Progetto finale di Reti Logiche

Prof. Gianluca Palermo - Anno di corso 2020-21

Francesco Pastore - Codice persona: 10629332

POLITECNICO
MILANO 1863

Indice

1 Introduzione

Il progetto richiede l'implementazione dell'algoritmo di equalizzazione dell'istogramma di un'immagine. Questo metodo di elaborazione permette di aumentare il contrasto di un'immagine andando a distribuire su tutto lo spetttro, in modo bilanciato, i valori di intensità precedentemente vicini. In particolare viene richiesta l'implementazione di una versione semplificata applicata solo ad immagini in scala di grigi (0-255) e grandi al massimo 128x128 pixel.

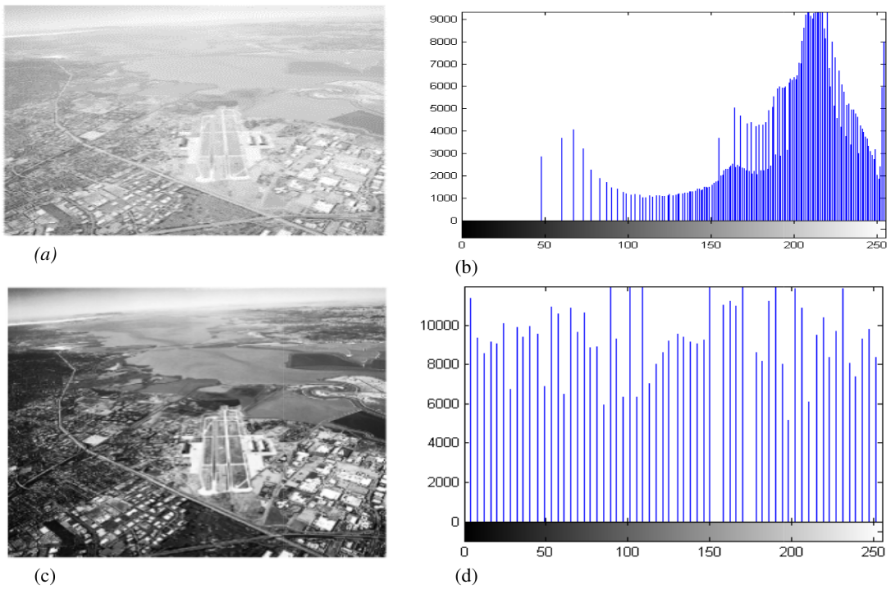

Figura 1: Esempio di equalizzazione dell'istogramma di un'immagine. [**equalization**]

1.1 Specifiche di progetto

L'interfaccia del componente è stata definita nella specifica con i relativi segnali di ingresso e di uscita. Oltre a questo, è stata anche definita la struttura della memoria e l'indirizzamento dei dati. All'indirizzo zero è possibile infatti trovare il numero di colonne, seguito all'indirizzo uno da quello di righe. Dall'indirizzo due iniziano invece i valori dei singoli pixel dell'immagine fino alla posizione NUM_COLS * NUM_ROWS + 1. La scrittura dei pixel equalizzati deve avvenire invece dall'indirizzo immediatamente successivo all'ultimo pixel dell'immagine.

1.2 Algoritmo in breve

Di seguito una breve descrizione dei punti principali dell'algoritmo da implementare. Fare riferimento alle formule per il calcolo dei valori considerati.

