Dipartimento di Elettronica Informazione e Bioingegneria

Corso di Laurea in Ingegneria Informatica

Politecnico di Milano

Prova Finale

(Progetto di Reti Logiche)

Edoardo Carrer
Codice Persona 10561353
Matricola 870718

Amedeo Cavallo
Codice Persona 10562259
Matricola 868665

Professore di riferimento
William Fornaciari

Tutor di riferimento
Davide Zoni

Politecnico di Milano 2019

Contenuti

Introduzione

Lo scopo del progetto è la realizzazione di un componente hardware in VHDL. Esso riceve in ingresso le coordinate di otto centroidi e di un punto da valutare. Dopo aver calcolato la "Manhattan Distance" di ogni centroide da tale punto è in grado di individuare quale/i tra questi ha distanza minore.

Il componente si interfaccia con un chip RAM nel quale sono stati precaricati i dati da analizzare:

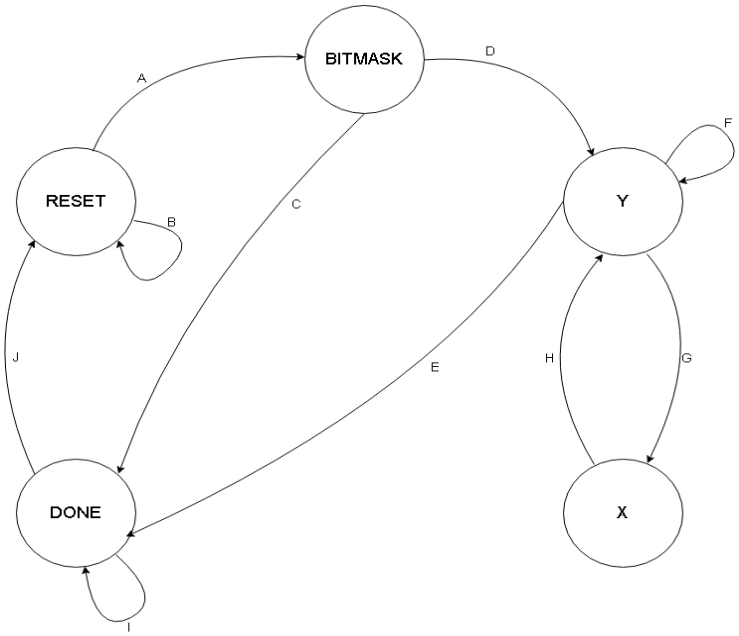
- All'indirizzo 0b0 (0) della RAM è memorizzata una bitmask di 8 bit che indica i centroidi attivi e dunque da valutare.
- Negli indirizzi di memoria successivi sono presenti alternativamente le coordinate X e Y dei centroidi.
- All'indirizzo 0b10001 (17) e 0b10010 (18) della RAM sono memorizzati rispettivamente la coordinata X e la coordinata Y del punto da valutare.
- Secondo le specifiche, il testbench richiede che il risultato della computazione sia salvato nella memoria RAM all'indirizzo 0b10011 (19).

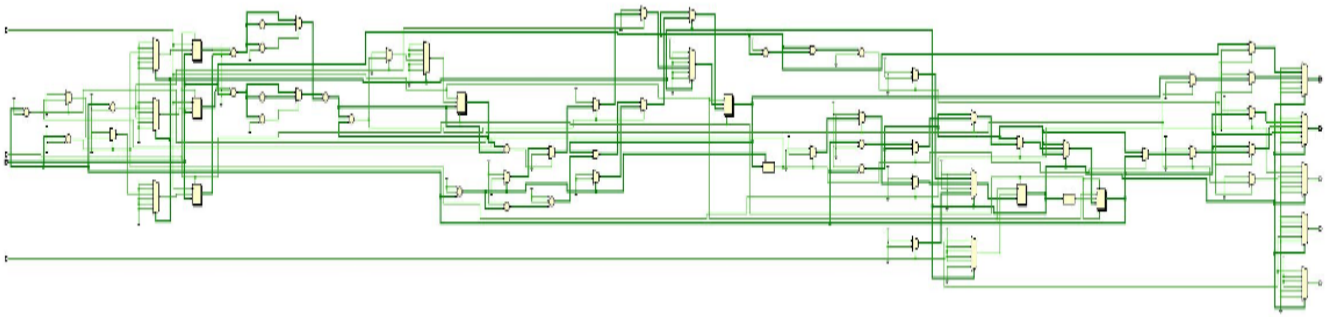
L'interfaccia del componente, così come presentata nelle specifiche, è la seguente:

```
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);
    o_done : out std_logic;
    o_en : out std_logic;
    o_we : out std_logic;
    o_data : out std_logic_vector (7 downto 0)
  );
end project_reti_logiche;
```

Scelte progettuali

Data la presenza di un segnale di start e di uno di reset, è ragionevole pensare che sia necessario un automa a stati finiti (FSM). In particolare, per modellare il design, abbiamo utilizzato una macchina di Mealy a 5 stati. Di seguito una breve descrizione degli stati con le relative transizioni.





1.1 Stato RESET

Lo stato iniziale è RESET, nel quale la macchina attende il segnale di i_start proveniente dal testbench. Ogni altro stato confluisce ovviamente in RESET ogni qualvolta i_rst è posto alto.

In questo stato tutti i segnali vengono portati al valore di default.

1.2 Stato BITMASK

In questo stato viene letta dalla RAM la bitmask dei centroidi attivi da analizzare.

Nel caso in cui non ci siano punti da analizzare o ci sia un solo punto attivo la bitmask letta viene propagata direttamente in uscita per essere scritta in memoria in quanto nessuna modifica è necessaria.

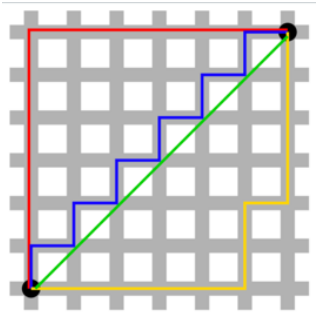
1.3 Stato Y

Stato in cui si legge la coordinata Y del punto da valutare e dei centroidi e a computazione terminata viene scritto il risultato finale nella RAM.

L'indirizzo per la lettura nella RAM viene via via decrementato ad ogni ciclo di clock, poiché è possibile accedere alla memoria RAM sequenzialmente. Nel caso in cui il centroide non sia attivo l'indirizzo viene decrementato due volte in modo da non effettuare letture superflue.

1.4 Stato X

Stato in cui si legge la coordinata X del punto e dei centroidi e nel secondo caso in cui si calcola la distanza tra il punto da valutare e il centroide in ingresso.



Nel caso in cui la distanza appena calcolata sia minore della distanza minima quest'ultima viene aggiornata.

Manhattan distance (la distanza tra due punti è la somma del valore assoluto delle differenze delle loro coordinate):

$$L_1(P_1, P_2) = |x_1 - x_2| + |y_1 - y_2|.$$

1.5 Stato DONE

Durante l'ultimo ciclo di clock la computazione è ormai terminata e viene posto alto il segnale in uscita o_done, la memoria RAM viene posta in IDLE abbassando il segnale o_en. Lo stato successivo è RESET che predispone il componente ad una nuova elaborazione.

Testing

Il componente supera correttamente la simulazione Behavioral, la simulazione Post-Syntesis Functional e la simulazione Post-Syntesis Timing. Ulteriori test sono stati effettuati nel corso della progettazione del componente anche in Post-Implementation

Abbiamo sviluppato un programma in linguaggio Python in grado di effettuare in Behavioral, Post-Syntesis Functional e Post-Syntesis Timing simulazioni con valori casuali. Nel caso di errori durante la sintesi vengono mostrati i log e a fine computazione viene analizzata la percentuale di test passati rispetto al numero test richiesto. Questo ci ha permesso di fare un elevato numero di test e di verificare il corretto funzionamento del componente.

Oltre ai test generati in modo randomico abbiamo individuato altri casi di test critici per verificare l'affidabilità del nostro componente:

- Test con tutti i centoidi attivi e, a partire dal primo, i centroidi hanno distanza decrescente, in modo che il componente venga stressato al massimo sovrascrivendo ogni volta la distanza minima trovata.
- Test con bitmask tutta a 0.
- Test con un solo 1.
- Test con due 1.

Risultati della sintesi

Vivado Post-Syntesis utilization report:

1. Slice Logic

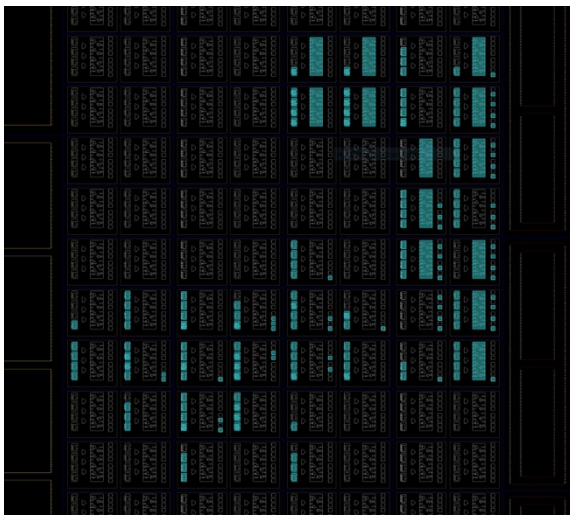
Site Type	Used	Fixed	Available	Util%
Slice LUTs*	112	0	134600	0.08
LUT as Logic	112	0	134600	0.08
LUT as Memory	0	0	46200	0.00
Slice Registers	49	0	269200	0.02
Register as Flip Flop	49	0	269200	0.02
Register as Latch	0	0	269200	0.00
F7 Muxes	0	0	67300	0.00
F8 Muxes	0	0	33650	0.00

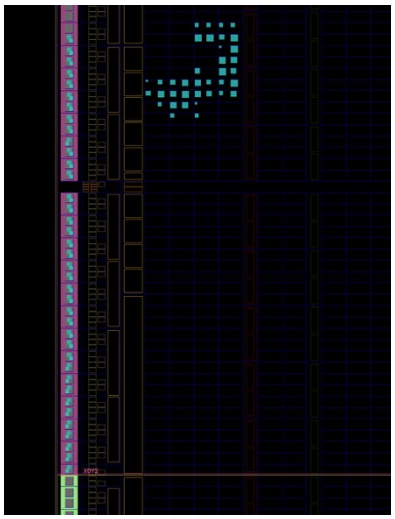
4. IO and GT Specific

Site Type	Used	Fixed	Available	Util%
Bonded IOB	38	0	285	13.33
Bonded IPADs	0	0	14	0.00
Bonded OPADs	0	0	8	0.00
PHY_CONTROL	0	0	10	0.00
PHASER_REF	0	0	10	0.00
OUT_FIFO	0	0	40	0.00
IN_FIFO	0	0	40	0.00
IDELAYCTRL	0	0	10	0.00
IBUFDS	0	0	274	0.00
GTPE2_CHANNEL	0	0	4	0.00
PHASER_OUT/PHASER_OUT_PHY	0	0	40	0.00
PHASER_IN/PHASER_IN_PHY	0	0	40	0.00
IDELAYE2/IDELAYE2_FINEDELAY	0	0	500	0.00
IBUFDS_GTE2	0	0	2	0.00
ILOGIC	0	0	285	0.00
OLOGIC	0	0	285	0.00

5. Clocking

Site Type	Used	Fixed	Available	Util%
BUFGCTRL	1	0	32	3.13
BUFIO	0	0	40	0.00
MMCME2_ADV	0	0	10	0.00
PLLE2_ADV	0	0	10	0.00
BUFMRCE	0	0	20	0.00
BUFHCE	0	0	120	0.00
BUFR	0	0	40	0.00





Come previsto vengono utilizzati 49 registri come Flip Flop e 38 pin di input e output. Per quanto riguarda la logica combinatoria vengono utilizzati 112 LUT su 134600 (0.08 %).

In figura vengono mostrati gli slices e i pin di input e output effettivamente utilizzati.

Ottimizzazioni

Di seguito una breve descrizione delle ottimizzazioni sviluppate per il progetto.

La prima ottimizzazione introdotta gestisce la bitmask nel caso in cui l'ingresso è 0 oppure un multiplo di 2, quindi nella matrice esiste al massimo un unico centroide attivo, di conseguenza si passa direttamente dallo stato BITMASK a quello di DONE scrivendo in memoria direttamente la bitmask appena letta senza dover verificare se gli altri centroidi siano attivi o meno, in quanto o non ci sono centroidi attivi o l'unico centroide attivo sarà sicuramente anche il più vicino.

Un ulteriore ottimizzazione è l'utilizzo della coordinata X dei centroidi utilizzata per il calcolo della distanza senza assegnarla ad un segnale, in modo da poter utilizzarla appena viene resa disponibile dalla RAM.

Infine, quando si è nello stato Y, se un centroide non è attivo il contatore viene decrementato di 2 in modo tale da evitare la transizione allo stato successivo e quindi la lettura della corrispondente X.