Progetto di Reti Logiche

Sarta Carmelo
868241

Ruffati Lorenzo
867257

1 settembre 2019

Anno Accademico 2018/2019
Professore: Palermo Gianluca
Tutor: Davide Conficconi

Indice

1 Specifica di Progetto

Sia dato uno spazio bidimensionale definito in termini di dimensione orizzontale e verticale, e siano date le posizioni di N punti, detti "centroidi", appartenenti a tale spazio. Si vuole implementare un componente HW descritto in VHDL che, una volta fornite le coordinate di un punto (da ora in poi ci riferiremo a questo punto con il denominativo *target*) appartenente a tale spazio, sia in grado di valutare a quale/i dei centroidi risulti più vicino. La distanza viene calcolata attraverso la definizione di Manhattan Distance:

$$L_1(P_1, P_2) = |x_1 - x_2| + |y_1 - y_2|$$

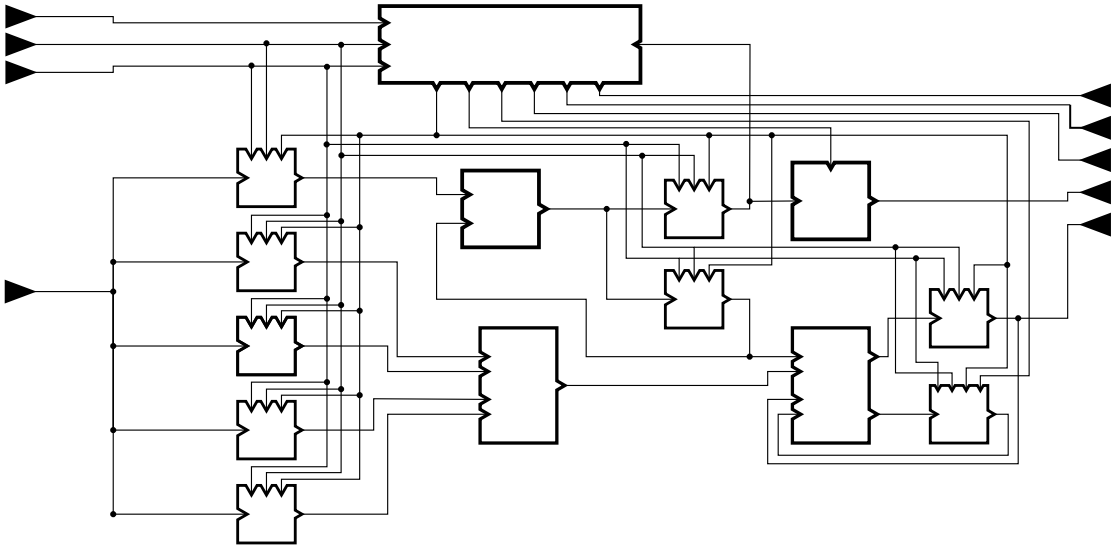
2 Implementazione

La rete necessita di una memoria RAM di 19 byte dove:

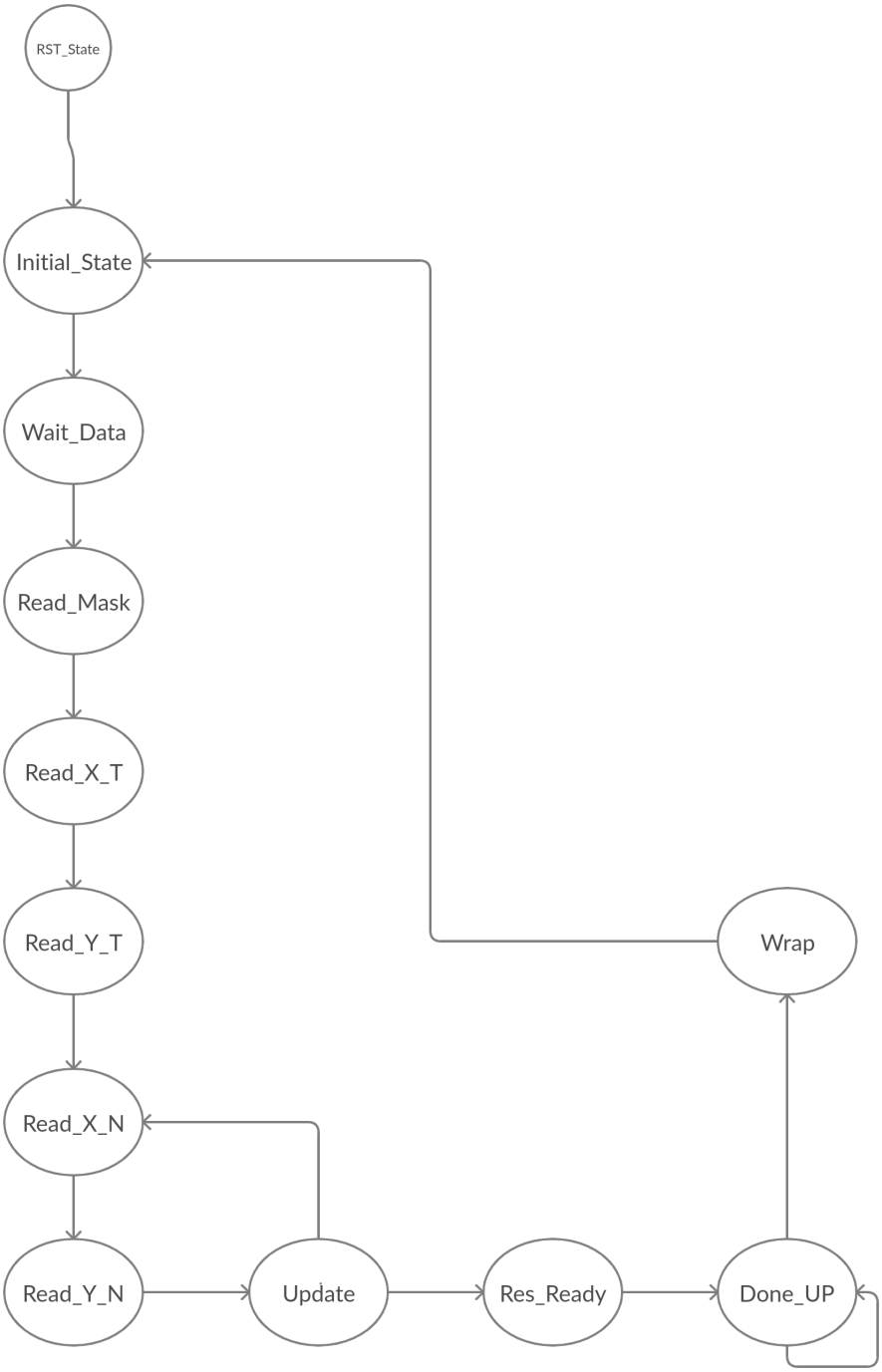
- Il byte in posizione zero viene utilizzato per ottenere il numero di centroidi ed indicarci quali di quelli salvati nei seguenti byte devono essere effettivamente comparati.
- I byte delle posizioni comprese tra uno e sedici vengono utilizzati per memorizzare, a due a due, le coordinate X e Y di ogni centroide.
- Nei byte in posizione diciassette e diciotto sono salvate le coordinate del target.
- L'ultimo byte viene utilizzato per scrivere la maschera di input in cui verranno indicati tutti i centroidi alla minima distanza dal target.

Si sfrutta un clock con un periodo di **100ns** e viene utilizzata la **FPGA xc7a200tfbg484-1**. Abbiamo scelto di descrivere esplicitamente le singole componenti in modo puramente comportamentale (behavioral). In questo modo ci è stato possibile testare individualmente tutti i componenti e siamo riusciti a evitare la comparazione con centroidi non evidenziati nella maschera del primo byte in memori.
Tra le componenti utilizzate abbiamo un Controller che si occupa di gestire l'ordine di funzionamento delle componenti restanti. Il tutto può essere schematizzato come una *macchina a stati finiti*.
Gli schemi del Circuito e della FSM sono visualizzabili nelle pagine seguenti con una breve descrizione per ogni stato.
Utilizziamo otto flip flop a otto bit ed uno a nove bit, attivandoli attraverso un segnale di enable specifico per ogni flip flop. Utilizziamo inoltre un enable della memoria per procedere alla manipolazione dei dati sulla memoria e un enable di write per permettere la sola scrittura sulla memoria.

3 Circuito



4 Macchina a Stati



- **RST_State:** Stato raggiunto esclusivamente in seguito alla ricezione del segnale di *reset*. Ogni componente viene riportato ad uno stato base iniziale, i flip flop saranno portati a otto zeri eslcuso il flip flop a nove bit di minima distanza che verrà riportato a 0x1fs. Viene azzerato l'enable di ogni flip flop, l'enable del write e l'enable della memoria, oltre al segnale di *done*. Si procede quindi allo stato *Initial_State*.