1. Lettura del numero di colonne.
2. Lettura del numero di righe.
3. Verificare che l'immagine non sia vuota, altrimenti terminare l'esecuzione.
4. Lettura dei pixel dell'immagine cercando il valore minimo e massimo.
5. Calcolare del delta_value e del relativo shift_level.
6. Lettura dei pixel dell'immagine e calcolo del temp_pixel_value.
7. Calcolare il new_pixel_value considerando il temp_pixel_value trovato al punto precedente.
8. Scrittura in memoria dei nuovi valori dei pixel.

```
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 2: Formule dell'algoritmo di equalizzazione fornite nella specifica.

1.3 Note di implementazione

Rispetto all'algoritmo descritto in precedenza, l'implementazione in VHDL richiede alcune modifiche.

- La lettura dalla memoria non è istantanea, ma necessita di un ciclo di clock di attesa dopo aver effettuato la richiesta. In particolare nel componente è stato implementato lo stato MEM_WAIT.
- L'assegnamento di valori ai segnali non è immediato, ma avviene al ciclo di clock successivo. Per questo motivo alcune operazioni vengono eseguite in più stati successivi.
- Non è possibile assegnare ad un segnale esso stesso seppure con modifiche. Questo richiede per il contatore di utilizzare un altro segnale come appoggio per l'incremento.

2 Architettura

2.1 Segnali utilizzati

2.2 Descrizione degli stati

Il modulo è stato realizzato come una macchina a stati, in particolare comprende 14 stati descritti di seguito.

2.2.1 RESET

Lo stato di RESET è lo stato iniziale della macchina ed è l'unico raggiungibile da tutti gli altri. Quando il componente riceve un segnale di i_rst alto, ferma l'esecuzione e tutto riparte dallo stato di reset. La macchina esce da questo stato solo con il segnale i_start alto.

2.2.2 READ_COLS_REQ

Nel primo byte della memoria è salvato il numero di colonne dell'immagine. Questo stato si occupa di effettuare la relativa richiesta di lettura. Essendo una lettura è necessario attendere che la memoria elabori la richiesta, per questo motivo lo stato successivo è MEM_WAIT.

2.2.3 READ_COLS

Dopo aver effettuato la richiesta di lettura nello stato READ_COLS_REQ in questo stato la macchina legge il numero colonne passatogli dalla memoria nel bus i_data.

