PROVA FINALE:

PROGETTO DI RETI LOGICHE

Politecnico di Milano, anno accademico 2020-21

Prof. Gianluca Palermo

Indice

Realizzato da:

Pavesi Andrea (cod. persona 10659804)

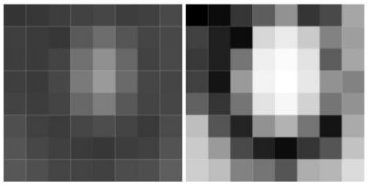
Radaelli Marta (cod. persona 10657046)

POLITECNICO
MILANO 1863

1 Introduzione

La specifica della Prova Finale è ispirata al metodo di equalizzazione dell'istogramma di una immagine.

Il metodo di equalizzazione dell'istogramma è un metodo pensato per ricalibrare il contrasto di un'immagine quando l'intervallo dei valori di intensità sono molto vicini effettuando una distribuzione su tutto l'intervallo di intensità, al fine di incrementare il contrasto.



Original Equalized

Una immagine non equalizzata

La stessa immagine dopo l'equalizzazione dell'istogramma

Lo scopo del progetto è dunque descrivere in VHDL e sintetizzare il componente hardware che possa leggere i dati di un'immagine da una memoria, "equalizzarli" e riscriverli al suo interno.

L'algoritmo da sviluppare è una versione semplificata del metodo spiegato poc'anzi, applicata solo ad immagini in scala di grigi a 256 livelli i cui pixel vengono trasformati nel modo seguente:

```
DELTA_VALUE = MAX_PIXEL_VALUE – MIN_PIXEL_VALUE
SHIFT_LEVEL = (8 – FLOOR(LOG2(DELTA_VALUE +1)))
TEMP_PIXEL = (CURRENT_PIXEL_VALUE - MIN_PIXEL_VALUE) << SHIFT_LEVEL
NEW_PIXEL_VALUE = MIN( 255 , TEMP_PIXEL)
```

Dove MAX_PIXEL_VALUE e MIN_PIXEL_VALUE, sono il massimo e minimo valore dei pixel dell'immagine, CURRENT_PIXEL_VALUE è il valore del pixel da trasformare, e NEW_PIXEL_VALUE è il valore del pixel equalizzato.

Il modulo da implementare dovrà leggere l'immagine da elaborare da una memoria in cui è salvata sequenzialmente e riga per riga. Ogni byte corrisponde ad un pixel dell'immagine.

Le celle 0 e 1 della memoria conterranno le informazioni relative al numero totale di pixel da trasformare, quelle comprese tra 2 e (n_col*n_righe)+1 conterranno in byte contigui i valori dei pixel pre-equalizzazione, mentre le celle da + 2 (n_col*n_righe)+2 i valori post trasformazione.

Il componente da descrivere ha la seguente interfaccia:

```
entity project_reti_logiche is
     port (
          i_clk     : in std_logic;
          i_rst     : in std_logic;
          i_start   : 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;
```

in particolare:

segnali di input:

- i_clk segnale di clock in ingresso.
- i_rst segnale di reset che inizializza la macchina in attesa di un segnale di start.

- i_start segnale di start che avvia il processo.
- i_data segnale che arriva dalla memoria in seguito ad una richiesta di lettura.

segnali di outpu:

- o_done segnale di uscita cge comunica la fine dell'elaborazione.
- o_en segnale di enable che permette la comunicazione con la memoria.
- o_we segnale di write enable verso la memoria, (=1) per scrittura, (=0) per lettura
- o_data segnale di uscita verso la memoria.
- o_address segnale di uscita che manda l'indirizzo alla memoria.

esempio:

Memoria	
0	dim_colonna
1	dim_riga
2	pixel #0
3	pixel #1
4	pixel #2
5	new pixel #0
6	new pixel #1
7	new pixel #2

