Giovanni Guidi
Codice persona 10523622
Maggio 2019

Progetto di Reti Logiche

Scaglione prof. Fabio Salice

Indice:

1. Introduzione

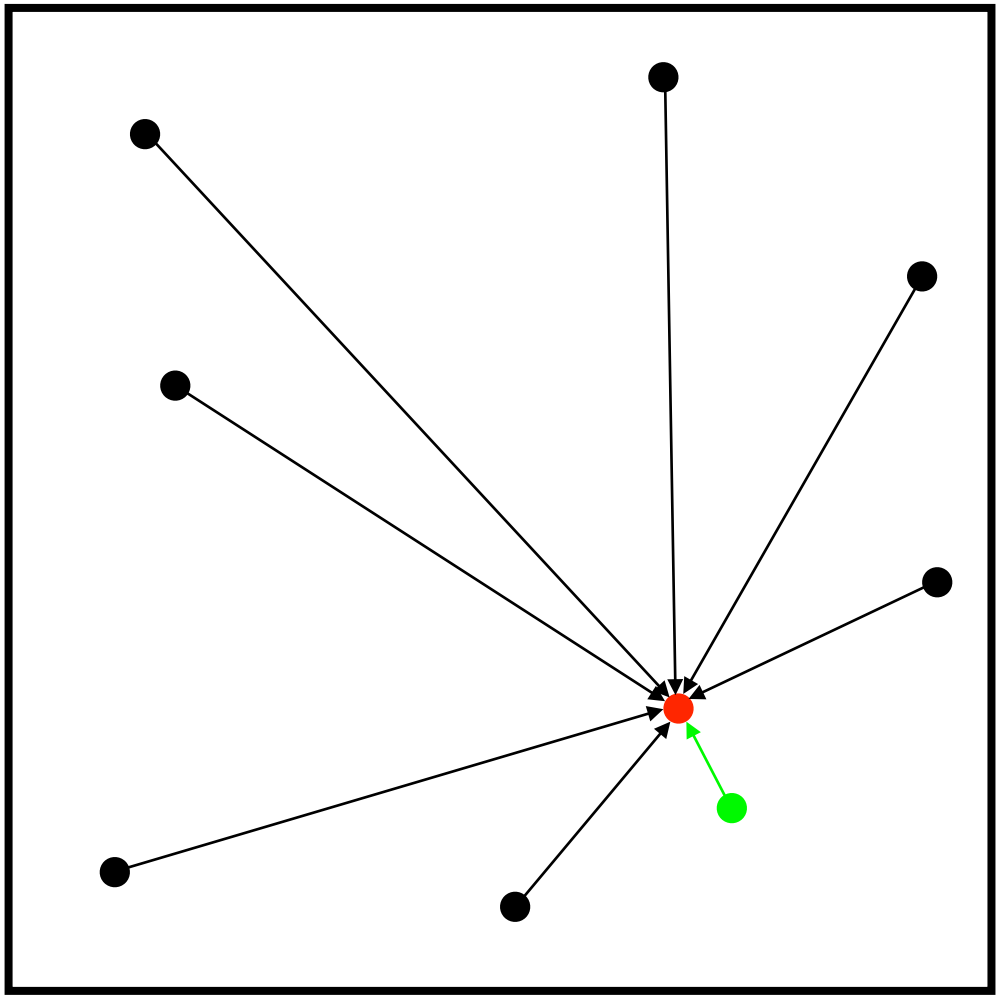
Lo scopo del progetto è quello di descrivere in VHDL un componente hardware che rispetti determinate specifiche.

In particolare si richiede l'implementazione di un componente che dati:

- uno spazio bidimensionale
- le posizioni di N punti detti centroidi
- le coordinate di un punto specifico

sia in grado di indicare quale (o quali) dei centroidi risulti più vicino a tale punto, utilizzando la distanza di Manhattan.

Si riporta un grafico che rappresenta un esempio di possibile scenario risolto: il **quadrato** è lo spazio da considerare, il punto rosso è il punto *speciale* per il calcolo delle distanze, il punto verde è tra gli N=8 centroidi quello a distanza minima.



