Prova Finale (Progetto di Reti Logiche)

Prof. Fabio Salice – 2020/2021

Gerosa Andrea (10583298)

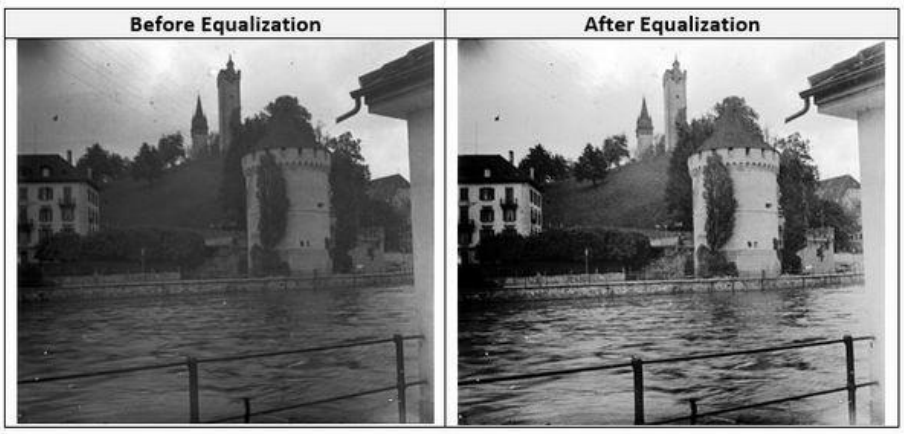
Longaretti Lorenzo (10575993)

Indice

1 Introduzione

1.1 Scopo e specifica

Questo progetto ha lo scopo di implementare un componente hardware con il linguaggio di descrizione VHDL, che realizzi l'equalizzazione dell'istogramma di un'immagine in scala di grigi in ingresso, ridistribuendo i valori d'intensità di ogni pixel in modo che la nuova immagine in uscita abbia un contrasto maggiore.



Le immagini da equalizzare possono avere una dimensione variabile da 1x1 pixel fino ad un massimo di 128x128 pixel e il valore di tonalità di ognuno di essi è contenuto in una memoria indirizzata al byte, a partire dall'indirizzo 2: ogni byte corrisponde infatti ad un pixel, in quanto le immagini su cui il componente lavora sono esclusivamente in bianco e nero. Negli indirizzi 0 ed 1 sono invece contenute rispettivamente la dimensione orizzontale e la dimensione verticale dell'immagine. Una volta che l'immagine è stata elaborata, essa viene scritta nella memoria a partire dall'indirizzo successivo a quello dell'ultimo pixel.

La seguente tabella è esplicativa di un esempio in cui si fa riferimento ad un'immagine 2x2.

Indirizzo di memoria (dec)	Valore del pixel (dec)	Note
0	2	Numero di colonne
1	2	Numero di righe
2	128	Primo pixel
3	0	Secondo pixel
4	34	Terzo pixel
5	236	Quarto pixel
6	255	Primo pixel scritto
7	0	Secondo pixel scritto
8	68	Terzo pixel scritto
9	255	Quarto pixel scritto
10	0	Primo indirizzo libero

L'operazione viene svolta con un algoritmo di equalizzazione in forma semplificata, illustrato di seguito in ogni suo passo:

```
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)
```

1) Viene calcolato un valore, che chiameremo delta_value, come la differenza tra i valori massimo e minimo dei pixel dell'immagine.
2) Viene determinato shift_level, ovvero il numero di bit di cui la posizione delle cifre deve essere spostata.
3) Viene sottratto dal valore corrente del pixel il valore minimo e sul risultato ottenuto viene eseguito uno shift verso sinistra di shift_level, così da calcolare il valore del pixel.
4) Se il risultato temporaneo eccede 255, che è il valore massimo che un pixel può assumere, gli si assegna tale valore. Altrimenti esso diventa il risultato finale del pixel equalizzato.

