ANNO ACCADEMICO: 2018/2019

PROVA FINALE

PROGETTO DI RETI LOGICHE

PROFESSORE: SALICE FABIO

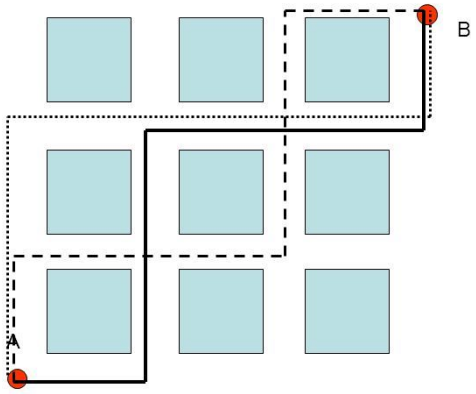
PIETRO LENTINI

POLITECNICO
MILANO 1863

INDICE

INTRODUZIONE

Il componente hardware progettato, dato uno spazio bidimensionale (256x256) ed 8 centroidi, è in grado di valutare se vi sono uno o più centroidi a distanza minima da un altro punto dato dello spazio (Manhattan distance).



I dati da utilizzare (ciascuno di 8 bit) sono letti da una memoria ad indirizzamento al byte:

- Indirizzo 0: maschera che stabilisce i centroidi da considerare per il risultato (i centroidi vengono identificati dal centroide 1 al bit meno significativo, al centroide 8 al bit più significativo; il bit è ad 1 se viene considerato, a 0 altrimenti);
- Indirizzi dall'1 al 16: Coordinate dei centroidi in ordine Coordinata X 1° centroide, Coordinata Y 1° centroide, Coordinata X 2° centroide, e così via fino all'ottavo centroide (valori interi senza segno)
- Indirizzi 17 e 18: Coordinate X e Y del punto da valutare (valori interi senza segno).

Il componente elabora quando un segnale in ingresso START viene portato ad 1. Al termine dell'elaborazione il componente porta un segnale in uscita DONE a 1, il quale verrà portato a 0 quando il segnale di START sarà portato a 0. Il componente riceve anche un segnale in ingresso RESET, il quale se portato ad 1 porta il componente allo stato iniziale e pronto ad elaborare i dati in memoria.

Il risultato verrà scritto nell'indirizzo 19 sotto forma di una maschera: i centroidi vengono identificati dal centroide 1 al bit meno significativo, al centroide 8 al bit più significativo (come per l'indirizzo 0) ed il bit è ad 1 se la sua distanza dal punto considerato è la minore tra le distanze dei punti da considerare, a 0 se la sua distanza dal punto non è la minore o non è da considerare (è possibile che ci siano più punti con la stessa distanza minima).

INTERFACCIA DEL COMPONENTE

```
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;
```

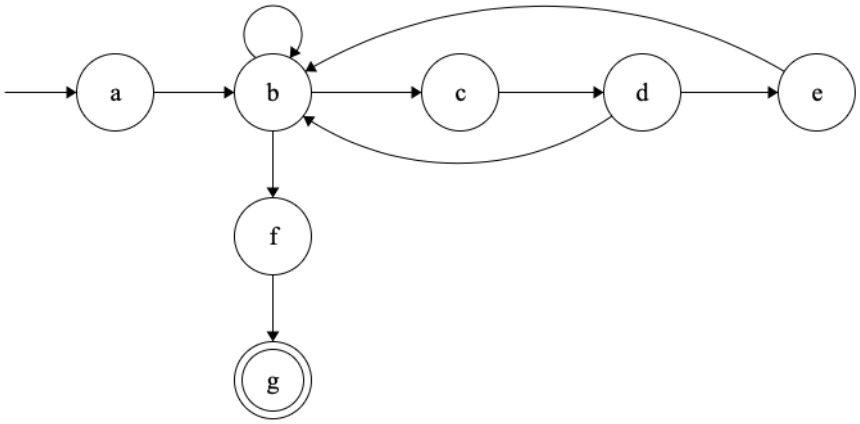
- `i_clk` : il segnale di clock generato dal test bench;
- `i_start` : il segnale di START generato dal test bench;
- `i_rst`: il segnale di RESET che inizializza la macchina pronta per ricevere il primo segnale;
- `i_data`: il segnale (vettore) che arriva dalla memoria in seguito ad una richiesta di lettura;
- `o_address`: il segnale (vettore) di uscita che manda l'indirizzo alla memoria;
- `o_done`: il segnale di uscita che comunica la fine dell'elaborazione e il dato di uscita scritto in memoria;
- `o_en`: il segnale di ENABLE da dover mandare alla memoria per poter comunicare (sia in lettura che in scrittura);
- `o_we`: il segnale di WRITE ENABLE da dover mandare alla memoria posto ad 1 per poter scriverci. Per poter leggere da memoria esso deve essere 0;
- `o_data`: il segnale (vettore) di uscita dal componente verso la memoria.