- **Initial_State:** Viene atteso il segnale di *start* per procedere allo stato *Wait_Data*.

- **Wait_Data:** Viene richiesto l'indirizzo della maschera alla memroia e si procede allo stato *Read_Mask*.

- **Read_Mask:** L'indirizzo in output viene portato all'indirizzo diciassette corrispondente alla coordinata X del target. Si procede allo stato *Read_X_T*.

- **Read_X_T:** Viene attivato l'enable del flip flop X_T in modo da memorizzare la coordinata X del target e viene fornita alla memoria la coordinata Y del medesimo punto. Viene inoltre attivato l'enable del flip flop *NXTMSK* e si procede allo stato *Read_Y_T*.

- **Read_Y_T:** Viene attivato l'enable del flip flop Y_T in modo da memorizzare la coordinata Y del target e viene fornita alla memoria la coordinata X del prossimo centroide ricavato dal flip flop *NXTMSK*. Viene inoltre attivato l'enable del flip flop *CRNMSK*, contenente lo stesso centroide ricavato da *NXTMSK*, e si procede allo stato *Read_X_N*.

- **Read_X_N:** Viene attivato l'enable del flip flop X_N in modo da memorizzare la coordinata X del centroide identificato da *CRNMSK*, viene attivato l'enable del flip flop *NXTMSK* (il cui contenuto non cambierà fino al prossimo ciclo di salita) e viene fornita alla memoria la coordinata Y. Si procede allo stato *Read_Y_N*.

- **Read_Y_N:** Viene attivato l'enable del flip flop Y_N in modo da memorizzare la coordinata Y del centroide identificato da *CRNMSK* e viene fornita alla memoria la coordinata X del prossimo centroide identificato da *NXTMSK*. Si procede allo stato *Update*.

- **Update:** Vengono attivati gli enable dei flip flop *MSKRES* e *MINDST* in modo da aggiornare la distanza minima dal target ottenuta finora e la maschera di output rappresentante quei centroidi. Viene anche attivato l'enable di *CRNMSK* per proseguire l'iterazione dei centroidi. Se non ci sono ulteriori centroidi da considerare il contenuto di *MSKRES* è il risultato da scrivere in memoria, pertanto viene portato a uno l'enable di write scrivendo *o_address* nell'indirizzo diciannove passando infine allo stato *Res_Ready*, altrimenti si prosegue alla lettura del prossimo centroide passando allo stato *Read_X_N*.

- **Res_Ready:** Si abbassa il segnale di enable del write e si porta alto il segnale *cntrl_rst* che si occupa di resettare il flip flop *MINDST* in modo da poter effettuare una nuova operazione senza dover attendere il segnale di reset. Si procede allo stato *Done_UP*.
- **Done_UP:** Si alza il segnale di *done* e si rimane in questo stato finché il segnale di *start* non scenda a zero. Si passa infine allo stato di *Wrap*.
- **Wrap:** Questo stato viene utilizzato per portare a zero ogni output del controller e tornare allo stato iniziale senza dover attendere il segnale di *reset*, come evidenziato in precedenza. Lo stato si differenzia da *RST_State* per i valori contenuti nei flip flop.

5 Ottimizzazione & Testing

Per ottimizzare l'esecuzione della rete non esistono stati che richiedono le coordinate del punto da *leggere*, bensì la macchina è descritta in modo procedurale e può essere riutilizzata senza dover inviare e ricevere un segnale di reset. Per migliorare i tempi di esecuzione viene sfruttato il ritardo tra la lettura della maschera in ingresso e il calcolo del primo indirizzo per leggere le coordinate del target. Dopo la lettura di ogni centroide attraverso il comparatore otteniamo il risultato ottenuto e aggiornato. Abbiamo comunque evitato l'utilizzo di costrutti aritmetici più complessi, come il prodotto.
Come test notevoli è stata passata una memoria con maschera di centroidi contenente solo zeri, una maschera contenente centroidi equidistanti dal target e centroidi coincidenti con il target. Sono stati inoltre generati dei test casuali attraverso un semplice codice scritta da noi in python. In tutti questi casi la simulazione comportamentale (behavioral pre-sintesi), funzionale e timing (post-sintesi) è stata portata a termine con successo.