POLITECNICO
MILANO 1863

Scuola di Ingegneria Industriale e
Dell'Informazione

Codice corso 085877- Reti logiche

Relazione del progetto di Reti Logiche 2021

Autore:
Alessandro Arbasino Matricola 909714

Maggio, 2021

Indice

1 Introduzione

1.1 Scopo del progetto

Lo scopo del progetto è quello di descrivere un componente hardware,ovvero un equalizzatore dell'istogramma di immagini di dimensione massima 128x128 pixel. Il componente descritto è simulato grazie al VHDL , riceve in ingresso il valore in scala di grigi del pixel e calcola il nuovo valore usando l'algoritmo di equalizzazione. Questo algoritmo verrà applicato esclusivamente a immagini in scala di grigi con massimo 256 livelli.

```
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)
```

Figura 1: Algoritmo di equalizzazione

1.2 Specifiche Generali

I pixel dell'immagine occupano un byte con il primo bit corrispondente al bit più significativo dello stesso. Essi hanno un valore compreso tra 0 e 255 per un totale di 256 livelli. Il processo di equalizzazione identifica il valore minimo e il valore massimo presenti nell' istogramma dell'immagine e calcola l'intero inferiore del logaritmo in base 2 del valore minimo sottratto al valore massimo. Successivamente questo valore viene sottratto al numero di default 8 per non fare uno shift eccessivo che ecceda i bit del risultato. Come ultimo passo il valore corrente del pixel diminuito del valore minimo verrà moltiplicato per 2(shift sinistro) un numero di volte pari al valore precedentemente trovato. Esso viene confrontato con il massimo valore in scala di grigi accettato (255) e se inferiore esso sarà il nuovo valore del pixel equalizzato altrimenti verrà posto a 255.

1.3 Interfaccia del componente

Interfaccia del Componente

Il componente da descrivere deve avere 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:

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

Figura 2: Uso dei segnali

1.4 Dati e descrizione della memoria

I dati ,ciascuno da 8 bit, sono memorizzati in una memoria con indirizzamento al byte.

- indirizzo 0 usato per il numero delle colonne dell'immagine da equalizzare
- indirizzo 1 rappresenta il numero di righe dell'immagine
- dall'indirizzo 2 all'indirizzo n+1 (con n la dimensione della foto da equalizzare) sono salvati i valori dei singoli pixel che compongono l'immagine.
- dal bit n+2 all'indirizzo 2n+1 verranno memorizzati i valori dell'immagine equalizzata

Numero colonne	indirizzo 0
numero righe	indirizzo 1
valore del primo bit dell'immagine	indirizzo 2
valore del secondo bit dell'immagine	indirizzo 3
valore del terzo bit dell'immagine	indirizzo 4
valore del'ultimo bit dell'immagine	indirizzo n+1
...	
valore del primo bit dell'immagine equalizzata	indirizzo n+2
valore del secondo bit dell'immagine equalizzata	indirizzo n+3
valore del terzo bit dell'immagine equalizzata	indirizzo n+4
...	
valore dell'ultimo bit dell'immagine equalizzata	indirizzo 2n+1

2 Design

Quando il segnale i_start vine portato a 1 il componente inizia la computazione spostandosi dallo state reset (sr) al primo stato della computazione effettiva. Il componente sintetizzato elabora i valori dei singoli pixel indipendentemente dalla dimensione della foto; in questo modo si possono trattare allo stesso modo immagini quadrate,rettangolari e anche rettilinee. Una volta finita la computazione il componente porta il segnale o_done a 1. Il test bench risponde abbassando i_start e il componente riporterà a 0 o_done tornando nello state reset, in attesa che i_start torni alto. Il

componente dispone inoltre di un segnale i_rst che setterà i valori a quelli di default prima dell'inizio di ogni computazione e in caso non si voglia proseguire con la stessa una volta iniziata.

2.1 Stati della macchina

- sr stato di reset della macchina
- colum_state salva il valore del numero colonne
- raw _state salva il numero delle righe
- dim_state calcola la dimensione dell'immagine
- Max_state usato per verificare se i_data è il massimo
- min_state usato per verificare se i_data è il minimo
- address_state usato per calcolare il prossimo o_address
- increment_state usato per incrementare il valore del logaritmo e calcolarne il valore
- shift_state usato per fare lo shift e calcolare temp_pixel
- new_bit_address_state calcola l'indirizzo del nuovo bit per calcolarne il valore equalizzato
- end_state porta o_done a 1 e torna in sr

Questi sono gli stati principali della macchina e sono seguiti da stati di wait per aspettare la memoria e da stati s adibiti a computazioni minori.

2.2 Scelte progettuali

La principale scelta progettuale è stata quella di descrivere il componente in 2 processi

1. il primo processo è delegato alla gestione del cambiamento di stato della macchina a stati e alla gestione del reset della macchina.
2. il secondo processo è utilizzato per compiere la vera e propria computazione e rappresenta la macchina a stati che computerà il risultato finale.

L'algoritmo utilizzato determina il risultato finale utilizzando come operazioni logiche la ADD per incrementare la dimensione della foto, i contatori per i numeri di interazioni, la SUB per decrementare i contatori per calcolare lo shift_level ,l'operazione di concatenazione per implentare il logaritmo e la shift, il cast per il confronto numerico e di conseguenza anche i simboli di ¡, ¿ e =.

3 Risultati dei test

Per verificare il corretto funzionamento del componente sono stati individuati vari corner case:

1. immagine composta da un solo valore per tutti i suoi bit
2. massimo e minimo valore rispettivamente nella prima e ultima posizione dell'immagine o viceversa
3. range massimo di valori con minimo uguale a 0 e massimo uguale a 255
4. caso di una immagine "rettilinea" con una delle due (o entrambe) le dimensioni a 0
5. immagine di dimensione massima (128x128)

4 Conclusioni

4.1 Risultati della sintesi

Il componente sintetizzato supera correttemente la behavioral simulation e la post syntesis functional simulation in tutti i corner case sopra descritti. Qui di seguito è possibile avere un confronto tra i 2 corner case che portano il componente (a parità di numero di immagini) verso una più veloce o più lunga computazione:

- 24.575 nano secondi-tempo di simulazione (behavioral) con immagine "rettilinea"
- 3.345.500 nano secondi-tempo di simulazione (behavioral) con immagine di dimensione massima

Questo risultato è dato dal fatto che il tempo di simulazione cresce linearmente con ilnumero pixel dell'immagine sia in fase di riconoscimento del massimo e del minimo sia durante il calcolo dell valore equalizzato.

4.2 Ottimizzazioni

Non è stata data particolare rilevanza al numero di stati ma si è voluto limitare la difficolta computazionale delle singole operazioni utilizzando l'operazione di concatenamento e calcolando la dimensione della foto tramite somme successive ,senza usare una effettiva moltiplicazione che sarebbe risultata molto più onerosa in termini di risorse utilizzate.