1.2 Interfaccia

L'interfaccia del componente è definita come di seguito:

```
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;
```

Il modulo e la memoria RAM comunicano attraverso dei segnali. La comunicazione viene abilitata tramite il segnale o_en, che se posto ad '1' permette lo scambio di segnali.
In particolare la scrittura sulla RAM viene concessa dal segnale o_we quando il suo valore è '1', mentre quando esso è '0' si abilita la lettura.
o_address è il segnale in uscita che rappresenta un indirizzo di memoria.
i_data e o_data sono i vettori che contengono rispettivamente i dati richiesti alla memoria e quelli da scrivere in essa.
I segnali generati nei test bench sono il segnale di clock i_clk, il segnale di reset i_rst (che serve ad inizializzare la macchina mettendola nelle condizioni di iniziare a lavorare) e il segnale di start i_start.
C'è infine il segnale o_done, che notifica il completamento dell'elaborazione.

2 Architettura

2.1 Scelte progettuali

Il componente hardware da noi descritto si compone di un singolo processo, nel quale vengono svolte tutte le operazioni necessarie per ottenere il risultato. Questo approccio ci è sembrato più adatto al problema in esame.

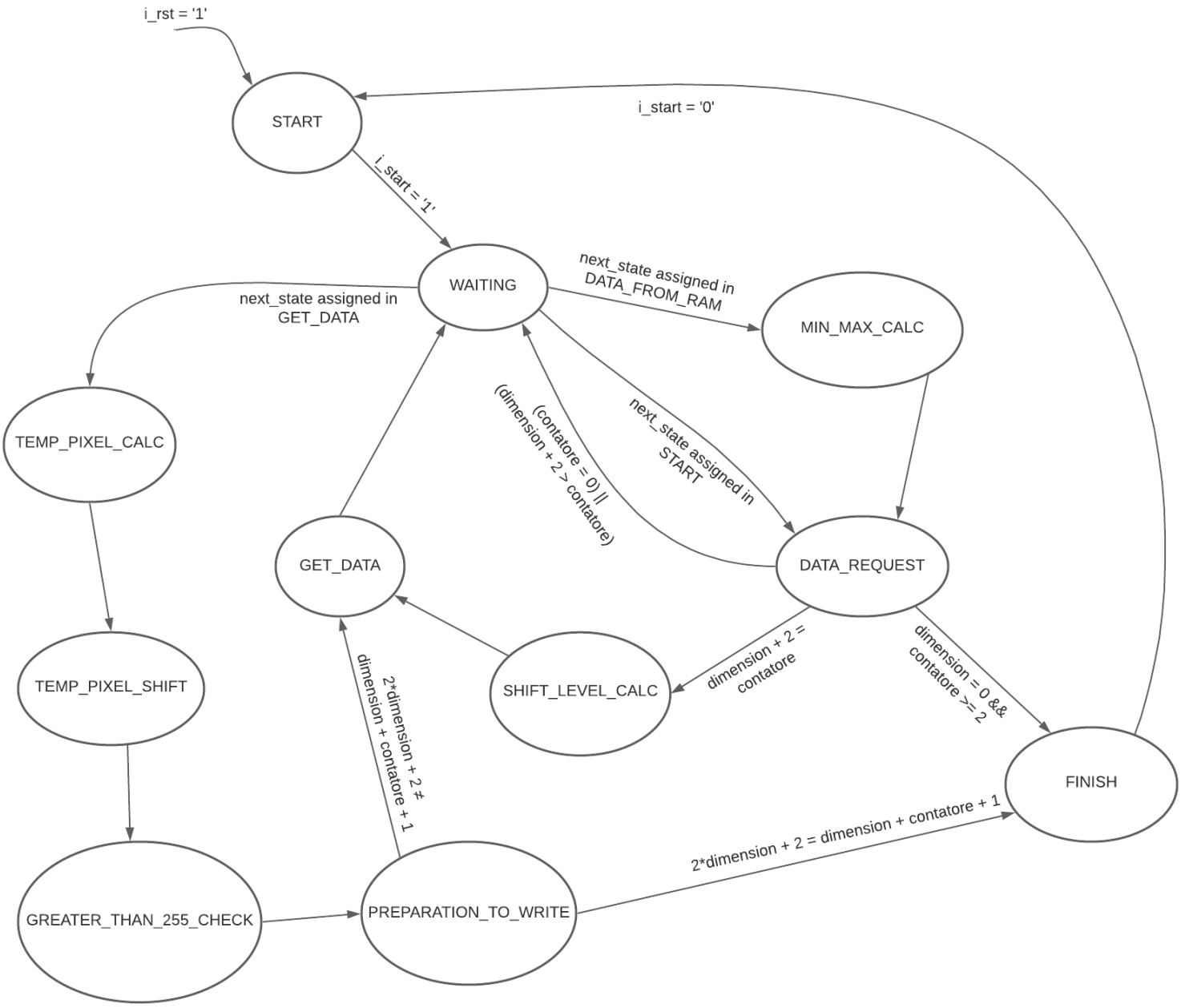
Abbiamo infine fatto in modo che ad ogni ciclo di clock venisse controllato il valore del segnale di reset inserendolo nella sensitivity list del process, così che il componente potesse gestire eventuali segnali asincroni inviati dai test bench.