lettura dei pixel (2->4) -> elaborazione -> riscrittura (5->7

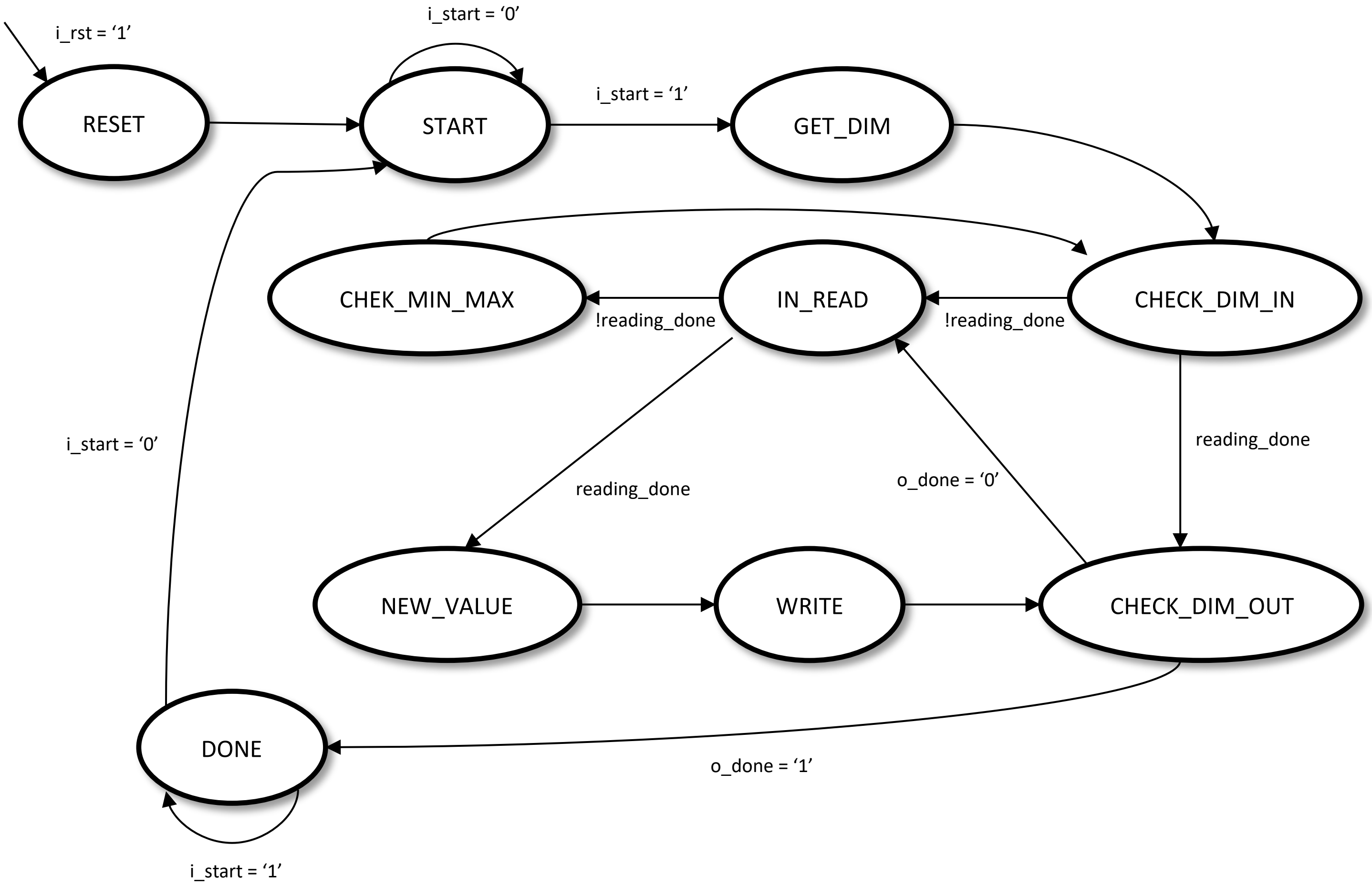
2. Architettura

Abbiamo deciso di approcciare la richiesta con una macchina a stati finiti.

Il funzionamento a cui abbiamo pensato è il seguente:

- lettura dimensioni.
- ciclo di lettura sui pixel per salvare max e min.
- secondo ciclo con rilettura equalizzazione e successiva scrittura.

Nella foto seguente è rappresentata la macchina a stati finiti poi realizzata in vhdl:



START

Stato iniziale in cui si attende il segnale di inizio i_start; in caso venga alzato a 1 il segnale i_rst si ritorna in questo stato, in cui, oltre ad una inizializzazione di alcune variabili, si chiede l'accesso all'indirizzo 0 della memoria.

GET_DIM

Stato utilizzato per salvare l'informazione relativa all'indirizzo 0, cioè la dimensione della colonna,e, successivamente, per accedere all'indirizzo 1 contenente il valore del numero di righe.