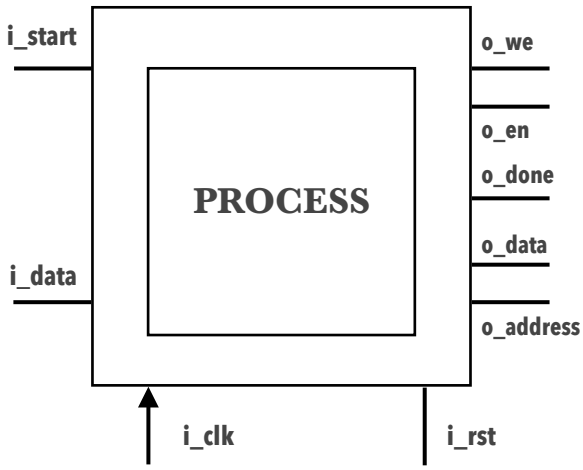
2. Architettura

L'interfaccia del componente è stata presentata nelle specifiche nel seguente modo:

```
entity project_reti_logiche is
port (
      i_clk         : in  std_logic;
      i_start       : in  std_logic;
      i_rst         : in  std_logic;
      i_data        : in  std_logic_vector(7 downto 0);
      o_address     : out std_logic_vector(15 downto 0) := "0000000000000000";
      o_done        : out std_logic := '0';
      o_en          : out std_logic := '0';
      o_we          : out std_logic := '0';
      o_data        : out std_logic_vector (7 downto 0) := "00000000"
      );
end project_reti_logiche;
```

Tale componente si interfaccia, inoltre, con un chip RAM in cui sono caricati tutti i dati necessari all'elaborazione.

La soluzione proposta prevede il seguente schema:



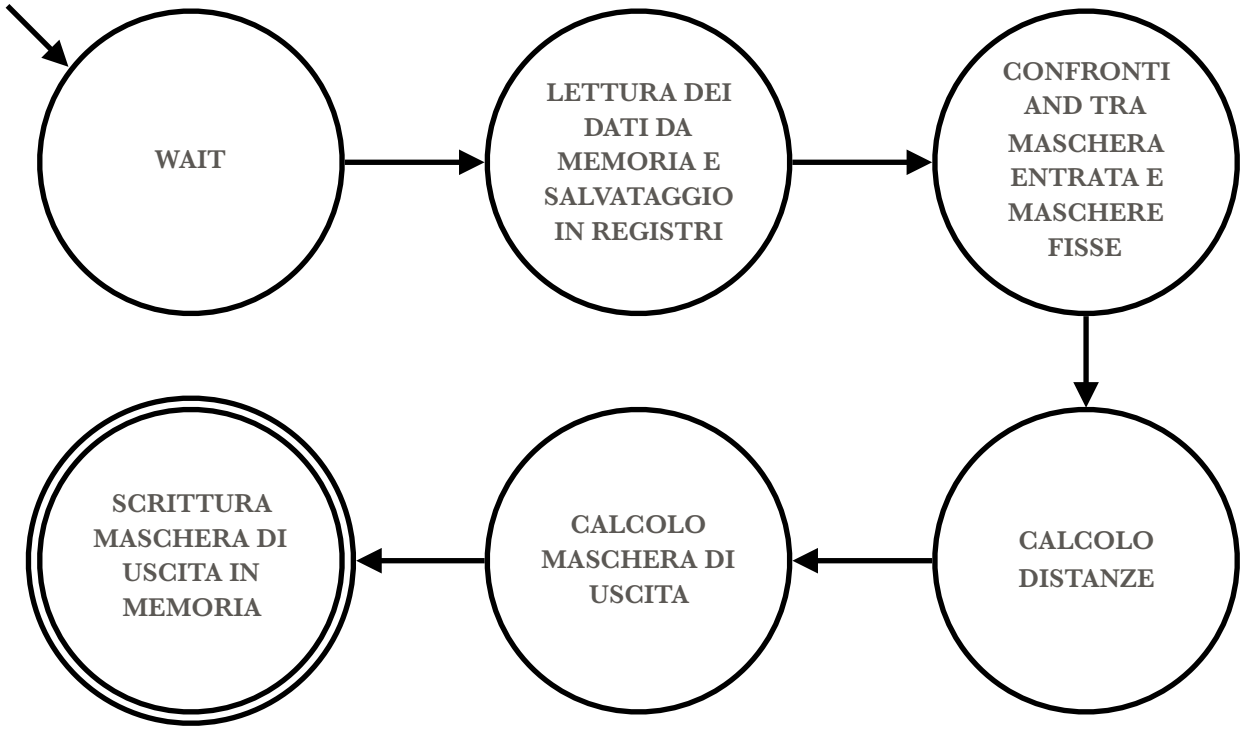
Per la realizzazione del componente HW, non essendo state fornite restrizioni per tempo di esecuzione e area utilizzata, si è decisa una soluzione semplice a livello di algoritmo, senza badare a particolari ottimizzazioni. A tal proposito il problema è stato risolto utilizzando un solo *process*, che implementa l'esecuzione di una macchina a stati finiti. Tale macchina prevede, a grandi linee, il seguente algoritmo di esecuzione:

1. Si accende la macchina e ci si pone in uno stato di attesa, fino a quando non si riceve un segnale di reset seguito da un segnale di start
2. Si legge la maschera di entrata e le coordinate di *tutti* i centroidi dalla memoria. Si salva ciascun dato in registri, in modo da poterli utilizzare in seguito (non badando

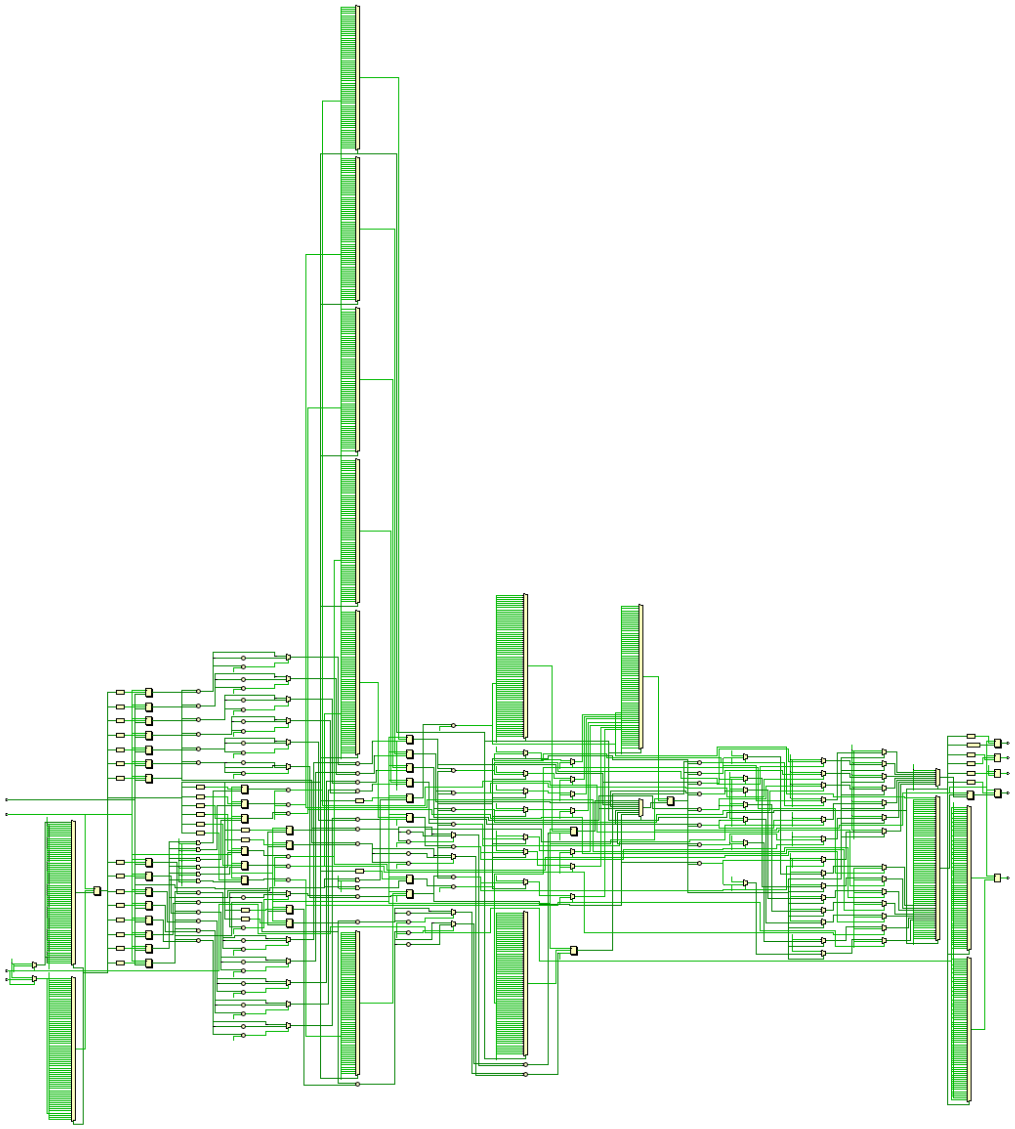
all'ottimizzazione si leggono tutti i dati, anche se poi alcuni potrebbero non venire utilizzati)