2.2 Signal interni

Abbiamo dichiarato i seguenti segnali interni all'architettura:

- *current_state:* segnale che memorizza lo stato corrente.
- *next_state:* segnale di supporto che memorizza uno stato futuro.
- *dimension:* vettore di 16 bit per memorizzare la dimensione dell'immagine.
- *contatore*: vettore di 16 bit usato come supporto per rappresentare la posizione degli indirizzi di memoria
- *max_number:* vettore di 8 bit usato per contenere il massimo valore tonale; viene inizializzato a 255.
- *min_number:* vettore di 8 bit usato per contenere il minimo valore tonale; viene inizializzato a 0.
- *shift_level:* vettore di 8 bit per memorizzare il valore dello shift_level calcolato nell'algoritmo.
- *temp_pixel:* vettore di 16 bit che viene usato per contenere i valori temporanei che il pixel elaborato assume durante lo svolgimento dei passi dell'algoritmo.
- *delta:* vettore di 8 bit per memorizzare il valore del delta_value calcolato nell'algoritmo.

2.3 Macchina a stati finiti



2.4 Stati

2.4.1 START state

È lo stato iniziale in cui si attende che il segnale i_start venga alzato ad '1' e in cui si inizializzano i segnali necessari. Viene inoltre richiesto il dato nell'indirizzo di memoria 0. Quando il segnale di reset i_rst viene posto ad '1' si riparte da questo stato.

2.4.2 WAITING state

Stato di attesa utile in diversi momenti dell'esecuzione per attendere che passi il ciclo di clock necessario a poter leggere i dati dalla RAM. Inoltre lo stato corrente viene cambiato in quello precedentemente settato nel signal next_state. (nella descrizione dei successivi stati, WAITING non verrà citato per chiarezza espositiva; si farà riferimento direttamente allo stato salvato in next_state).

2.4.3 DATA_REQUEST state

Stato in cui vengono svolti dei controlli sui segnali contatore e dimension e vengono di conseguenza intraprese diverse azioni:

- Se contatore = 0, viene eseguita la lettura del numero di colonne che era stato richiesto in START e viene fatta la richiesta del dato nell'indirizzo di memoria successivo.
- Se contatore = 1, viene letto il numero di righe richiesto e calcolata la dimensione dell'immagine.
- Se l'immagine ha una dimensione di 0 pixel viene effettuato il reindirizzamento a FINISH.
- Se invece l'immagine è valida, vengono prima fatti scorrere gli indirizzi di memoria contenenti i dati dei pixel, con reindirizzamento, di volta in volta, allo stato MAX_MIN_CALC, in cui vengono determinati i valori minimo e massimo. Quando i dati sono stati tutti letti, viene svolto il calcolo del delta_value come differenza dei valori massimo e minimo (primo passo dell'algoritmo), viene reimpostato il signal contatore al valore 2 e viene fatto il reindirizzamento a SHIFT_LEVEL_CALC.

2.4.4 MAX_MIN_CALC state

Stato con il compito di svolgere controlli sui valori dei pixel letti e di trovare ogni volta il massimo e il minimo valore di tonalità dell'immagine.

2.4.5 SHIFT_LEVEL_CALC state

Viene qui effettuato il calcolo dello shift_level (secondo passo dell'algoritmo) tramite la funzione "floor": `SHIFT_LEVEL = (8 - FLOOR(LOG2(DELTA_VALUE + 1)))`

Siccome shift_level è dipendente solo da delta_value, abbiamo deciso di quantizzare la funzione floor, osservando i valori costanti che assume in funzione di delta_value e impostando direttamente lo shift_level con una serie di controlli if-else, secondo la seguente tabella.