2.2.4 READ_ROWS_REQ

Il secondo elemento in memoria dopo il numero di colonne è il numero di righe. Anche in questo caso è necessario effettuare la richiesta di lettura, aspettare un ciclo di clock nello stato MEM_WAIT e solo dopo leggere il valore richiesto.

2.2.5 READ_ROWS

Dopo aver effettuato la richiesta di lettura in READ_ROWS_REQ e aspettato per l'elaborazione da parte della memoria in MEM_WAIT in questo stato viene letto il numero di righe passato al componente tramite i_data.

2.2.6 READ_PIXELS_START

In questo stato viene inizializzato il contatore e i segnali di minimo e massimo prima di effettuare la prima scansione dell'immagine. Il minimo viene settato a 255 che corrisponde al più alto valore possibile, il massimo invece a zero che rappresenta rispettivamente quello più basso. Viene controllato inoltre che non sia stata data un'immagine vuota, altrimenti si passa direttamente allo stato di DONE.

2.2.7 READ_PIXEL_REQ

Dopo aver calcolato il numero di pixel contenuti nell'immagine grazie al numero di colonne e di righe, è possibile leggerli scansionandola dall'inizio alla fine. In questo stato viene quindi settato l'indirizzo per la lettura del prossimo che verrà poi letto nello stato READ_NEXT_PIXEL. Il contatore necessario per la lettura viene incrementato in questo stato sfruttando un segnale temporaneo d'appoggio.

2.2.8 READ_PIXEL

Dopo aver effettuato la richiesta di lettura e aspettato l'elaborazione da parte della memoria, in questo stato la macchina legge il pixel passatole nell'ingresso i_data. Viene inoltre salvato il valore del contatore, che era stato incrementato nello stato precedente sfruttando un segnale d'appoggio.

2.2.9 MEM_WAIT

La memoria richiede un ciclo di clock per l'elaborazione di una richiesta di lettura. Questo stato serve quindi come attesa dopo aver settato o_addr e o_en.

2.2.10 CHECK_MIN_MAX

La prima scansione serve per trovare i valori minimi e massimi dei pixel dell'immagine, in modo da poter poi effettuare l'equalizzazione. In questo stato viene quindi controllato ciascun pixel dopo la prima lettura e confrontato con i valori di massimo e minimo temporanei.