ARCHITETTURA

Per la descrizione della funzionalità del componente è stato scelto il livello di astrazione algoritmico (comportamentale - behavioral) mediante l'uso di un singolo process sensibile ai segnali di CLOCK e di RESET.

PROGETTAZIONE

L'algoritmo è stato progettato sulla base della seguente macchina a stati finiti:



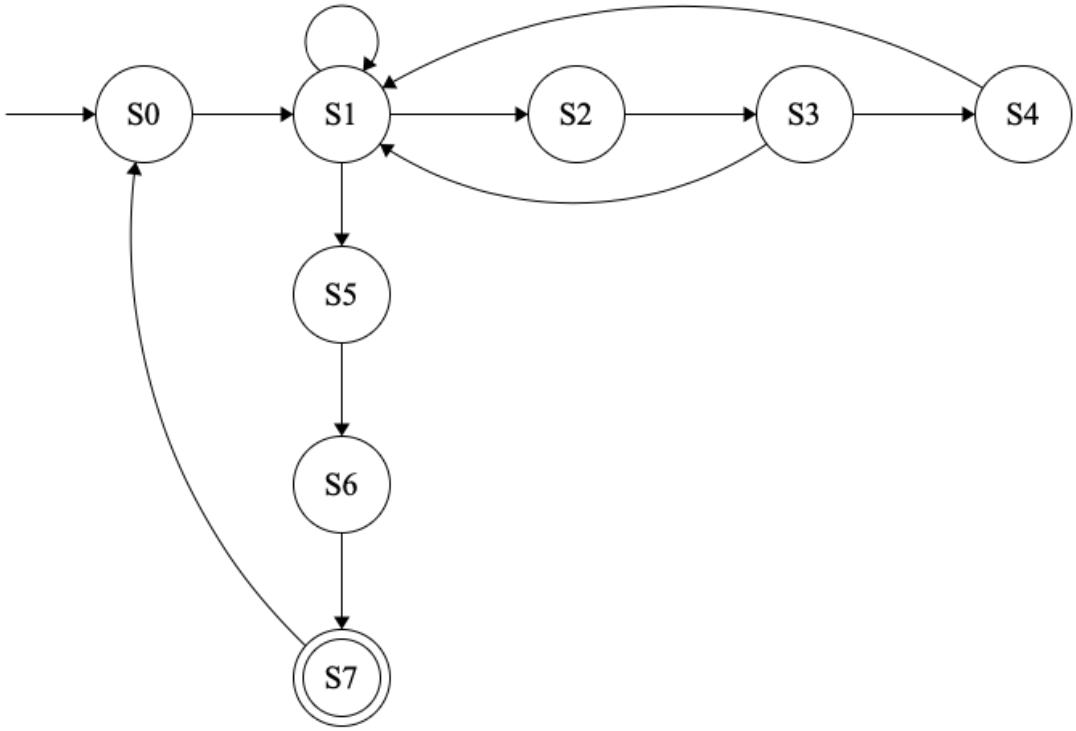
a) Inizio;
b) Gestisci l'indirizzo di lettura della memoria ed invia i segnali alla memoria:
c) Attesa;
d) Leggi un dato dalla memoria;
e) Confronta la distanza dell'ultimo centroide letto con distanza minima trovata fino ad ora e determina la distanza minima;
f) Scrivi il risultato in memoria;
g) Termina.