CHECK_DIM_IN

Nel caso in cui il booleano reading_done, usato per segnalare la fine della prima lettura dei pixel in memoria, sia false e il contatore sia 2, viene salvato il numero di pixel da analizzare facendo una semplice moltiplicazione riga*colonne. Se reading_done è ancora false, ma il contatore è maggiore di 2, si entra in questo stato per il controllo del numero di pixel letti e successivo aggiornamento dell'indirizzo di memoria da leggere. Una volta letti tutti gli indirizzi iniziali, reading_done viene posto a true e si comincia il vero processo di equalizzazione.

IN_READ

Stato utilizzato per incrementare il contatore e fare un check sul booleano che permette di raggiungere due stati diversi che trattano il valore in input in modi diversi a seconda di quando viene chiamato.

CHECK_MIN_MAX

Stato in cui si controlla se il valore letto è un valore minimo o massimo e, in un caso o nell'altro, si sostituisce il valore in MIN_PIXEL_VALUE o MAX_PIXEL_VALUE con quello nuovo.

CHECK_DIM_OUT

Stato usato per controlli sul secondo ciclo di lettura nella memoria e per il passaggio, nel caso questa sia finita, allo stato finale di DONE.

NEW_VALUE

Stato per il calcolo del valore equalizzato del pixel e per un check sul valore stesso: si seleziona il minimo tra 255 e il numero appena calcolato.

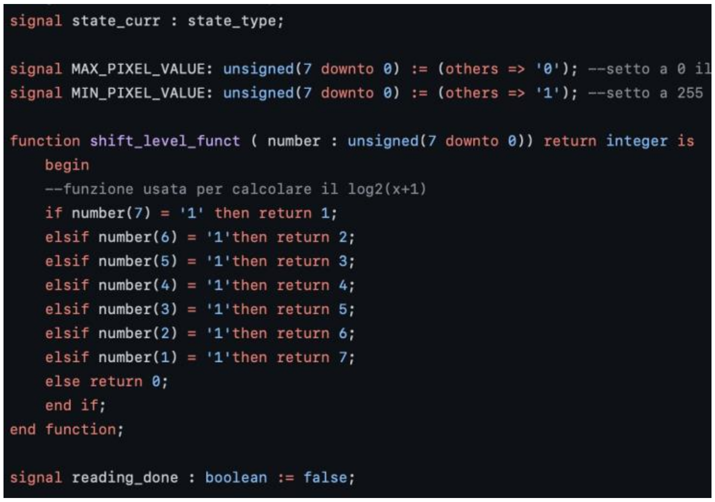
WRITE

Stato per la scrittura del nuovo valore in memoria.

DONE

Stato finale in cui alzare il segnale di done e poi ripassare in START pronti per una nuova elaborazione o restare in done in attesa di un segnale di start pari a zero.

Per quanto riguarda segnali e funzioni utilizzate:



```
signal state_curr : state_type;

signal MAX_PIXEL_VALUE: unsigned(7 downto 0) := (others => '0'); --setto a 0 il
signal MIN_PIXEL_VALUE: unsigned(7 downto 0) := (others => '1'); --setto a 255

function shift_level_funct ( number : unsigned(7 downto 0)) return integer is
    begin
    --funzione usata per calcolare il log2(x+1)
    if number(7) = '1' then return 1;
    elsif number(6) = '1'then return 2;
    elsif number(5) = '1'then return 3;
    elsif number(4) = '1'then return 4;
    elsif number(3) = '1'then return 5;
    elsif number(2) = '1'then return 6;
    elsif number(1) = '1'then return 7;
    else return 0;
    end if;
end function;

signal reading_done : boolean := false;
```

In particolare:

state_curr: segnale che tiene conto dello stato corrente della macchina.

MAX_PIXEL_VALUE: registro che indicano il valore massimo dei pixel dell'immagine.

MIN_PIXEL_VALUE: registro che indicano il valore minimo dei pixel dell'immagine.

shift_level_function: funzione usata per calcolare il valore shift_level, analizzando il valore del pixel in una determinata posizione reimplementando il seguente calcolo 8 - floor(log2(x+1)).

reading_done: booleano che tiene conto se la prima lettura è stata completata.

Una possibile rappresentazione del funzionamento logico della nostra macchina a stati potrebbe essere: