Prova finale
Progetto di reti logiche
aa 2018/2019
Eugenio Ostrovan
Matricola : 866052
Codice persona : 10527025
Sezione del professore Fabio Salice

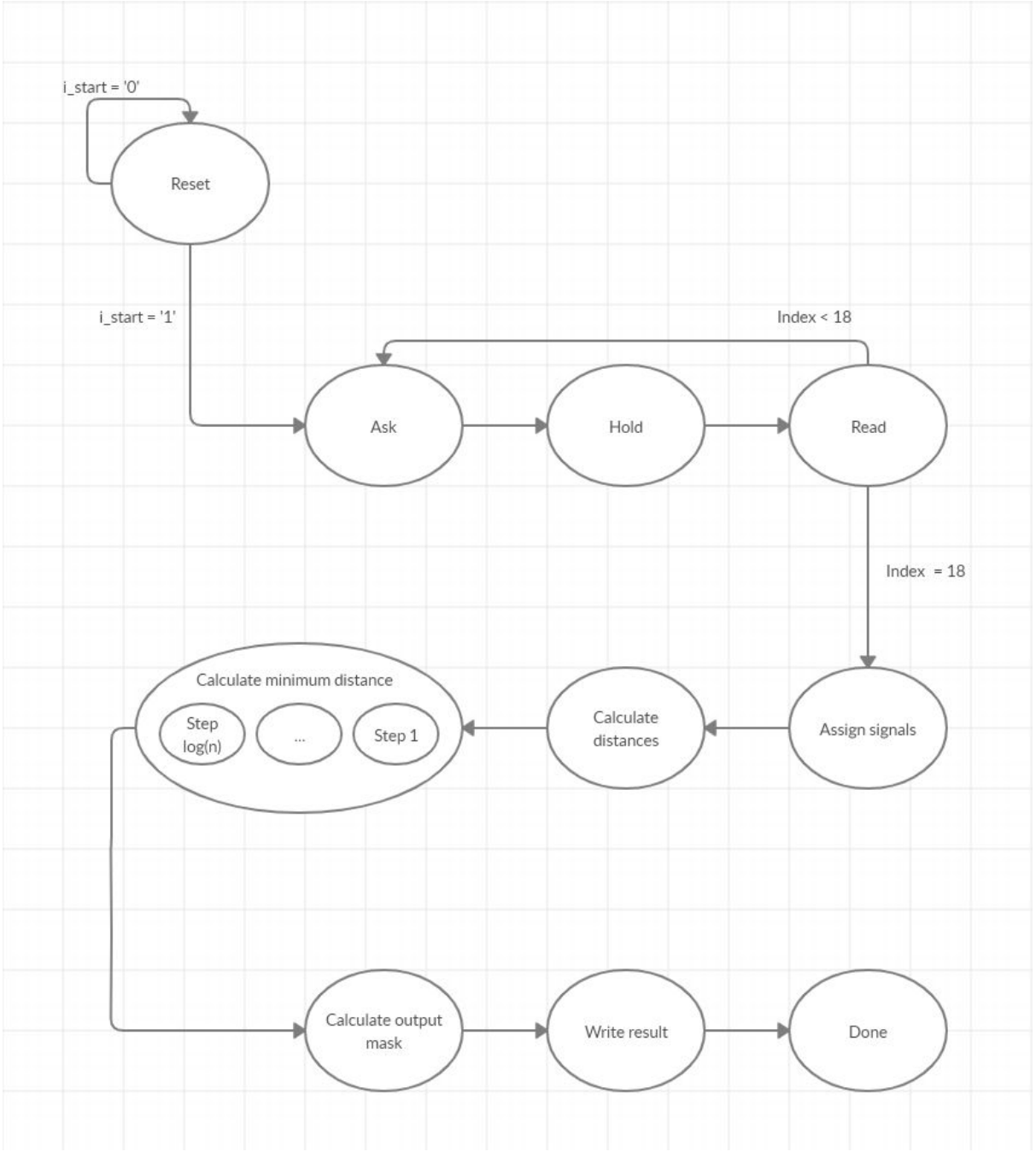
Introduzione.
L'obiettivo del progetto è stato l'implementazione corretta della specifica proposta. Non essendoci vincoli stringenti sulle risorse di area e tempo computazionale ho fatto le scelte progettuali con il criterio di realizzare un sistema concettualmente semplice. Come obiettivo personale mi sono posto quello di familiarizzarmi con un nuovo paradigma di elaborazione dei dati diverso dalla programmazione procedurale e sequenziale.