SCELTE PROGETTUALI

Nella fase di progettazione si è dedicata una particolare attenzione alla divisione dei compiti degli stati per favorire un'agile comprensione e gestione di questi ultimi.
Gli stati implementati e le costanti definite offrono la possibilità di modificare il codice con facilità per eventuali cambiamenti (per esempio alcuni cambiamenti di indirizzo per i dati).
Il segnale `i_rst` permette di portare il componente nello stato iniziale sia se `i_start` è posto a 0 sia se è posto a 1.

IMPLEMENTAZIONE

La macchina a stati finiti implementata è la seguente:



IMPLEMENTAZIONE ARCHITETTURA

Gli stati definiti sono 8 ed un singolo `signal` che tiene traccia dello stato corrente.

```
architecture behavioral of project_reti_logiche is
    type state_type is (S0, S1, S2, S3, S4, S5, S6, S7);
    signal CURRENT_STATE : state_type;
begin
…
end behavioral;
```

IMPLEMENTAZIONE PROCESS

Il process definito ha `i_clk` e `i_rst` nella sua sensitivity list. Le variabili vengono inizializzate nello stato S0. Se il segnale `i_rst` è posto ad 1 la macchina viene portata allo stato S0, qualsiasi stato fosse il precedente. Quando il segnale `i_clk` è posto ad 1 (nello specifico sul fronte di salita) e i_rst è posto a 0 lo stato corrente viene elaborato.

```
process(i_clk, i_rst)

    --Costanti
    constant pointsToConsiderAddress: std_logic_vector(15 downto 0) :=
"0000000000000000"; --Indirizzo maschera con i centroidi da considerare (Indirizzo 0)
    constant numberOfPoints : integer := 8; --Numero di centroidi in memoria
    constant xPointAddress : std_logic_vector(15 downto 0) := "0000000000010001"; --
Indirizzo coordinata X del punto da valutare(Indirizzo 17)
    constant yPointAddress : std_logic_vector(15 downto 0) := "0000000000010010"; --
Indirizzo coordinata Y del punto da valutare(Indirizzo 18)
    constant startingPoint : std_logic_vector(15 downto 0) := "0000000000000000"; --
Primo indirizzo della lista di indirizzi di coordinate: Indirizzo 0 (NOTA: DEVE ESSERE
DI 1 MINORE RISPETTO ALL'INDIRIZZO REALE)
    constant whereToWrite : std_logic_vector(15 downto 0) := "0000000000010011"; --
Indirizzo del risultato (Indirizzo 19)
```

```
    --Variabili
    variable pointsToConsider : std_logic_vector(7 downto 0); --Variabile che contiene
la maschera con i centroidi da considerare (da leggere dalla memoria)
    variable isFirstRound : std_logic; --Variabile che permette di far utilizzare
l'indirizzo 0000000000000000 come pointsToConsiderAddress
    variable currentPoint : integer; --Intero con il numero del centroide analizzato
    variable minDistance : integer range 0 to 510 := 0; --Variabile che contiene la
distanza minima trovata
    variable currentDistance : integer range -1 to 510:= 0; --Variabile che contiene la
distanza dal punto dal centroide analizzato
    variable currentAddress : std_logic_vector(15 downto 0); --Variabile che contiene
l'indirizzo di memoria da utilizzare/appena utilizzato
    variable xPoint : integer range -1 to 255; --Variabile che contiene la coordinata X
del punto da valutare (da leggere dalla memoria)
    variable yPoint : integer range -1 to 255; --Variabile che contiene la coordinata Y
del punto da valutare (da leggere dalla memoria)
    variable currentXPoint : integer range -1 to 255; --Variabile che contiene la
coordinata X del centroide analizzato (da leggere dalla memoria)
    variable currentYPoint : integer range -1 to 255; --Variabile che contiene la
coordinata Y del centroide analizzato (da leggere dalla memoria)
    variable result : std_logic_vector(7 downto 0);

    begin
    if i_rst = '1' then
        CURRENT_STATE <= S0;
    end if;
    if (rising_edge(i_clk) and i_rst = ‘0’) then
        case CURRENT_STATE is
    …
    end if;
    end process;
```

