchina si sposta nello stato ASP_CENTR.

ASP_CENTR: si aspetta un ciclo di clock per poter salvare anche la coordinata Y del centroide nell'apposito registro. Si entra nello stato CALCOLA_DIST.

CALCOLA_DIST: calcola la distanza di Manhattan, ne salva la minima, incrementa 'i' e se questo è pari a 7, allora si passa allo stato WRITE_MASK altrimenti si torna ad ESTRAZIONE_X.

WRITE_MASK: scrive la maschera d'uscita sulla RAM. Si prosegue con lo stato di DONE.

DONE: segnala la fine del processo settando a 1 il segnale o_done. Segue lo stato PRE.

PRE: svolge la funzione di stato pozzo.

Testing:

I seguenti test sono stati effettuati sia in Pre-sintesi che in Post-sintesi.

1. **Testbench fornito come esempio;**

2. **Maschera in input pari a 00000000;**
   Obiettivo: verificare il corretto funzionamento degli stati ESTRAZIONE_X e WRITE_MASK quando non vengono forniti centroidi validi al componente.

3. **Maschera in input pari a 11111111 con tutti i centroidi coincidenti;**
   Obiettivo: verificare il corretto funzionamento di CALCOLA_DIST con i centroidi equidistanti dal punto di riferimento.

4. **Maschera in input pari a 00000011 con le coordinate fornite in Figura 2**:

```
signal RAM: ram_type := (0 => std_logic_vector(to_unsigned( 3 , 8)),
                         1 => std_logic_vector(to_unsigned( 75 , 8)),
                         2 => std_logic_vector(to_unsigned( 32 , 8)),
                         3 => std_logic_vector(to_unsigned( 111 , 8)),
                         4 => std_logic_vector(to_unsigned( 213 , 8)),
                         5 => std_logic_vector(to_unsigned( 79 , 8)),
                         6 => std_logic_vector(to_unsigned( 33 , 8)),
                         7 => std_logic_vector(to_unsigned( 1 , 8)),
                         8 => std_logic_vector(to_unsigned( 33 , 8)),
                         9 => std_logic_vector(to_unsigned( 80 , 8)),
                         10 => std_logic_vector(to_unsigned( 35 , 8)),
                         11 => std_logic_vector(to_unsigned( 12 , 8)),
                         12 => std_logic_vector(to_unsigned( 254 , 8)),
                         13 => std_logic_vector(to_unsigned( 215 , 8)),
                         14 => std_logic_vector(to_unsigned( 78 , 8)),
                         15 => std_logic_vector(to_unsigned( 211 , 8)),
                         16 => std_logic_vector(to_unsigned( 121 , 8)),
                         17 => std_logic_vector(to_unsigned( 75 , 8)),
                         18 => std_logic_vector(to_unsigned( 32 , 8)),
                         others => (others =>'0'));
```

Figura 2: specifiche del quarto caso di test

Obiettivo: verificare il valido funzionamento di CALCOLA_DIST nel caso in cui il punto preso in analisi sia sovrapposto ad uno dei due centroidi da analizzare.

5. **Maschera in input pari a 00000001 con le coordinate fornite in Figura 3**:

```
signal RAM: ram_type := (0 => std_logic_vector(to_unsigned( 1 , 8)),
                         1 => std_logic_vector(to_unsigned( 255 , 8)),
                         2 => std_logic_vector(to_unsigned( 255 , 8)),
                         3 => std_logic_vector(to_unsigned( 111 , 8)),
                         4 => std_logic_vector(to_unsigned( 213 , 8)),
                         5 => std_logic_vector(to_unsigned( 79 , 8)),
                         6 => std_logic_vector(to_unsigned( 33 , 8)),
                         7 => std_logic_vector(to_unsigned( 1 , 8)),
                         8 => std_logic_vector(to_unsigned( 33 , 8)),
                         9 => std_logic_vector(to_unsigned( 80 , 8)),
                         10 => std_logic_vector(to_unsigned( 35 , 8)),
                         11 => std_logic_vector(to_unsigned( 12 , 8)),
                         12 => std_logic_vector(to_unsigned( 254 , 8)),
                         13 => std_logic_vector(to_unsigned( 215 , 8)),
                         14 => std_logic_vector(to_unsigned( 78 , 8)),
                         15 => std_logic_vector(to_unsigned( 211 , 8)),
                         16 => std_logic_vector(to_unsigned( 121 , 8)),
                         17 => std_logic_vector(to_unsigned( 0 , 8)),
                         18 => std_logic_vector(to_unsigned( 0 , 8)),
                     others => (others =>'0'));
```

Figura 3: specifiche del quinto caso di test

Obiettivo: verificare il regolare funzionamento di CALCOLA_DIST nel caso in cui il punto preso in analisi sia alla distanza massima con l'unico centroide valido.

Warnings:

Il codice presenta in totale 12 messaggi di warning poiché nonostante la porta sia da 16 bit per le scelte progettuali il componente utilizza i cinque meno significativi.

Ottimizzazioni:

Non si individuano eventuali ottimizzazioni significative.