Prova Finale
Progetto di Reti Logiche

William Fornaciari – Davide Zoni

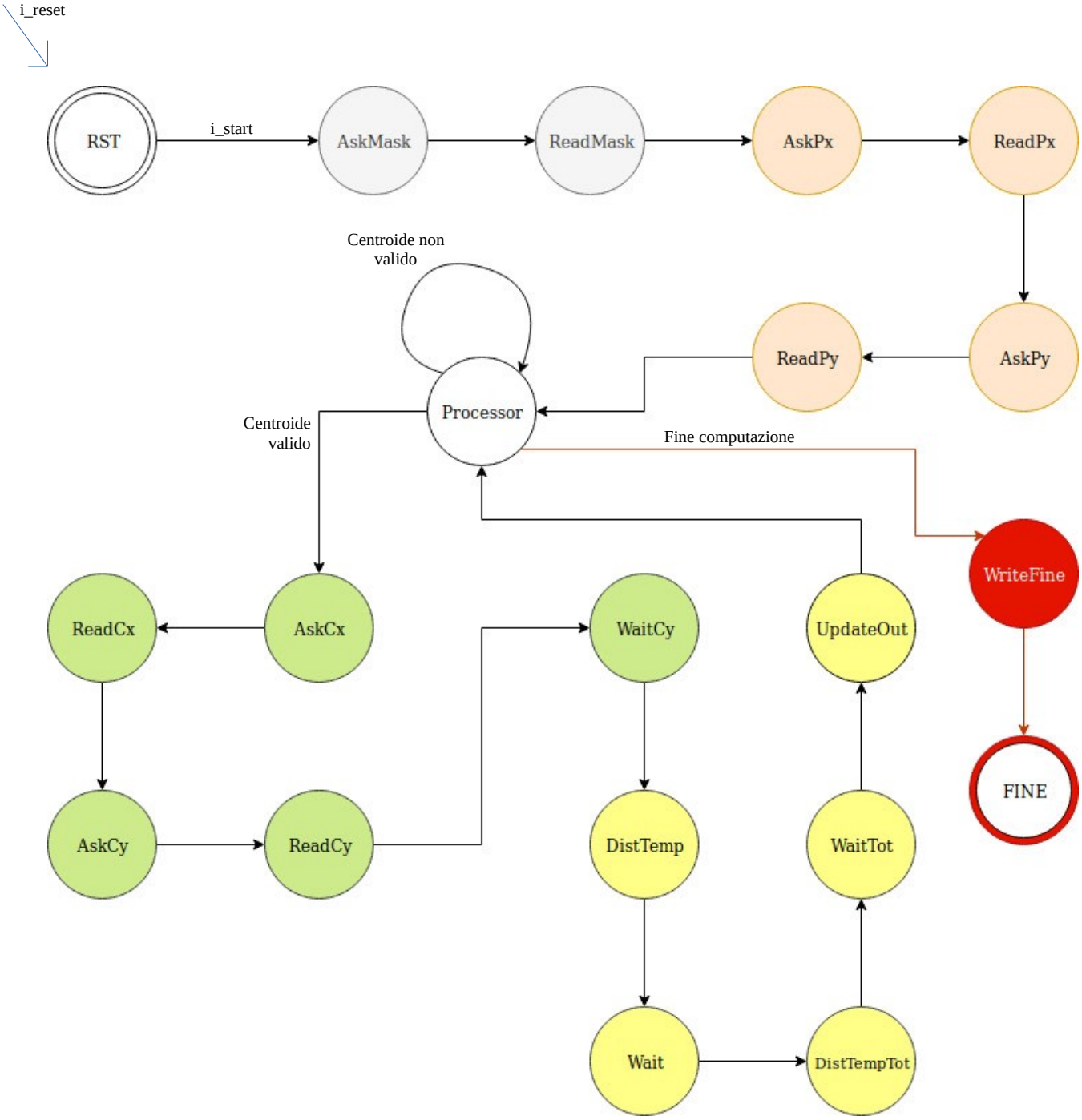
A cura di:

Andrea Alfieri
868085 – 10539026

Anno Accademico 2018 - 2019

Indice

Diagramma della macchina a stati finiti



Descrizione del componente

Il componente hardware da me progettato si occupa di trovare, tra tutti i centroidi “attivi”, quelli a distanza minima da un altro centroide detto pivot.
I centroidi e la maschera di quelli attivi si trovano in memoria e vengono acceduti ad indirizzi prestabiliti tramite i segnali di *enable*, *write-enable*, *address*, e *data.*
Una volta calcolati i centroidi a distanza minima, viene inviata alla memoria una maschera dove l’i-esimo bit è alto solo se l’i-esimo centroide si trova a distanza minima dal pivot.