STATO S0

Lo stato S0 si occupa di inizializzare le variabili con i valori necessari per la corretta esecuzione degli stati successivi.

```
when S0 => --Stato d'inizio
            if i_start = '1' then
            --Inizializzazione
            minDistance := 510;
            currentDistance := -1;
            pointsToConsider := "00000000";
            isFirstRound := '1';
            currentPoint := 0;
            currentAddress := pointsToConsiderAddress;
            xPoint := -1;
            yPoint := -1;
            currentXPoint := -1;
            currentYPoint := -1;
            result := "00000000";
            o_en <= '0';
            o_we <= '0';
            o_address <= currentAddress;
            o_done <= '0';
            o_data <= "00000000";
            CURRENT_STATE <= S1;
            end if;
```

STATO S1

Lo stato S1 si occupa di gestire l'ordine degli indirizzi da leggere in memoria ed invia i segnali per la lettura alla memoria. I dati vengono effettivamente letti dalla memoria solo nello stato S3.
Ad `o_address` vengono assegnati in ordine per la lettura gli indirizzi in cui sono memorizzati la maschera dei centroidi da considerare, la coordinata X del punto da valutare e la coordinata Y del punto da valutare; successivamente vengono assegnati gli indirizzi della coordinata X e poi Y per

ogni centroide da considerare fino alla fine del numero dei centroidi (gli indirizzi dei centroidi da non considerare vengono saltati, ciò è possibile in quanto gli indirizzi in memoria sono ordinati come da specifiche a *Pagina 3*).
Si passa allo stato S2 se si legge un dato dalla memoria, se i centroidi da considerare sono finiti si passa allo stato S5 per la scrittura in memoria del risultato.

```
when S1 => --Stato che gestisce l'ordine degli indirizzi da leggere in memoria ed invia i
segnali per la lettura alla memoria

            --Gestione indirizzo della maschera dei centroidi da considerare
            if currentAddress = pointsToConsiderAddress then
                --If necessario se pointsToConsiderAddress è l'indirizzo 0
                if isFirstRound = '0' then
                    currentAddress := xPointAddress;
                end if;

                o_address <= currentAddress;
                o_en <= '1';
                o_we <= '0';
                CURRENT_STATE <= S2;

            --Dopo aver impostato la variabile xPoint viene preparata la lettura di yPoint
            elsif ( currentAddress = xPointAddress and yPoint = -1) then
                currentAddress := yPointAddress;

                o_address <= currentAddress;
                o_en <= '1';
                o_we <= '0';
                CURRENT_STATE <= S2;

            --Dopo aver letto la maschera dei centroidi da considerare e le coordinate del
punto da valutare, lo stato farà scorrere gli indirizzi delle coordinate dei centroidi
            else
                --Gestione degli indirizzi dei centroidi da analizzare
                if currentAddress = yPointAddress then
                    currentAddress := startingPoint;
                    end if;

                --Se ci sono ancora centroidi da analizzare
                if currentPoint < numberOfPoints then

                    --Se il centroide è da considerare (è ad 1 nella maschera) lo analizza
                    if pointsToConsider(currentPoint) = '1' then
                        currentAddress := currentAddress + "0000000000000001";
                        o_address <= currentAddress;
                        o_en <= '1';
                        o_we <= '0';
                        CURRENT_STATE <= S2;

                    --Altrimenti vi è un autoanello ed analizza il centroide successivo
                    else
                        currentAddress := currentAddress + "0000000000000010";
                        currentPoint := currentPoint + 1;
                        CURRENT_STATE <= S1;
                    end if;
                 --Se non ci sono più centroidi da verificare
                 else
                    CURRENT_STATE <= S5;
                end if;
            end if;
```

