PROVA FINALE (Progetto Reti Logiche)

Anno 2020/2021

Prof. Fabio Salice

Lorenzo Motelli

Codice Persona:10580746

Matricola:890473

Politecnico di Milano

Consegna 1° Settembre 2021

Sommario

1. **Introduzione**

1.1 Scopo del progetto

Lo scopo del progetto è l'implementazione del metodo di equalizzazione di un'immagine. L'equalizzazione è data dalla lettura in ordine di ogni byte dell'immagine.

1.2 Specifiche del progetto

I primi due byte in memoria segnano rispettivamente il numero di colonne ed il numero di righe di cui è composta l'immagine in scala di grigi (quindi un solo byte per pixel basta a rappresentare il colore dell'immagine in quel punto).
I valori dell'immagine equalizzata sono scritti immediatamente dopo quelli dell'immagine originale.
Per valutare l'equalizzazione del singolo pixel si è utilizzato il seguente algoritmo:

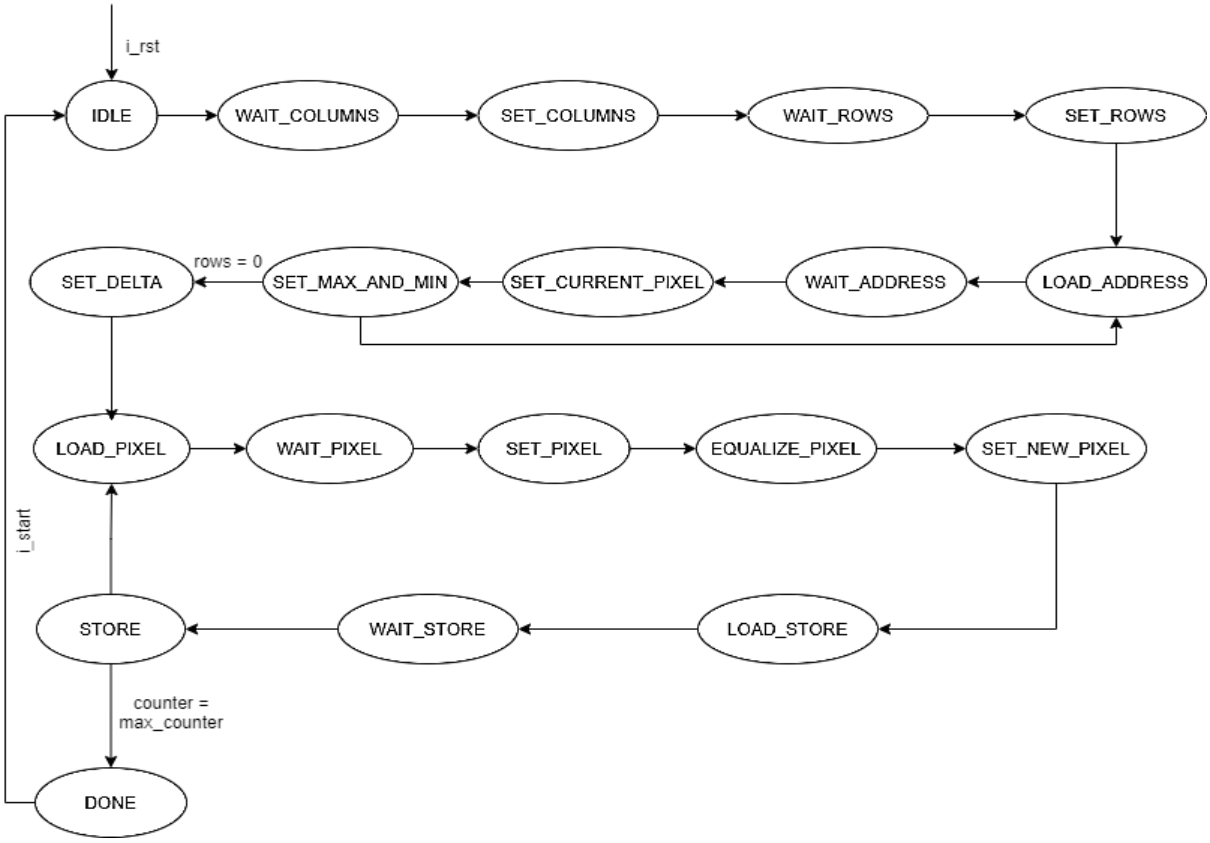
```
DELTA_VALUE = MAX_PIXEL_VALUE  –  MIN_PIXEL_VALUE
SHIFT_LEVEL = (8 – FLOOR(LOG2(DELTA_VALUE +1)))
TEMP_PIXEL = (CURRENT_PIXEL_VALUE  -  MIN_PIXEL_VALUE)  << SHIFT_LEVEL
NEW_PIXEL_VALUE = MIN( 255 , TEMP_PIXEL)
```

