A. A. 2018-2019

Prova finale (Progetto di reti logiche)

Politecnico di Milano

Corso di ingegneria informatica

DOCENTE: PALERMO GIANLUCA

TUTOR: CONFICCONI DAVIDE

Tagliabue Daniele	867017	10529737
Valeriani Angelica Sofia	865744	10520299

SOMMARIO

PRESENTAZIONE

Lo scopo del progetto è la descrizione di un componente hardware in VHDL che calcoli in uno spazio bidimensionale limitato quali punti soddisfino il requisito di essere attivi ed avere la minima distanza[1] da un ulteriore punto fornito. Lo spazio viene considerato limitato in quanto considera la sola porzione di superficie quadrata con lati da 256 e un vertice nell'origine così da consentire il completo sviluppo della figura internamente al primo quadrante. Ne consegue quindi che tutti i punti possano avere coordinate cartesiane comprese tra (0;0) e (255;255).

Anche i punti forniti si dimostrano essere limitati poiché vengono assegnate le sole coordinate di otto centroidi, più un nono aggiuntivo rappresentante il punto da cui considerare le distanze, che devono però soddisfare il requisito di essere considerati attivi per essere inclusi nelle operazioni di calcolo delle distanze. Tale condizione è dettata dal fatto che viene fornita una sequenza di 8 bit dove ciascuna cella identifica lo stato di attivazione del centroide che si trova nella medesima posizione, considerando quindi il primo punto come bit più significativo del vettore prima indicato.

L'output dovrà quindi essere trasmesso in modo similare alla maschera di attivazione in quanto i soli punti attivi aventi distanza minima vedranno attribuirsi alla propria cella identificativa nel vettore un valore pari a 1.

Saranno presenti inoltre dei segnali di reset e di start che guideranno le elaborazioni riportandole rispettivamente allo stato iniziale oppure dando l'avvio al componente hardware.

1. Con il termine distanza ci si riferisce alla Manhattan distance ovvero un calcolo che vede come risultato la somma dei valori assoluti delle differenze tra le coordinate di due punti.

SCELTE DI PROGETTO

Per le scelte progettuali si è deciso di creare un primo processo sensibile al reset che, in caso di reset a 1, re-inizializzi sia i segnali di uscita che le variabili interne riportando lo stato attuale all'inizio, ovvero allo stato RESET. Il successivo, sensibile al fronte di salita del clock, è costituito al suo interno da sette case che rappresentano gli stati di esecuzione.

RESET: E' lo stato iniziale, ovvero uno stato in cui si aspetta che il segnale i_rst sia portato ad 1 per andare ad inizializzare sia le variabili che gli output a dei valori predefiniti, in particolar modo o_address conterrà l'indirizzo in cui è salvata la maschera nella RAM, la N verrà portata a 0 e il CS, ovvero lo stato corrente, diventerà START.

Case START: abbassa il segnale di o_done quando o_start assume valore 0, contrariamente, se o_start assumesse valore 1, abilita la lettura e passa ad uno stato di RALLENTA in cui si attende la risposta della RAM.

Case RALLENTA: in base al valore di un indice intero N, che viene incrementato di 1 ogni volta che si esce dal case, sceglie un case in cui direzionare l'esecuzione. Si è deciso di effettuare in questo stato la scelta opportuna della successione di case da eseguire, per frammentare la sequenza e al contempo dare il tempo necessario per la risposta della RAM. Alla prima chiamata RALLENTA seleziona il case READ_MASK. Alla seconda e alla terza (in corrispondenza dei valori di N, 0 e 1) viene selezionato il case READ_C. Alle chiamate successive per scegliere se eseguire il case di lettura della X o della Y si è deciso

di effettuare la distinzione tra le due attraverso l'operazione di resto modulo due come condizione sulla variabile N, in modo tale da eseguire la lettura della X in corrispondenza di valori dispari di N e quella delle Y in corrispondenza dei valori pari di N. Nota importante che la variabile N viene aggiornata dopo aver aggiornato lo stato prossimo.

Case READ_MASK: si occupa della lettura della maschera di input, che viene salvata nella variabile *MASK*. Successivamente lo stato corrente diventa nuovamente RALLENTA.