Delta Value	FLOOR(x)	Shift Level
0	0	8
1 - 2	1	7
3 - 6	2	6
7 - 14	3	5
15 - 30	4	4
31 - 62	5	3
63 - 126	6	2

127 - 254	7	1
255	8	0

2.4.6 GET_DATA state

In questo stato vengono di nuovo richiesti alla RAM i dati corrispondenti ai pixel dell'immagine. Lo stato viene poi reindirizzato a TEMP_PIXEL_CALC.

2.4.7 TEMP_PIXEL_CALC state

Stato in cui viene calcolata la differenza tra il valore attuale e il valore minimo precedentemente salvato (terzo passo dell'algoritmo).

2.4.8 TEMP_PIXEL_SHIFT state

Stato in cui viene shiftata verso sinistra di shift_level la differenza calcolata nello stato precedente. Per effettuare questa operazione abbiamo utilizzato la funzione "shift_left", che prende come parametri d'ingresso il risultato ottenuto in TEMP_PIXEL_CALC e il valore dello shift_level.

2.4.9 GREATER_THAN_255_CHECK state

In questo stato (che corrisponde al quarto passo dell'algoritmo) viene effettuato un controllo sul valore ottenuto in TEMP_PIXEL_SHIFT: se esso supera 255, allora gli viene assegnato proprio tale numero, che è il più alto valore di tonalità che il pixel può raggiungere.

2.4.10 PREPARATION_TO_WRITE state

Il pixel equalizzato viene scritto in memoria.

Dopodiché viene effettuato un controllo sulla posizione raggiunta con i dati scritti in memoria: se coincide con quella finale attesa, allora l'equalizzazione dell'immagine è stata completata e si va allo stato FINISH; in caso contrario, ci sono altri pixel da elaborare e si ritorna allo stato GET_DATA incrementando il contatore.

2.4.11 FINISH state

Stato finale in cui si attende che i_start si abbassi a '0', per poter così abbassare o_done e tornare nello stato di START.

3 Risultati sperimentali

Il componente che abbiamo descritto è risultato essere correttamente sintetizzabile ed implementabile. La sintesi richiede l'utilizzo di 277 LUT e di 109 FF.

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF
synth_1	constrs_1	synth_design Complete!								277	109
impl_1	constrs_1	route_design Complete!	NA	NA	NA	NA	NA	7.655	0	216	109

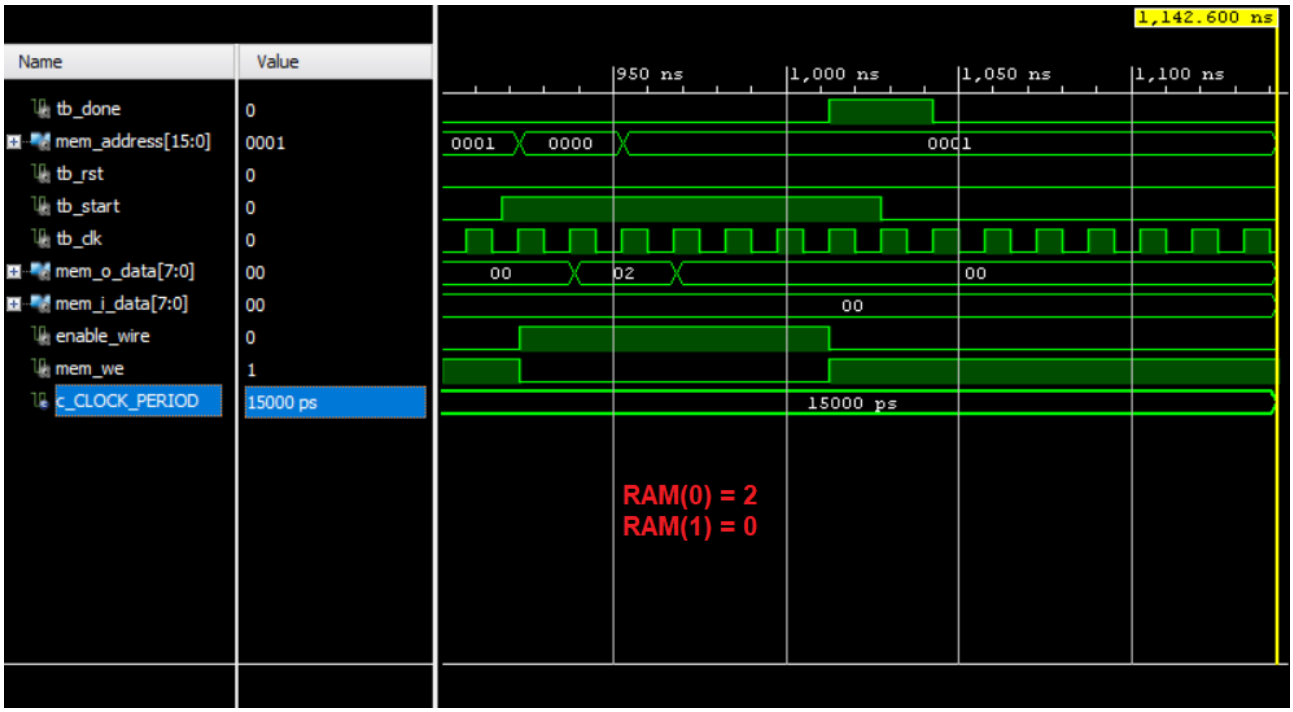
Tutti i test bench utilizzati nel progetto, che analizzeremo nel dettaglio nella sezione successiva, hanno passato la simulazione post-synthesis functional.

