Prova finale (PROGETTO DI RETI LOGICHE)

Anno accademico 2018-2019

Componenti del gruppo

Tutor

Fabio Salice

Professore

Fabio Salice

**Giuseppe Italia**

869897

10563398

**Gianluca Misiano**

869441

10572478

INDICE

Diagramma a stati finiti ……………………….……………………….……………………….……………………….……………………….………………………. 1

Descrizione funzionamento modulo ……………………….……………………….……………………….……………………….……………………….. 2

Descrizione dei casi di test ………………………..………………………..………………………..………………………..………………………..……………..3

Possibili ottimizzazioni ………………………..………………………..………………………..………………………..………………………..…………………….4

DIAGRAMMA A STATI FINITI

If i!=111

If i<111 and distance>=minDistance

If i<=111 and distance<minDistance

If i=111

If i=111 and distance>=minDistance

If i!=111

If i=111

1

DESCRIZIONE MODULO

L’elaborazione e la struttura del progetto si basa su tre aspetti fondamentali: lettura dei dati in ingresso dalla memoria, analisi degli stessi e infine scrittura sempre in memoria dei diversi valori di output. Il primo step del diagramma degli stati chiamato “Reset” interpreta lo stato di inizio di ogni ciclo della macchina: ad ogni segnale alto di i\_rst verremo catapultati in questo stato inizializzando tutte le variabili in nostro possesso. Non appena il segnale i\_start viene posto ad 1 viene iniziato il ciclo della macchina con la lettura della maschera di ingresso nello stato “GET\_MASK” e le cordinate del punto da valutare negli stati “GET\_CORDX” e “GET\_CORDY”; gli stalli sono necessari per una questione di sincronizzazione di tempo per acquisire da memoria e scrivere su memoria. Arrivati a “REQ-TEMPX” viene avviato un ciclo di 8 fasi(con la variabile i contatore), 8 quanti sono i centroidi da analizzare. Superati gli stati di “REQ-TEMPX” e “REQ-TEMPY” con i quali memorizziamo da memoria le coordinate dei centroidi passiamo a “SELECT\_BIT” iniziando un’analisi attenta dei bit della maschera d’ingresso; in particolare se l’i-esimo bit della maschera d’ingresso(input\_mask) è uguale a 1 allora passiamo allo stato “ASSIGN-DIST” calcolando appunto la distanza tra il punto da valutare e il centroide considerato, altrimenti riiniziamo il ciclo con “REQ-TEMPX” acquisendo le coordinate del centroide seguente; @quest’ultimo avviene appunto perché dato l’i-esimo bit uguale a zero allora non necessitiamo il calcolo della distanza perché si tratta di un centroide non considerato.@ Lo stato “CALC-AND-UPDATE” invece analizza la variabile di incremento i con la quale discriminiamo lo stato seguente; in particolare se:

1. i<”111” o i=”111” e la distanza calcolata in questo stato tra il centroide e il punto considerato è la distanza minima fin’ora registrata allora attraverso “STALLO-ASSIGN” passiamo in “ASSIGN-ONE” ponendo a 1 il bit corrispondente nella maschera d’uscita azzerando tutti gli altri precedenti ad esso; per ultimo se i<”111” passiamo a “INCREMENT” per poi iniziare una nuova fase delle 8 del ciclo, altrimenti arriviamo a “FINAL-PREP”;
2. i<”111” e la distanza calcolata è uguale o maggiore alla distanza minima allora attraverso “INCREMENT” ,con il quale incrementiamo il contatore i, saremmo catapultati ad inizio ciclo “REQ-TEMPX” se i ha un valore minore di “111” o in “FINAL-PREP” altrimenti;
3. i=”111” e la distanza calcolata uguale o maggiore alla distanza minima allora saremmo catapultati direttamente in “FINAL-PREP”

Da FINAL-PREP, finito il ciclo di 8 fasi scriviamo la maschera d’uscita(outmask) in “WRITE-MASK” e terminiamo il processo proiettandoci in END-PROCESS e aspettando il i\_rst=1 per riiniziare il ciclo della macchina.

2

Testing

Una volta testato il componente con il test bench fornito ed aver constatato il corretto funzionamento in Behavioral e Post-Synthesis Functional sono stati cercati alcuni casi di test che potessero stressare di più il componente tra i quali:

* - Maschera generica e centroidi generici;
* - Centroidi con coordinate tutte coincidenti tra loro => maschera output=maschera  input;
* - Tutti i centroidi equidistanti dal punto sentinella => maschera output=maschera input;
* - Centroidi con coordinate generiche e maschera con tutti zero => maschera  output=tutti zero;
* - Centroidi con coordinate generiche e maschera con tutti uno => maschera  output=uno nei centroidi effettivamente più vicini alla sentinella, zero altrimenti;
* - Tutti i centroidi coincidono tra loro e con il punto (distanza=0 per ogni centroide)=>  maschera output=maschera input;
* - Maschera con unico uno in corrispondenza del centroide più distante dal punto  sentinella => maschera output=maschera input;
* - Maschera con tutti uno e i tre centroidi più vicini posti ad ugual distanza dal punto  sentinella => maschera output= 3 “uno” in corrispondenza dei tre centroidi a uguale  distanza;
* - Maschera con tutti uno, centroidi disposti a due a due sui quattro angoli e punto  sentinella al centro => maschera output= tutti uno;
* - Modifica del clock period;
* - Inserimento segnali di reset.

Con i test effettuati si è cercato di coprire più casi limite possibili in modo da poter testare a fondo la struttura logica dell’algoritmo; si è inoltre tentato di abbassare il clock period per osservare come un clock eccessivamente basso non permetta alla macchina di portare a termine la parte di programma relativa agli stati più onerosi.

3

OTTIMIZZAZIONI ARCHITETTURALI

Tra le possibili ottimizzazioni è sicuramente possibile innanzitutto ridurre il numero di accessi in memoria: nell’algoritmo da noi implementato, dopo aver acquisito la maschera di attivazione, vengono lette dalla memoria le coordinate di tutti gli otto centroidi, tuttavia una vasta ed interessante casistica presenta delle maschere in cui diversi centroidi vengono messi a 0, essendo dunque inattivi non sarebbe necessario andar ad accedere alla memoria per leggere le loro coordinate; a tal fine sarebbe sufficiente inserire un controllo (un semplice “if”) sull’indice della maschera relativo al centroide di interesse e nel caso in cui questo sia 0 andare direttamente allo stato successivo.

4