STATO S2

Lo stato S2 assicura che in uscita dalla memoria ci sia il dato richiesto nel ciclo di clock successivo. Si è scelto di usare uno stato di attesa perché il componente, in questo modo, si può usare sia con memorie che rendono disponibile il dato sul fronte di salita del clock, sia con memorie che lo rendono disponibile sul fronte di discesa.

```
            when S2 => --Stato di attesa per la lettura della memoria
                       CURRENT_STATE <= S3;
```

STATO S3

Lo stato S3 legge il segnale `i_data`, in uscita dalla memoria con il dato richiesto dallo stato S1, e salva il dato nella variabile corrispondente conoscendo il `current_address` usato dallo stato S1. Se ha letto la coordinata Y di un centroide passa allo stato S4, altrimenti torna allo stato S1 per proseguire la lettura della memoria (le variabili `currentXPoint` e `currentYPoint`, che contengono le coordinate X e Y di del centroide corrente, vengono poste a -1 dopo essere state usate nei calcoli dello stato S5 - se durante la lettura delle coordinate dei centroidi `currentXPoint` contiene un valore diverso da -1, allora il valore letto è `currentYPoint`).
Le variabili che contengono coordinate vengono assegnate mediante la conversione, tramite funzioni di libreria IEEE, da std_logic_vector ad integer.

```
when S3 => --Stato per la lettura della memoria e scrittura variabili principali
            o_en <= '0';
            o_we <= '0';

            if currentAddress = pointsToConsiderAddress then
                pointsToConsider := i_data;
                isFirstRound := '0';
                CURRENT_STATE <= S1;

            elsif currentAddress = xPointAddress then
                xPoint := to_integer(unsigned(i_data));
                CURRENT_STATE <= S1;

            elsif currentAddress = yPointAddress then
                yPoint := to_integer(unsigned(i_data));
                CURRENT_STATE <= S1;

            elsif currentXPoint = -1 then
                currentXPoint := to_integer(unsigned(i_data));
                --Deve essere letta anche la coordinata Y del centroide corrente
                CURRENT_STATE <= S1;

            else
                currentYPoint := to_integer(unsigned(i_data));
                --Si passa al confronto
                CURRENT_STATE <= S4;
            end if;
```

STATO S4

Lo stato S4 calcola la distanza del centroide utilizzando le coordinate attualmente contenute nelle variabili `currentXPoint` e `currentYPoint` e le coordinate del punto da valutare salvate nelle variabili `xPoint` e `yPoint`: viene sottratta la coordinata X minore alla coordinata X maggiore ed il risultato viene salvato nella variabile `currentDistance`; a `currentDistance` viene sommata la differenza tra le due coordinate Y con la stessa logica delle coordinate Y. La `currentDistance` così calcolata è la Manhattan distance, definita in geometria come la somma del valore assoluto delle differenze delle coordinate di due punti.
Dopo essere calcolata la `currentDistance` viene confrontata con la variabile `minDistance`:

- `currentDistance` < `minDistance`: la variabile result viene posta a “00000000” e viene posto ad 1 il bit corrispondente al centroide analizzato;
- `currentDistance` = `minDistance`: viene posto ad 1 il bit corrispondente al centroide analizzato nella variabile result;
- `currentDistance` > `minDistance`: lo stato prosegue nell’esecuzione.

Successivamente lo stato assegna -1 alle variabili `currentXPoint` e `currentYPoint`, incrementa la variabile con il numero di centroidi analizzati e torna allo stato S1.

```
when S4 => --Stato che gestisce il confronto

            --Calcola la distanza con il centroide corrente

            if xPoint > currentXPoint then
                currentDistance := xPoint - currentXPoint;
            else
                currentDistance := currentXPoint - xPoint;
            end if;

            if yPoint > currentYPoint then
                currentDistance := currentDistance + (yPoint - currentYPoint);
            else
                currentDistance := currentDistance + (currentYPoint - yPoint);
            end if;

            --Confronta la distanza calcolata con la distanza precedente
            if currentDistance <= minDistance then
                if currentDistance < minDistance then
                    result := "00000000";
                    minDistance := currentDistance;
                end if;
                result(currentPoint) := '1';
            end if;

            --Continua l'analisi degli altri centroidi
            currentXPoint := -1;
            currentYPoint := -1;
            currentPoint := currentPoint + 1;
            CURRENT_STATE <= S1;
```