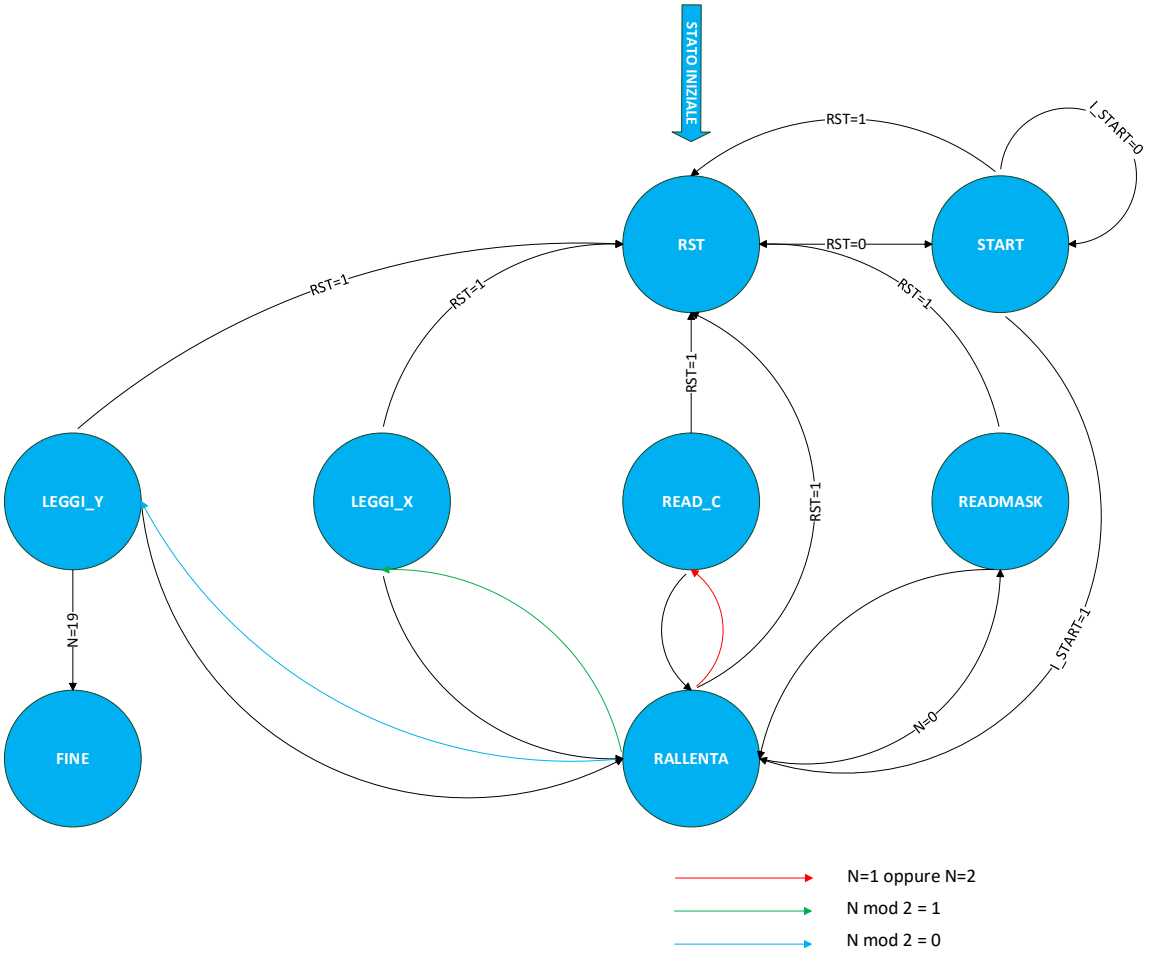
Case READ_C: si occupa della lettura di entrambe le coordinate (X e Y) del punto da valutare, viene eseguito in corrispondenza dei valori N pari a 2 o 3 per distinguere la lettura del valore di X da quella del valore di Y. Una volta entrati nel case, nel caso in cui la N valesse 2 si salva nella variabile X il contenuto della i_data, infatti questa, avendo nello stato READ_MASK posto la o_address all'indirizzo di RAM della coordinata X del punto da valutare ed avendo aspettato un ciclo di clock, conterrà la coordinata richiesta. Nel caso in cui valesse 3, contrariamente, verrebbe salvata la coordinata Y del punto da valutare nella variabile Y. In ognuno di questi casi, l'indirizzo contenuto in o_address viene aggiornato. Al termine viene portato lo stato corrente a RALLENTA.

Case LEGGI_X: si occupa della lettura della X di ogni centroide da confrontare con il punto da valutare, la X di ogni centroide viene di volta in volta salvata in una variabile per calcolare allo step successivo la Manhattan Distance del centroide (si sovrascrive ogni volta l'ascissa del centroide con la precedente, non si usano otto variabili per diminuire il numero di variabili da utilizzare). Dopo aver letto il valore della coordinata X viene confrontato con la X del punto di riferimento in modo da porter effettuare correttamente la sottrazione salvarlo nella variabile DX. Al termine si aggiorna l'indirizzo o_address e si torna allo stato RALLENTA.