2. **Architettura**

2.1 Scelte progettuali

Si è scelto di utilizzare un unico modulo integrante sia la macchina a stati finiti sia l'algoritmo di equalizzazione dell'immagine e che comunichi con la memoria RAM. Inoltre la macchina a stati finiti è stata implementata usando un singolo process, nel quale i segnali di output sono gestiti attraverso istruzioni combinatorie al suo interno, così da minimizzare l'area utilizzata.

2.2 Macchina a stati finiti



2.2.1 IDLE

Inizializzazione di tutti i registri interni, viene precaricato l'indirizzo di memoria iniziale da leggere; appena si legge il livello alto di start inizia la computazione;

2.2.2 WAIT_COLUMNS

Attesa del ciclo di clock per abilitare l'effettivo caricamento dell'indirizzo di memoria;

2.2.3 SET_COLUMNS

Settaggio il valore di colonne che ci sono nell'immagine, inoltre carica l'indirizzo di memoria successivo;

2.2.4 WAIT_ROWS

Attesa del ciclo di clock per abilitare l'effettivo caricamento dell'indirizzo di memoria e setta il valore del registro columns_to_decrease uguale a quelle del registro columns;

2.2.5 SET_ROWS

Settaggio del valore di righe che ci sono nell'immagine. Questo valore, insieme a quello di columns_to_decrease serviranno per il primo ciclo di lettura dell'immagine;

2.2.6 LOAD_ADDRESS

Inizia a caricare il seguente valore della memoria da leggere, partendo dall'indirizzo 3 e aggiungendo il valore di un contatore (inizializzato a zero in IDLE);

2.2.7 WAIT_ADDRESS

Attesa del ciclo di clock per abilitare l'effettivo caricamento dell'indirizzo di memoria e decrementa di uno il valore di columns_to_decrease;

2.2.8 SET_CURRENT_PIXEL

Setta il valore del registro current_pixel, che verrà usato per valutare se è il valore massimo e/o minimo letto dei vari pixel;

2.2.9 SET_MAX_AND_MIN

Valutazione effettiva del registro current_pixel e aumento del valore di counter. Valutazione dei registri columns_to_decrease e rows. Se columns_to_decrease non è zero allora si ritorna a LOAD_ADDRESS visto che la riga non è finita; se invece columns_to_decrease è zero significa che la riga dell'immagine è finita, quindi si decrementa il valore del registro rows. Essendo che la lettura effettiva di rows avverrà al ciclo successivo si valuta se rows è uguale ad 1 e non a 0 per vedere se l'immagine è del tutto finita, poiché la macchina a stati finiti effettua un ciclo in più di operazioni (sarebbero serviti due stati, uno per leggere columns_to_decrease e uno per rows, così facendo invece si è collassato tutto in un unico stato). Una volta che finisce i cicli per leggere del tutto l'immagine (si avrà columns_to_decrease e rows a zero) la macchina passa allo stato SET_DELTA, aggiornando il registro max_counter a counter+1, sempre per la lettura di counter al ciclo dopo, inoltre counter viene ri-settato a zero;