Descrizione della specifica.
Dati otto punti e un nono di riferimento in uno spazio bidimensionale (256x256) e una maschera a 8 bit che specifica i punti da valutare, produrre una maschera a 8 bit che indichi quali dei punti tra quelli da considerare si trova alla distanza minima dal punto di riferimento.

Descrizione della soluzione e delle scelte di progetto.
Per calcolare il risultato è necessario calcolare la distanza tra il punto di riferimento e, uno alla volta, tutti i punti da considerare; calcolare il minimo tra i valori ricavati; impostare a '1' i bit corrispondenti ai centroidi a distanza minima nella maschera di output.
Per calcolare la distanza minima è necessario avere a disposizione le informazioni su tutti i punti, tuttavia è possibile leggere dalla memoria un solo dato alla volta, qui si presentano due possibilità:

1. ricalcolare la maschera di output e la distanza minima ogni volta che viene letto un nuovo centroide
2. leggere le coordinate di tutti i punti e poi calcolare distanza minima e maschera di output

Ho scelto la seconda opzione, perché è quella in cui il sistema deve fare meno operazioni, è da notare tuttavia che questo non comporta una riduzione del tempo che il sistema impiega per calcolare la soluzione, perché nel caso della prima opzione è possibile eseguire in parallelo la lettura di un nuovo punto e il ricalcolo della maschera. La seconda soluzione comporta un maggiore utilizzo di memoria, perché deve conservare contemporaneamente le coordinate di tutti i punti.
Il comportamento è descritto dalla seguente macchina a stati:



La macchina parte nello stato di reset e c'è una transizione da tutti gli altri stati allo stato di reset non riportata nello schema che viene fatta quando i_rst = '1'. Segue un ciclo di tre stati in cui vengono letti dalla memoria tutti i dati necessari per calcolare il risultato, questi vengono memorizzati in un vector. Successivamente i dati vengono assegnati a segnali individuali, questo è un passaggio non necessario, ma semplifica la scrittura del codice e rende più trasparente il funzionamento. Nello stato 'calculate distances' vengono calcolate le distanze tra ogni punto e il punto di riferimento. Seguono tre stati che insieme servono a calcolare la distanza minima tra tutte quelle calcolate : a ogni passo le distanze vengono raggruppate due a due e tra queste si sceglie la minore fino ad ottenere un unico valore. Generalizzando, per calcolare la distanza minima servono un numero di passi pari a $log_2(numero\ di\ centroidi)$. Ogni passo, così come ogni altro stato della FSM richiede un ciclo di clock. Una volta calcolata la distanza minima, viene calcolata la maschera di output : il bit

corrispondente a un centroide viene impostato a ‘1’ se questo è da considerare e se la sua distanza dal punto di riferimento è pari alla distanza minima, altrimenti viene impostato a ‘0’. Il risultato viene scritto in memoria e la macchina si ferma nello stato ‘done’ in attesa di reset.

Test e verifica della correttezza.
Per verificare che il sistema rispetta la specifica sono state fatte simulazioni con il test bench di esempio. Questo è stato poi utilizzato come fondamento per creare altri casi di test usando dati diversi : diverse disposizioni dei punti e varie configurazioni della maschera di input i.e.

