Progetto finale di Reti Logiche

Roland Reylander [10539438]

1 Aprile 2019

Indice

1 Specifiche di progetto

Lo scopo del progetto è la realizzazione di un componente hardware usando il linguaggio VHDL che risolve il seguente problema: in uno spazio quadrato di 256x256 vengono posizionati 8 centroidi e si vuole trovare il centroide o i centroidi pi vicini ad un punto dato.
Il componente dovrà interfacciarsi con una memoria RAM dalla quale legge i dati di input e scrive il risultato della computazione. La memoria RAM di input contiene la maschera dei centroidi da considerare, le coordinate dei centroidi, le coordinate del punto dal quale calcolare la distanza e un inidirizzo dedicato al risultato, cio la maschera di uscita.

INDIRIZZO RAM	CONTENUTO
0	maschera dei centroidi
1	coordinata X del centroide
2	coordinata Y del centroide
⋮	⋮
17	coordinata X del punto da considerare
18	coordinata Y del punto da considerare
19	maschera di output del risultato
⋮	indirizzi non utili

2 Scelte progettuali

Per affrontare la risoluzione di questo problema è stata pensata una FSM con i seguenti stati:

- **reset**: tutti i signal vengono inizializzati ad un valore di inizio;
- **changeAddress**: in base al valore che assume `cnt` (cioè `readMask, readXPoint, readYPoint, readXCoord, readYCoord`) viene aumentato l'indirizzo per la prossima lettura della RAM;
- **waitClock**: la lettura dalla memoria richiede 2 ns e quindi non è immediata. In questo stato si aspetta un ciclo di clock affinché il valore `i_data` assuma il valore desiderato che verrà poi usato nello stato successivo;
- **readData**: in base al valore di `cnt` in questo stato vengono salvati su dei signal dedicati i valori della maschera e delle coordinate del punto. Successivamente verranno usati i signal `xAddress` e `yAddress` per memorizzare le coordinate dei centroidi;
- **calcDistance**: viene calcolata la distanza e memorizzata in `tempDistance`;
- **compareDistance**: la distanza calcolata al punto precedente viene confrontata con `bestDistance` in modo da tenere solo i centroidi più vicini;

- **sendMask**: `t_data` assume il valore del risultato finale e l'indirizzo da mandare alla RAM il 19 e `o_we` viene settato a 1 per la scrittura del risultato;
- **load**: segnale di `o_done` viene messo a 1 per indicare la fine dell'elaborazione e l'avvenuta scrittura del risultato;
- **last**: `o_done` torna a 0 e il componente torna allo stato di `reset` pronto per ricevere il prossimo segnale di start.

Dopo la lettura della maschera dalla RAM, nello stato `changeAddress` viene scandita la maschera dei centroidi da considerare tramite il signal `maskPos` in modo da passare alla lettura delle coordinate dei centroidi in caso il bit sia 1 altrimenti verrà letto il prossimo bit della maschera.
I signal `tempDistance` e `bestDistance` sono entrambi `std_logic_vector(8 DOWNTO 0)` dato che la distanza massima delle coordinate X e Y dei centroidi è $255 + 255 = 510$ e necessita quindi di 8 bit. Inoltre `bestDistance` viene inizializzata nello stato di *reset* con `OTHERS => '1'` cioè 511 che è maggiore della massima distanza possibile dei centroidi e quindi permette il corretto funzionamento dell'algoritmo.

3 Risultati dei test

4 Risultati della sintesi