2.2.11 WRITE_START

Una volta effettuata la prima scansione e aver trovato quindi il massimo e il minimo, è possibile calcolare il delta_value dato dalla differenza dei due valori. Tramite uno switch e le relative soglie, viene determinato lo shift_level e il relativo overflow_threshold. Prima di poter effettuare la nuova scansione in questo stato è necessario inizializzare nuovamente il contatore.

2.2.12 EQUALIZE_PIXEL

Per evitare di effettuare più volte lo stesso calcolo, in questo stato viene salvata nel segnale new_pixel_value la differenza tra il valore di ciascun pixel e il relativo minimo dopo ogni lettura.

2.2.13 WRITE_NEW_PIXEL

È in questo stato che la macchina scrive il valore del pixel equalizzato facendo attenzione ad effettuare lo shift solo quando non c'è overflow. A questo fine viene

sfruttato il valore di soglia definito nello stato WRITE_START sulla base del delta_value.

2.2.14 DONE

È lo stato finale in cui giunge la macchina al termine di un'esecuzione completa. Viene settato o_done alto e lo stato successivo è quello di RESET, in modo che il componente rimanga pronto per un'altra possibile esecuzione.

2.3 Diagramma degli stati

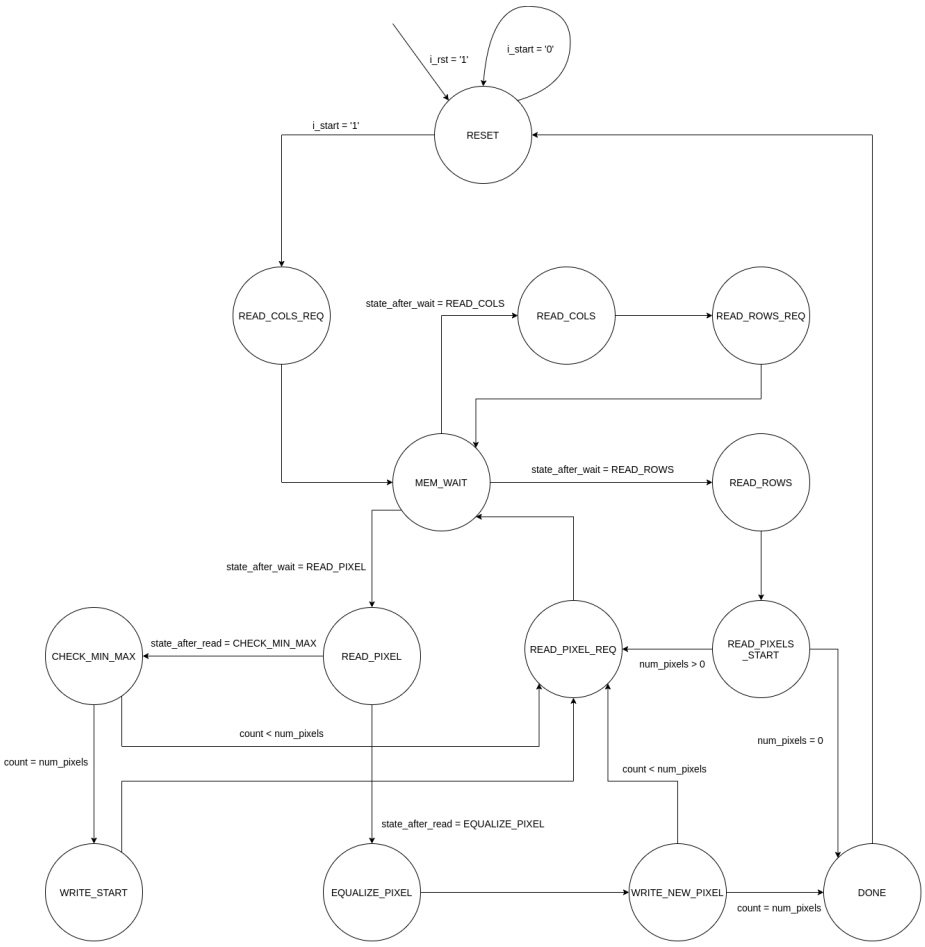

Figura 3: Diagramma della macchina a stati, in figura sono specificate le condizioni in caso di più possibili stati successivi.

3 Risultati sperimentali

3.1 Test con immagine vuota

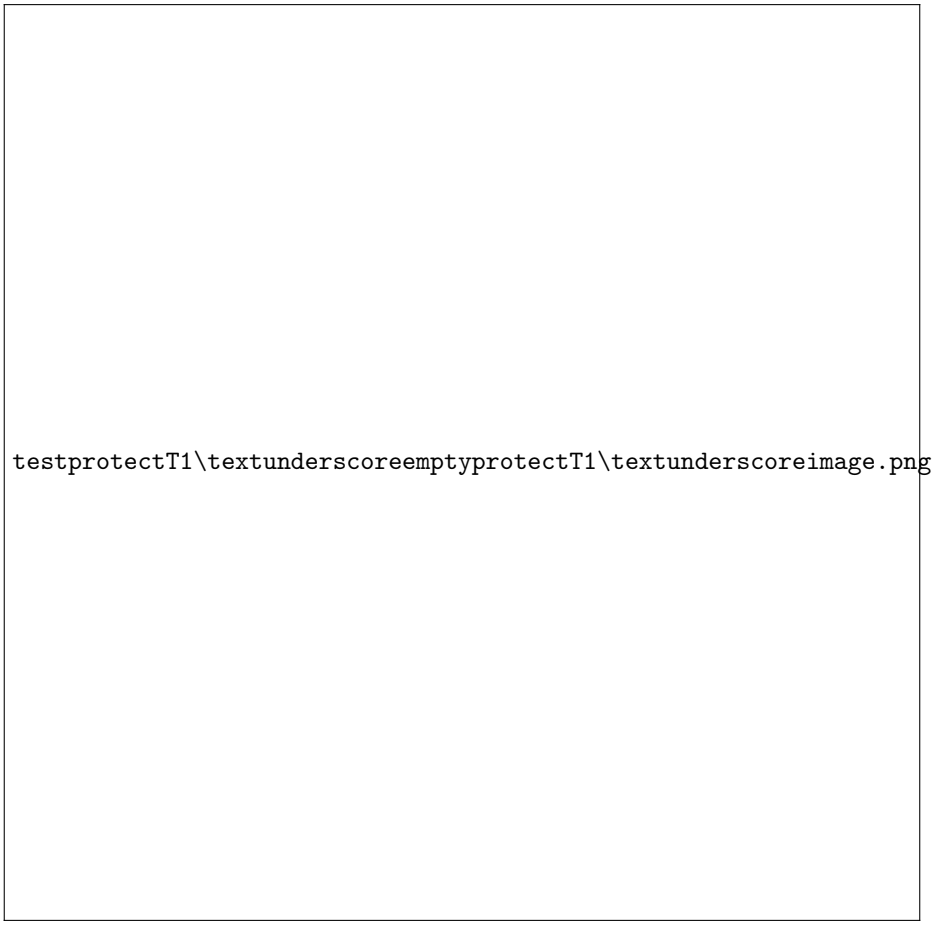

Figura 4: Schema del componente sintetizzato.

3.2 Report di sintesi

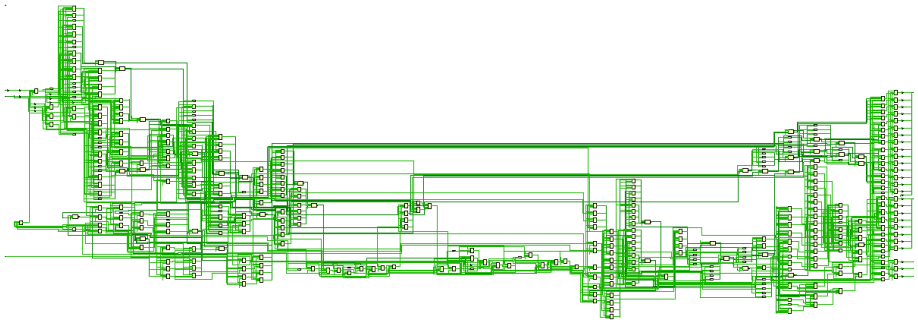

Figura 5: Schema del componente sintetizzato.

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	93,856 ns	Worst Hold Slack (WHS):	0,148 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:	271	Total Number of Endpoints:	271	Total Number of Endpoints:	128

All user specified timing constraints are met.

Figura 6: Dettagli del report di timing del componente.

4 Conclusioni

Possibili ottimizzazioni.

Resource	Utilization	Available	Utilization %
LUT	190	134600	0.14
FF	127	269200	0.05
IO	37	285	12.98

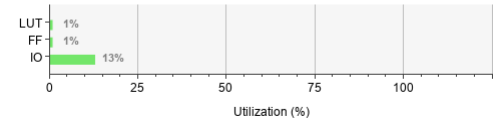

Figura 7: Dettagli del report di utilizzo del componente.