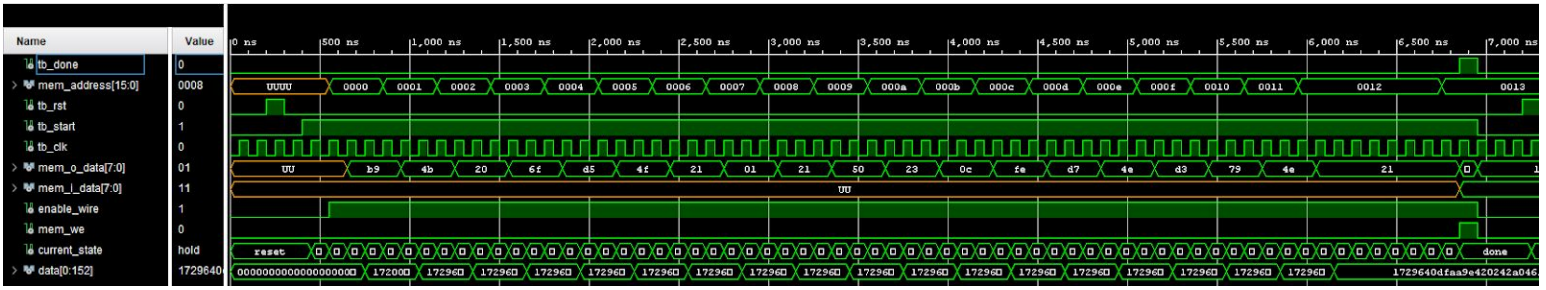
- centroidi molto vicini
- centroidi molto lontani
- centroidi coincidenti tra loro o con il punto di riferimento
- maschere di input con nessuno o tutti i bit attivi

cercando di rilevare comportamenti anomali. Questa fase è stata utile per scoprire errori importanti nelle precedenti versioni della soluzione.

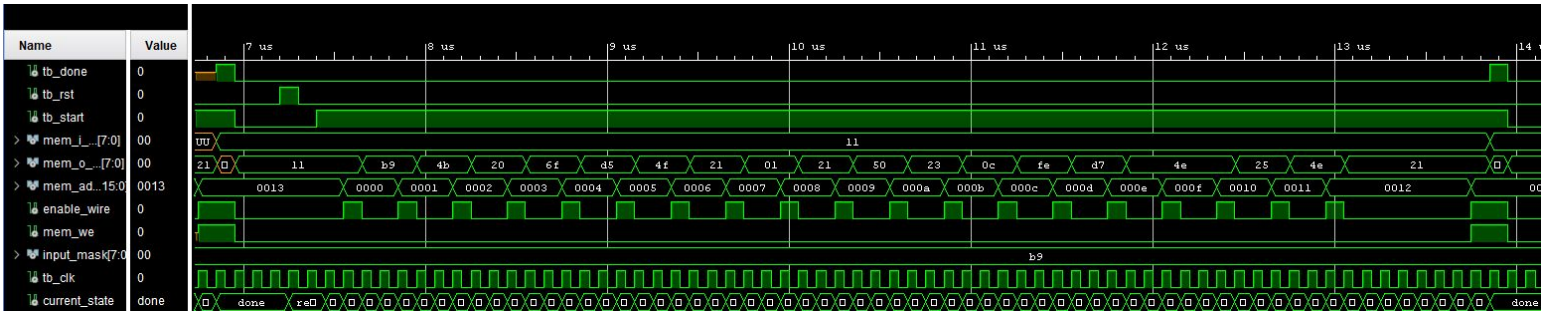
Per verificare ulteriormente il design ho scritto un programma C per la generazione casuale di dati di test che ho inserito manualmente nel simulation source, creando un sistema di verifica semiautomatico.
Esempio output del generatore:

```
eugenio@DESKTOP-EDRI7KL:/mnt/c/Users/Eugenio/Desktop$ ./generatore
x
200 141 175 65 175 242 72 174 21
y
239 168 168 43 25 102 182 253 96
signal RAM: ram_type := (0 => std_logic_vector(to_unsigned( 255 , 8)),
1 => std_logic_vector(to_unsigned( 200 , 8)),
2 => std_logic_vector(to_unsigned( 239 , 8)),
3 => std_logic_vector(to_unsigned( 141 , 8)),
4 => std_logic_vector(to_unsigned( 168 , 8)),
5 => std_logic_vector(to_unsigned( 175 , 8)),
6 => std_logic_vector(to_unsigned( 168 , 8)),
7 => std_logic_vector(to_unsigned( 65 , 8)),
8 => std_logic_vector(to_unsigned( 43 , 8)),
9 => std_logic_vector(to_unsigned( 175 , 8)),
10 => std_logic_vector(to_unsigned( 25 , 8)),
11 => std_logic_vector(to_unsigned( 242 , 8)),
12 => std_logic_vector(to_unsigned( 102 , 8)),
13 => std_logic_vector(to_unsigned( 72 , 8)),
14 => std_logic_vector(to_unsigned( 182 , 8)),
15 => std_logic_vector(to_unsigned( 174 , 8)),
16 => std_logic_vector(to_unsigned( 253 , 8)),
17 => std_logic_vector(to_unsigned( 21 , 8)),
18 => std_logic_vector(to_unsigned( 96 , 8)),
 Maschera (numero decimale) : 8
eugenio@DESKTOP-EDRI7KL:/mnt/c/Users/Eugenio/Desktop$
```