STATO S5

Lo stato S5 imposta i segnali per la scrittura in memoria.

```
               when S5 => --Imposta i segnali per la scrittura in memoria
                   o_en <= '1';
                   o_we <= '1';
                   o_address <= whereToWrite;
                   o_data <= result;
                   CURRENT_STATE <= S6;
```

STATO S6

Lo stato S6 porta il segnale `o_done` ad 1 (indica la fine dell'elaborazione).

```
                  when S6 => --Porta il segnale o_done ad 1
                              o_en <= '0';
                              o_we <= '0';
                              o_done <= '1';
                              CURRENT_STATE <= S7;
```

STATO S7

Lo stato S7, quando il segnale `i_start` è a 0, porta il segnale `o_done` a 0 (come da specifiche).

```
       when S7 => --Porta il segnale o_done a 0 quando il segnale i_start è a 0
                   if i_start = '0' then
                       o_done <= '0';
                       CURRENT_STATE <= S0;
                   end if;
               end case;
```

RISULTATI SPERIMENTALI

FPGA Target: FPGA xc7a200tfbg484-1.
Il componente è sintetizzabile e supera sia il test bench fornito sia altri ideati per testare casi limite (in pre-sintesi ed in post-sintesi - functional e timing).

REPORT DI SINTESI - TIMING

È stato aggiunto un constraint per il segnale di `i_clk` con un periodo di 100ns per ottenere un report sulle tempistiche con il componente implementato. Di seguito il report ottenuto.

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	87,915 ns	Worst Hold Slack (WHS):	0,140 ns	Worst Pulse Width Slack (WPWS):	49,500 ns
Total Negative Slack (TNS):	0,000 ns	Total Hold Slack (THS):	0,000 ns	Total Pulse Width Negative Slack (TPWS):	0,000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	365	Total Number of Endpoints:	365	Total Number of Endpoints:	141

All user specified timing constraints are met.

SIMULAZIONI

In questa sezione si descrivono di seguito i casi test ideati.

TEST BENCH 1

Esamina tutti i centroidi, i centroidi ed il punto da valutare hanno coordinate (0, 0). Risultato atteso: “11111111”.

TEST BENCH 2

Non esamina alcun centroide, i centroidi ed il punto da valutare hanno coordinate (0, 0). Risultato atteso: “00000000”.

TEST BENCH 3

Non esamina alcun centroide, i centroidi hanno coordinate (0, 0) ed il punto da valutare ha coordinate (255, 255). Risultato atteso: “00000000”.

TEST BENCH 4

Esamina tutti i centroidi, i centroidi hanno coordinate (0, 0) ed il punto da valutare ha coordinate (255, 255). Risultato atteso: “11111111”.

TEST BENCH 5

Esamina tutti i centroidi, i centroidi ed il punto da valutare hanno coordinate (255, 255). Risultato atteso: “11111111”.

TEST BENCH 6

Contenuto in memoria identico al test bench fornito. Dopo 8 cicli di clock il segnale di `i_rst` è portato ad 1 e dopo un ciclo di clock riportato a 0. Risultato atteso: “00010001”.

POSSIBILI OTTIMIZZAZIONI

Si potrebbe ridurre il numero di cicli di clock necessari per l'elaborazione se si facesse elaborare la macchina a stati finiti sul fronte di discesa del clock, così da poter eliminare lo stato di attesa S2, in quanto la memoria da utilizzare rende disponibile il dato sul fronte di salita del clock. La scelta è ricaduta sull'introduzione dello stato di attesa in quanto si è data più importanza alla versatilità del componente.

SYNTHESIZED DESIGN - SCHEMATIC