Case LEGGI_Y: si occupa di calcolare di volta in volta la distanza del centroide considerato aggiornando qualora necessario il valore del minimo, memorizzato nella variabil MIN, e aggiornando la maschera di output. Dopo aver letto la coordinata Y del punto ed aver calcolato la Manhattan Distance in modo simile al punto precedente, viene fatto un controllo sul fatto che tale punto sia da considerare (usando la maschera di riferimento) e si controlla se sia o meno il nuovo minimo "temporaneo". Ci si trova quindi di fronte a 3 casi, infatti se fosse un nuovo minimo verrebbe azzerata tutta la maschera di uscita (o_data) e posto ad 1 il solo punto appena considerato, oltre ad aggiornare il minimo. Nel caso in cui fosse uguale al punto precedente, allora viene posto ad 1 il punto appena considerato senza però modificare gli altri eventuali 1 presenti in o_data. Nel caso in cui fosse minore del minimo, viene ignorato e o_done resta invariata. Successivamente, nel caso in cui il punto considerato fosse l'ultimo, si provvede a portare o_address all'indirizzo 19 della RAM, abilitare la scrittura e o_done, in questo caso lo stato successivo sarà FINE. In caso contrario verrebbe semplicemente aggiornato il prossimo indirizzo da cui leggere e si provvederebbe a tornare nello stato RALLENTA per attendere la risposta della RAM.

Case FINE: si occupa di terminare l'esecuzione portando a zero il valore di o_done, viene selezionato in corrispondenza del valore N pari a 19 nel case LEGGI_Y.

RAPPRESENTAZIONE GRAFICA



FASE DI TEST

Oltre al testbench di esecuzione fornito come materiale, si è deciso di testare l'esecuzione usando come dati sia casi limite casuali per verificare la correttezza del programma. In particolare, si è deciso di testare i seguenti casi che hanno dato esito positivo:

- Maschera 00000000
- Maschera 11111111
- Centroidi tutti coincidenti
- Punto da valutare con coordinate (0, 0)
- Punto da valutare con coordinate (255, 255)
- Punto da valutare coincidente con uno o più centroidi attivi
- Distanze dal punto da valutare superiori a 255
- Centroide con distanza massima non attivo
- Coordinate dei centroidi sia superiori che inferiori a quelle del punto da valutare

In quest'ultimo caso, in modo particolare, ci è stato utile effettuare il test successivamente alla prima stesura del codice in quanto ci ha permesso di rilevare un errore in corrispondenza del calcolo delle distanze. Si generavano infatti errori andando a considerare distanze negative pur con l'utilizzo del

valore assoluto, motivo per cui si è deciso di effettuare un controllo su quale delle coordinate fosse la maggiore in modo da svolgere la successiva sottrazione senza l'utilizzo del valore assoluto.

Abbiamo inoltre modificato i segnali di reset e start in modo da verificare che questi non andassero a influire sul corretto svolgimento delle elaborazioni, prestando particolare attenzione nel fare in modo che il segnale di reset fosse portato ad 1 sia nel momento esatto in cui si riscontrava un fronte di salita del clock, sia in istanti in cui il clock non variava il suo segnale.

MIGLIORAMENTI E PROPOSTE DI EVOLUZIONE

Come possibile miglioria si era pensato di integrare la lettura delle coordinate del punto da valutare, che nel programma si trova nel case READ_C, all'interno dell'alternanza tra LEGGI_X e LEGGI_Y, ovvero di ricavare un unico stato generale (case LEGGI_X) per la lettura di tutte le ascisse dei punti in input al programma (centroidi e punto da valutare), e una struttura analoga (case LEGGI_Y) per la lettura delle ordinate. Si è deciso tuttavia di mantenere la lettura delle coordinate separate perché l'uso di soli due stati per effettuare la read di tutti i dati avrebbe comportato una struttura molto più articolata per i due case, con diverse ramificazioni if per gestire i vari casi interni, e di conseguenza una minore leggibilità e soprattutto una minor manutenibilità del programma. Oltre a questo, un numero minore di stati avrebbe però provocato un aumento del numero di variabili o segnali da utilizzare.

Avendo ottenuto delle buone performances per tempi di esecuzione e memoria occupata non si sono pensate delle modifiche sostanziali all'algoritmo di esecuzione, ovvero alla vera e propria struttura del programma, ma più che altro a delle piccole variazioni di carattere puramente generale, non rilevanti per il miglioramento dell'efficienza del codice.

È da notare che il tempo di esecuzione non varia al variare del numero di elementi della maschera posti ad 1, contrariamente dipende dal numero di reset. In quest'ottica si sarebbe potuto ottimizzare il progetto con stati aggiuntivi per ottimizzare il tempo di esecuzione in alcuni casi limite nonché per saltare le fasi di lettura delle coordinate dei centroidi non attivi, ovvero con bit della maschera posti a 0. Un esempio di alcuni casi limite che abbiamo considerato sono la maschera di ingresso composta da tutti zeri, a fronte della quale si sarebbe potuto generare nell'immediato un output di tutti zeri. Anche in quest'ambito è stato deciso di evitare tali stati aggiuntivi per aumentare la facilità di manutenzione del codice oltre al fatto che sarebbero stati molto rari casi di miglioramento significativo del tempo di elaborazione.