Politecnico di Milano

Relazione prova finale di reti logiche

A.A. 2018/19

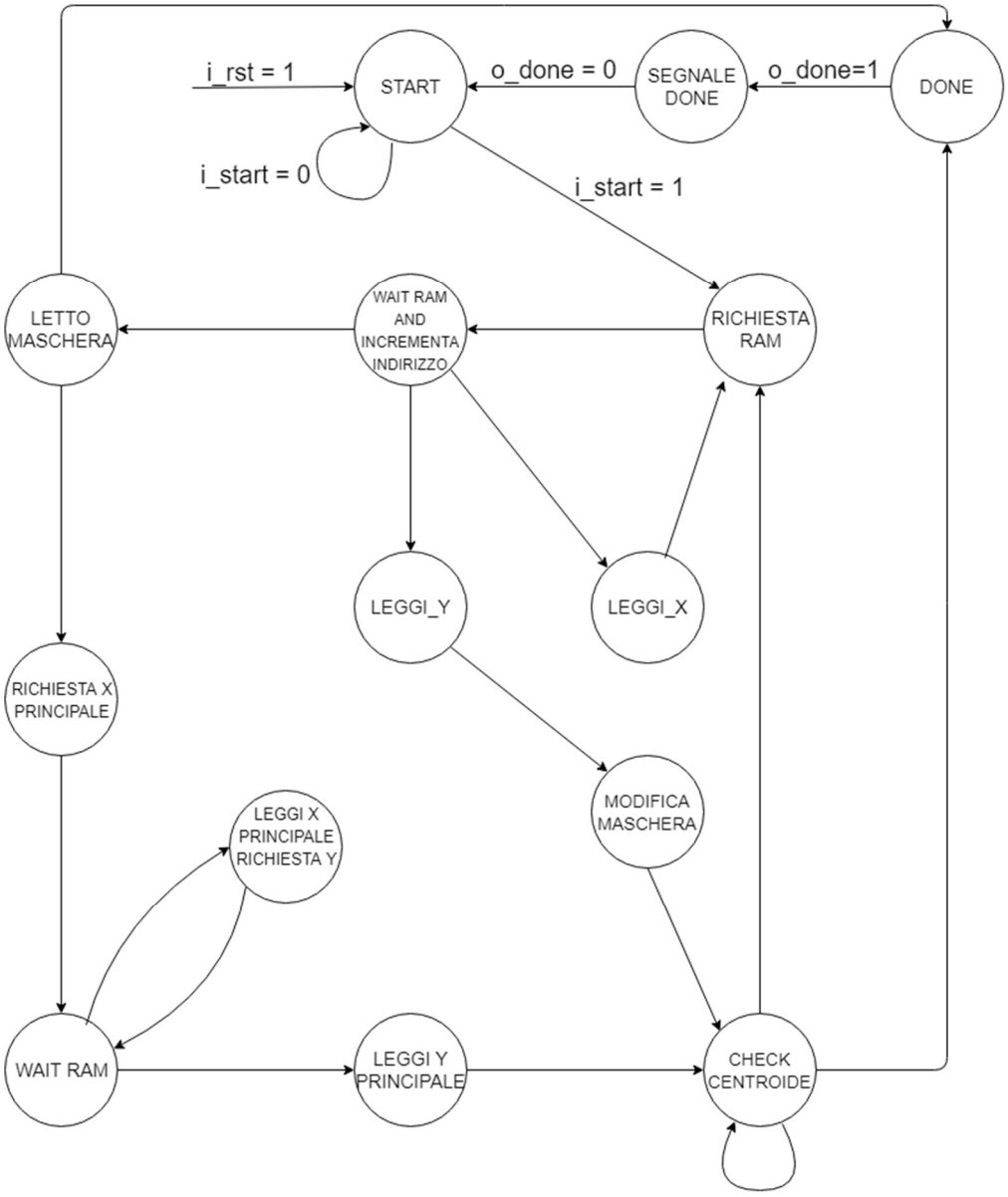
POLITECNICO
MILANO 1863

Marco Pianta
Davide Parretta

INDICE:

La prova finale di reti logiche e la relativa documentazione sono stati elaborati da:
Marco Pianta e da Davide Parretta

MACCHINA A STATI



Per poter risolvere il problema presentato nella specifica si è deciso di progettare una macchina a stati che rendesse possibile la descrizione in linguaggio VHDL usando il pattern behavioural.
Per descrivere la macchina a stati si è deciso di specificare un tipo di dato che contenesse una lista di tutti gli stati della macchina, questo tipo è stato usato poi per gestire le transizioni di stato.