3. Si operano dei confronti *and* tra la maschera di entrata e delle maschere fisse, per individuare i centroidi validi che dovranno essere considerati per il calcolo delle distanze. Si salva il risultato di tali confronti in registri, per utilizzarli in seguito
4. Si calcolano le distanze dei centroidi validi dalle coordinate del punto fornito. Per capire quali siano i centroidi effettivamente da considerare, si prende come discriminante il fatto che se il confronto *and* del punto precedente ha avuto esito "positivo", il dato salvato nel registro dovrà avere almeno un bit posto a *1*
5. Si procede con il calcolo della maschera di uscita: per ogni centroide si verifica se questo si trova ad una distanza inferiore rispetto alla distanza minima salvata. La distanza minima viene memorizzata a partire dal primo centroide valido, ed eventualmente aggiornata se si scopre un centroide più vicino. È possibile che ci sia anche più di un centroide a distanza minima, in questo caso si pongono a *1* tutti i bit della maschera di uscita corrispondenti a tali centroidi
6. Si scrive in memoria la maschera di uscita calcolata nel punto precedente, che indica il centroide (o eventualmente i centroidi) a distanza minima

Si può riassumere il comportamento della macchina con i seguenti macro-stati:



Si riporta, inoltre, lo schema del design elaborato da Vivado:



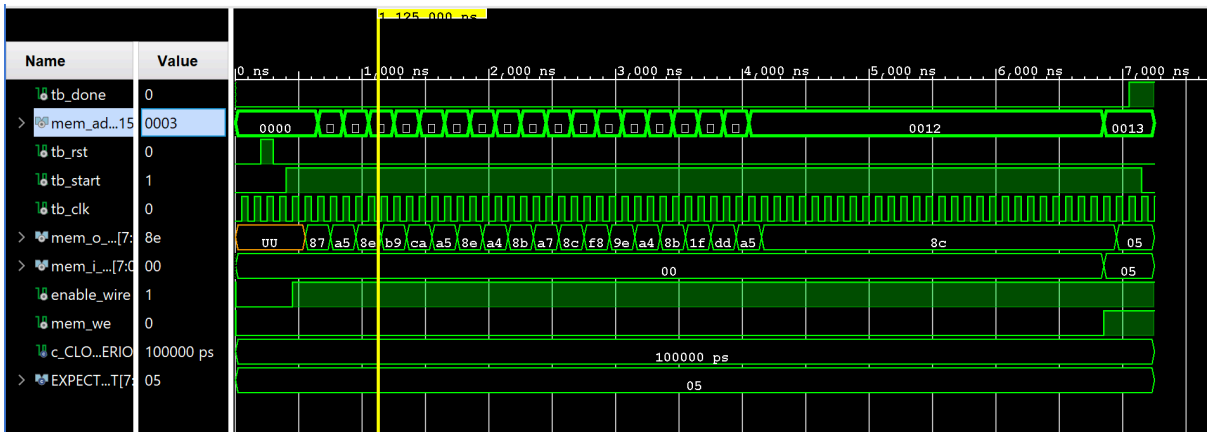
2. Risultati sperimentali

Oltre al *testbench* fornito e ad alcuni test in cui si sono generati casualmente i parametri dei dati memorizzati in memoria, si è ritenuto opportuno testare i casi limite più rilevanti, cioè quelli che con più probabilità avrebbero potuto mettere in errore la macchina sviluppata. Per questo motivo sono stati creati e verificati i seguenti casi di test:

A. Tutti i centroidi sono validi e si trovano a distanza massima dal punto
B. Tutti i centroidi sono validi e sono coincidenti
C. Tutti i centroidi sono validi e si trovano a distanza minima (coincidenti al punto)
D. Nessun centroide è valido (maschera di ingresso nulla)

Tutti questi casi di test hanno terminato la simulazione con esito positivo, dimostrando il corretto funzionamento del componente HW.

Infine si riporta la schematica dei segnali di un test di simulazione preso da *Vivado*:



4. Conclusioni

Il componente realizzato ha dimostrato di superare correttamente, con tutti i casi di test, la simulazione Behavioral, la simulazione Post-Syntesis Functional e quella Post-Syntesis Timing, in linea con quanti richiesto da specifiche.

Si ritiene quindi di aver sviluppato un componente hardware in grado di risolvere il problema richiesto.