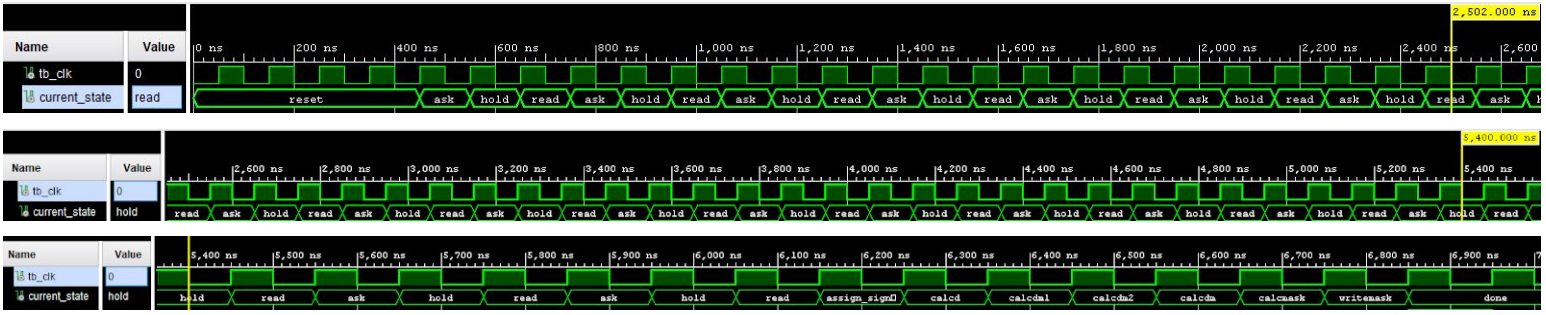
Simulazione pre sintesi con test bench di esempio:



Esecuzione di un altro test alla conclusione del primo:



Dettaglio segnale current_state:



Sintesi.

Il design viene sintetizzato e implementato correttamente dagli strumenti automatici di vivado. Sono stati risolti tutti gli errori e molti dei warning segnalati. Il design simulato post sintesi presenta un comportamento non corretto.

Dal report di sintesi:

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	619	0	134600	0.46
LUT as Logic	619	0	134600	0.46
LUT as Memory	0	0	46200	0.00
Slice Registers	509	0	269200	0.19
Register as Flip Flop	213	0	269200	0.08
Register as Latch	296	0	269200	0.11
F7 Muxes	0	0	67300	0.00
F8 Muxes	0	0	33650	0.00

Dai report di progetto:

Utilization Post-Synthesis | **Post-Implementation**

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	619	133800	0.46
FF	213	267600	0.08
IO	37	285	12.98
BUFG	2	32	6.25

Conclusione.
Il design creato opera correttamente in fase pre sintesi.