2.2.10 SET_DELTA

Inizializza l'indirizzo di memoria per l'output, uguale a 2 + max_counter e definisce il delta come max_pixel – min_pixel. Il delta verrà usato per lo shift per l'equalizzazione dell'immagine;

2.2.11 LOAD_PIXEL

Caricamento dell'indirizzo di memoria del pixel da equalizzare, partendo dall'indirizzo 2, andando a scalare col valore di counter;

2.2.12 WAIT_PIXEL

Attesa del ciclo di clock per abilitare l'effettivo caricamento dell'indirizzo di memoria;

2.2.13 SET_PIXEL

Settaggio del valore temporaneo del pixel come valore contenuto nell'indirizzo di memoria meno valore minimo di pixel (min_pixel) in un registro a 16 bit (tmp_pixel_16bit);

2.2.14 EQUALIZE_PIXEL

In base a delta si seleziona uno tra gli 8 shift a sinistra possibili, minore è il delta più è alto lo shift da eseguire;

2.2.15 SET_NEW_PIXEL

Dopo lo shift si valutano gli 8 bit più significativi di tmp_pixel_16bit: se sono zero allora non vi è stato overflow, quindi possono salvare gli 8 bit meno significativi nell'indirizzo di memoria designato, che sono il risultato dello shift; se invece gli 8 bit più significativi hanno un valore diverso da zero vi è stato overflow e quindi il nuovo valore del pixel sarà il massimo (255). Inoltre si precarica il valore di memoria di output tramite un registro precaricato e il segnale counter che indica a quale pixel si è arrivati (starting_address_for_equalization + counter);

2.2.16 LOAD_STORE

Carica in o_data il valore di current_pixel determinato nello stato precedente;

2.2.17 WAIT_STORE

Stato nel quale si aspetta il ciclo di clock per finire di settare o_data;

2.2.18 STORE

Si riportano le variabili necessarie per lo shift a zero (tranne delta), si aumenta il valore di counter e si valuta se questo registro è uguale a max_counter, se lo è allora l'equalizzazione dell'immagine è finita, altrimenti bisogna caricare il prossimo indirizzo/pixel (si ritorna a LOAD_PIXEL);

2.2.19 DONE

Il segnale di done viene portato a livello alto per segnalare la fine della computazione, si aspetta il livello logico basso di start per ritornare nello stato di IDLE in modo da poter codificare la seguente immagine.

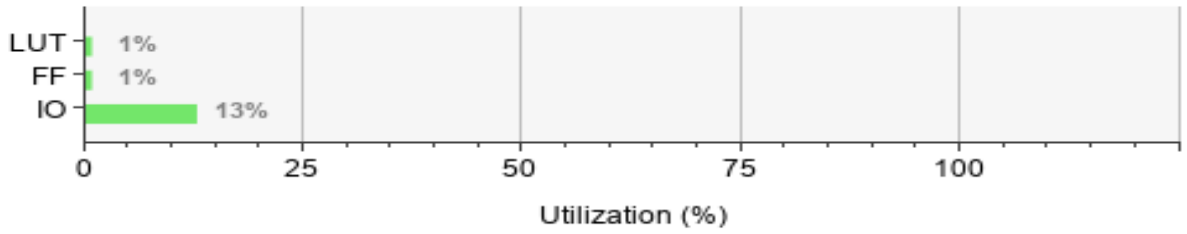
3. Risultati sperimentali

3.1 Risultati della sintesi

Il componente viene correttamente sintetizzato da Vivado e supera entrambe le simulazioni Post-Synthesis (Functional e Timing) con il periodo di clock richiesto dalle specifiche (100 ns).

Di seguito viene riportato un estratto delle parti più significative del report di sintesi riguardanti l'utilizzo dei componenti logici presenti sulla scheda FPGA.