L’algoritmo sfruttato è una semplice scansione lineare di tutti i centroidi attivi:
grazie all’utilizzo di due signal *minDistance* e *outputMask* è possibile infatti calcolare, tramite pochi confronti, la maschera d’uscita già durante la scansione stessa delle coordinate dei centroidi.

In particolare il funzionamento è basato sulla macchina a stati reppresentata alla pagina precedente:

- RST: è lo stato iniziale, nel quale tutti i segnali vengono inizializzati. Una volta che il segnale di i_start diviene alto, si passa allo stato successivo ed inizia effettivamente la computazione. Tutti gli stati hanno la capacità di tornare allo stato di reset se il segnale di i_rst è alto durante il fronte di salita del clock.

- AskMask e ReadMask: questi due stati si occupano di richiede la maschera dei centroidi attivi alla memoria. La lettura è però divisa in richiesta (Ask) e lettura effettiva (Read) per dare il tempo alla memoria di porre il dato sul segnale di i_data. Il tempo che passa tra uno stato e l’altro è di un ciclo di clock.

- AskPx, ReadPx, AskPy, ReadPy: questi quattro stati si occupano di leggere le coordinate del pivot dalla memoria, sfruttando la stessa logica degli stati precedenti.

- Processor: questo stato si occupa di valutare, tramite un contatore, se il nuovo centroide da considerare è attivo o meno. Nel caso in cui sia attivo, si passa ad AskCx, altrimenti si rimane sullo stato Processor aumentando il contatore. Una volta terminati tutti i controlli, si passa infine allo stato WriteFine.

- AskCx, ReadCx, AskCy, ReadCy, WaitCy: qui viene effettuata la richiesta in memoria delle coordinate del centroide considerato. Esiste inoltre lo stato WaitCy che permette al segnale corrispondente alla coordinata Y di stabilizzarsi, prima di essere letto allo stato successivo. Senza questo stato d’attesa non sarebbe infatti possibile accedere immediatamente a tale segnale.

- DistTemp, Wait, DistTempTot, WaitTot, UpdateOut: in questi stati vengono calcolate le due distanze in ciascuna direzione (DistTemp), sommate per ottenere la distanza effettiva (DistTempTot), e in base a quest’ultimo valore vengono aggiornati i segnali di *minDistance* e/o *outputMask.* Infine viene aumentato il contatore di 1 unità prima di passare nuovamente a Processor.
  Esistono anche qua alcuni stati d’attesa per permettere ai segnali di stabilizzarsi prima di essere letti nuovamente.

- WriteFine, Fine: alla fine della computazione vengono sfruttati questi due stati per scrivere il valore della maschera d’uscita in memoria e per alzare il segnale di o_done. Infine la macchina si prepara ad iniziare nuovamente la computazione.

Descrizione dei casi di test

Test riferiti alla maschera dei centroidi attivi

Per questa prima classe di test ho focalizzato la mia attenzione sulla maschera dei centroidi attivi. In particolare, mantenendo costante la lista dei centroidi e del pivot, ho testato che diverse configurazioni della maschera generassero tutte i risultati attesi.
Ho provato ad esempio ad impostarla a “00000000” e “11111111” ed altri valori significativi.
Poichè inizialmente la mia macchina a stati dava problemi quando l’ultimo bit della maschera era a 0, ho testato in maniera più approfondita le configurazioni di questo tipo (0xxxxxxx).
Infine, ho controllato che il signal del contatore si comportasse in maniera adeguata nei casi per lui più particolari, come “01010101” oppure “10101010”.

Test riferiti alla lista di centroidi attivi

Per questa fase sono state scelte alcune configurazioni particolari dei centroidi nel loro spazio di esistenza.
E’ stato testato che, con la maschera a “11111111”, 8 centroidi nella stessa posizione generassero i risultati attesi. La stessa maschera è stata utilizzata poi posizionando i centroidi in posizioni diverse, facendo però in modo che la maggior parte di essi fosse a distanza minima dal pivot, generando quindi configurazioni della maschera d’uscita con molti 1.

Test generati casualmente

Per questa classe di test ho scritto un programma in linguaggio C che generasse casualmente testbench in maniera automatizzata.
Sono quindi stati generati e testati 100 file di testbench in modo da verificare la robustezza del mio componente hardware.
Il codice sorgente del generatore può essere visionato nella cartella di consegna del progetto, chiamato *10539026.c* .

Note

La sintesi è stata effettuata ponendo come constrain:

create_clock -name clkconstrain -period 100 [get_ports i_clk]

Per tutti i test effettuati, è stato controllato che la simulazione funzionasse sia in Behavioral, che in Post-synthesis funtional e timing.