DESCRIZIONE STATI

- START: Inizializzazione a valori di default per variabili e segnali. La transizione allo stato successivo avviene, come da specifica, alla ricezione del segnale i_start = 1
- DONE: Assegnamento del valore della maschera ottenuta a o_address e o_done posto a 1
- SEGNALE DONE: o_done posto a 0 nell'attesa di una nuova computazione
- RICHIESTA RAM / RICHIESTA X PRINCIPALE: Richiesta alla RAM dei dati contenuti all'indirizzo o_address

- WAIT RAM AND INCREMENTA INDIRIZZO / WAIT RAM: Stato di attesa per la ricezione del valore richiesto alla RAM, nel primo stato si incrementa anche l'indirizzo della prossima richiesta
- LEGGI_X / LEGGI_Y / LETTO MASCHERA / LEGGI Y PRINCIPALE: Leggono e salvano i rispettivi valori
- LEGGI X RICHIESTA Y PRINCIPALE: Lettura della coordinata x del punto da valutare e richiesta alla RAM della coordinata y
- MODIFICA MASCHERA: Modifica il valore della maschera da assegnare a o_address in base al calcolo della distanza effettuato durante gli stati precedenti
- CHECK CENTROIDE: Verifica tramite la maschera in ingresso se il centroide va considerato nella computazione

POSSIBILI OTTIMIZZAZIONI:

Come si può notare dalla descrizione precedente degli stati, si sarebbe potuto ottimizzare il numero degli stati della macchina collassando lo stato WAIT RAM nello stato WAIT RAM AND INCREMENTA INDIRIZZO, lo stato RICHIESTA X PRINCIPALE nello stato RICHIESTA RAM, lo stato LEGGI X PRINCIPALE RICHIESTA Y negli stati RICHIESTA RAM e LEGGI_X infine LEGGI Y PRINCIPALE in LEGGI_Y.
Queste ottimizzazioni avremmo potuto effettuarle inserendo nel codice degli stati di destinazione dei costrutti if-else, che avrebbero consentito di differenziare l'evoluzione degli stati nel caso in cui si sarebbero dovute considerare le coordinate dei centroidi, oppure nel caso in cui si sarebbero dovute considerare quelle del punto "principale".

Non avendo notato grosse differenze nelle prestazioni, abbiamo optato per non applicare queste ottimizzazioni a favore di una maggiore chiarezza e leggibilità nel codice, evitando di progettare stati troppo complessi.

SCELTE PROGETTUALI

Si è scelto di descrivere in VHDL un componente in logica sequenziale. Il componente è descritto da un solo processo che viene attivato dalla variazione del clock o da una eventuale variazione del segnale di reset.
Il controllo del segnale di reset risulta asincrono in quanto la sua variazione viene valutata indipendentemente dalla variazione del segnale di clock; l'evoluzione della macchina invece è sincronizzata con il fronte di salita del clock.

La lettura delle coordinate dalla RAM è stata pensata iniziando a leggere e salvare in segnali la maschera di ingresso e le coordinate del punto da valutare, per poi rendere possibile la creazione di un ciclo tra gli stati per la lettura e la computazione dei vari centroidi.

Negli stati in cui si deve attendere la ricezione dei dati dalla RAM si è deciso, per ottimizzare la macchina e risparmiare cicli di clock (rendendo la macchina più efficiente), di effettuare operazioni quali l'incremento dell'indirizzo da richiedere alla memoria.

Per ottimizzare ulteriormente la macchina si è deciso di calcolare la distanza corrente da un centroide negli stati in cui la coordinata viene ricevuta invece di salvare le varie coordinate per poi calcolare la distanza in un ulteriore stato.

Le variabili che contengono la distanza dei punti dai centroidi sono dimensionate a nove bit, in quanto la distanza massima possibile risulta essere 510. Per i valori numerici si è deciso di utilizzare come tipo unsigned, per evitare problemi legati al primo bit che avrebbe indicato altrimenti il segno.