Resource	Utilization	Available	Utilization %
LUT	177	134600	0.13
FF	176	269200	0.07
IO	38	285	13.33



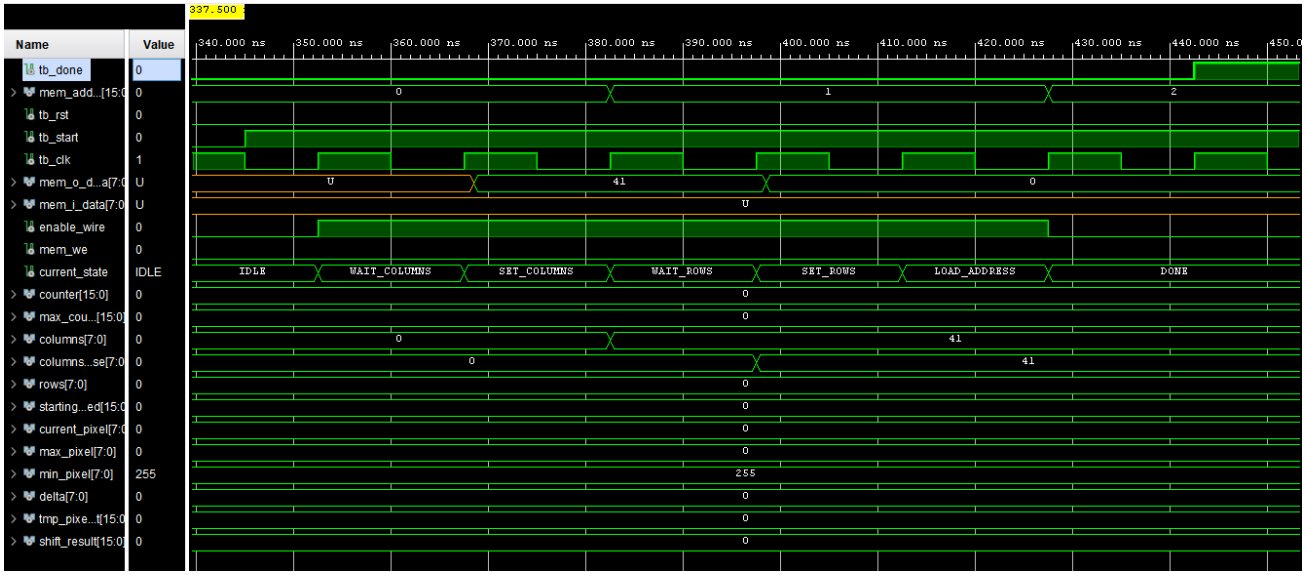
Il modulo utilizza 177 LUT (cioè lo 0.13% di quelle della FPGA assegnata) per implementare la logica dell'equalizzatore, inoltre necessita di 176 registri Flip-Flop (0.07% dei registri totali sulla scheda) per la memorizzazione dello stato della macchina e dei dati per la codifica.

4. Simulazioni

Oltre ai test forniti sono stati fatti dei test specifici per ogni delta e per valutare i casi limite.

4.1 Nessuna colonna/Nessuna riga

Nel caso in cui non vi sia un numero positivo in uno dei due registri iniziali (0 e 1) il programma termina immediatamente dopo il precaricamento del primo registro contenente il primo pixel dell'immagine. Tutti gli altri registri utili al corretto funzionamento dell'algoritmo sono stati inizializzati durante la fase di IDLE.
Questo test verifica che il programma finisce immediatamente se l'immagine ha righe e/o colonne nulle.
A seguito un'immagine esemplificativa della wave form di un'immagine "Nx0".



4.2 Immagine di massima dimensione

Sono stati fatti dei test con l'immagine a dimensione massima (128x128 pixel) con due valori di riferimento: 0 e 255.
Questo test valuta se un'immagine totalmente bianca o nera viene equalizzata nel modo corretto, cioè non viene alterata.
Successivamente sono state testate varie immagini con delta variabile per controllare il corretto funzionamento dell'algoritmo.

5. **Conclusioni**

Il modulo risulta corretto nelle simulazioni Behavioral, Post-Synthesis Functional e Post-Synthesis Timing; inoltre è stato testato anche in Post-Implementation Functional e Post-Implementation Timing, dando ancora risultati corretti