TEST

TEST GENERICI

Per eseguire le simulazioni abbiamo scritto un generatore casuale di testbench in python, di seguito è riportata la parte più rilevante dello script ai fini del calcolo dei valori da inserire nel testbench

```
dist = [] #lista che contiene le distanze dai centroidi calcolate
x = [None]*8  #lista che contiene le coordinate x dei centroidi
y = [None]*8  #lista che contiene le coordinate y dei centroidi
out = ''
maschera_in = SystemRandom().choice(range(0,256)) #estrazione casuale maschera ingresso
x[0] = SystemRandom().choice(range(0,256))        #estrazione casuale coordinata centroide
                                    .
                                    . #estrazione delle coordinate dei punti rimanenti
                                    .
y_principale = SystemRandom().choice(range(0,256))
a = 7
for i in range(0,8):        #ciclo che calcola la distanza dei soli centroidi da valutare
    if("{0:08b}".format(maschera_in)[a-i] == '1'):   #maschera_in trasformata in binario
        dist.append(abs(x_principale-x[i]) + abs(y_principale-y[i]))
    else:       #se il centroide non va valutato viene inserita una distanza di default
        dist.append(600)        #600 supera la possibile distanza massima
minimo = min(dist)      #calcolo della minima distanza
a = 7
for i in range(0,8):    #ciclo che valuta quali centroidi sono a distanza minima
    if("{0:08b}".format(maschera_in)[a-i] == '1'):
        if(dist[i] == minimo):   #creazione della maschera
            out = '1' + out
        else:
            out = '0' + out
    else:
        out = '0' + out
```

Una volta terminata la fase di calcolo dei valori, questi vengono inseriti in un file di testbench creato seguendo l'esempio di quello fornito.
Usando questo generatore abbiamo prodotto circa 50 file di testbench, simulati poi in pre e post sintesi tramite un apposito script eseguito dalla TCL console di Vivado.

TEST SPECIFICI

Dopo aver superato i test generici si è deciso di generare dei testbench di casi particolari, in cui, dove non diversamente specificato, la maschera di ingresso è formata da bit posti tutti a 1. Questa scelta è dettata dal fatto che, avendo effettuato test casuali, si è appurato il corretto funzionamento della parte di codice riguardante la computazione dei centroidi a seconda della maschera d'ingresso.

- Test con centroidi coincidenti:
  In questo test i centroidi risultano essere tutti nella stessa posizione, differente da quella del punto principale.
- Test con centroidi aventi distanza uguale ma posizione differenti:
  In questo test i centroidi risultano essere tutti equidistanti dal punto principale, ma posizionati in punti differenti.

- Test con centroidi coincidenti con il punto principale
  In questo test alcuni centroidi hanno coordinate coincidenti con quelle del punto principale, i centroidi e il punto risultano avere distanza uguale a zero.
- Test con centroidi coincidenti a distanza massima dal punto principale
  In questo test tutti i centroidi hanno coordinate x = 0 e y = 0, mentre il punto principale ha coordinate x = 255 e y = 255.
- Test con machera di ingresso con tutti i bit posti a 0
  In questo test il valore delle coordinate dei centroidi è prodotto dal generatore casuale sopradescritto, la maschera d'ingresso ha tutti i bit posti a 0, di conseguenza la maschera d'uscita ha tutti i bit uguali a 0.

Abbiamo deciso di effettuare questi test poiché in questi casi il calcolo della distanza dei centroidi dal punto principale, e della relativa maschera d'uscita, può risultare critico.

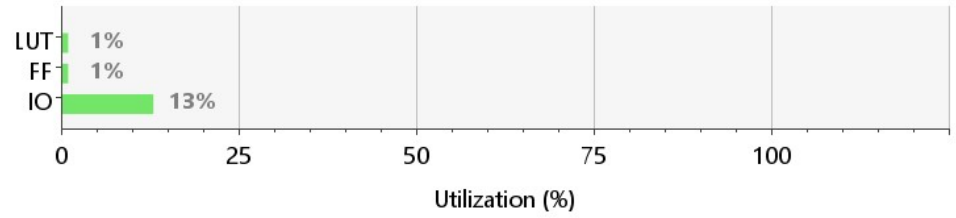
Oltre a effettuare test modificando le posizioni dei centroidi, del punto principale e il valore della maschera, abbiamo effettuato dei test relativi ai segnali della sensitivity list del processo del componente

- Test con reset casuale
  In questo test, in vari momenti della computazione, abbiamo alzato e successivamente abbassato il segnale di reset per testare il funzionamento del componente al variare del segnale di reset, non solo all'inizio della computazione.
- Test con clock minore di 100 ns
  Dopo aver eseguito vari test, diminuendo via via la durata del periodo di clock, abbiamo concluso che la soglia minima per il funzionamento del componente in pre e post sintesi è di 6 ns, questo porta chiaramente ad una diminuzione del tempo necessario per concludere la computazione.

VIVADO REPORT UTILIZATION

Di seguito riportiamo il summary del report utilization generato da Vivado.

Resource	Utilization	Available	Utilization %
LUT	155	134600	0.12
FF	121	269200	0.04
IO	38	285	13.33



Dal report si può notare che la quantità di LUT e Flip Flop utilizzati è molto bassa per via della poca complessità della funzionalità del componente.