4 Simulazioni

Abbiamo effettuato diverse simulazioni per verificare il corretto funzionamento del componente hardware che abbiamo descritto.

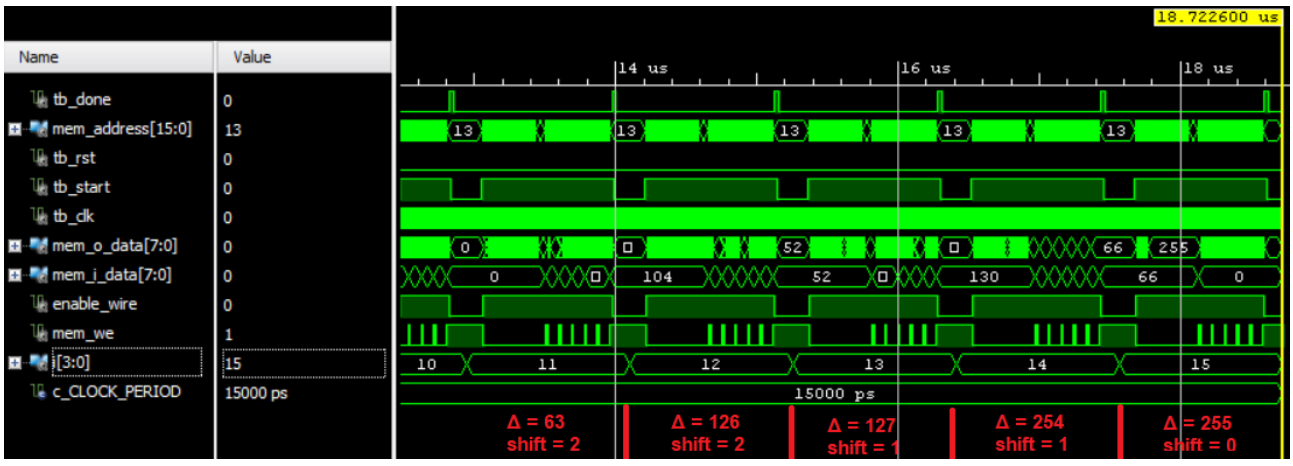
Oltre al test bench fornitoci dal docente, che rappresenta una situazione generica e testa perciò il funzionamento in condizioni standard, abbiamo scritto ulteriori test bench che cercassero di coprire i casi limite e le configurazioni particolari dell'immagine da equalizzare o che testassero il comportamento del componente in risposta a determinati segnali. Di seguito sono illustrati i vari test bench, dei quali i più significativi sono accompagnati dal grafico della simulazione.

- Immagine con dimensione 0: questo test verifica il comportamento del componente di fronte ad un'immagine avente una o entrambe le dimensioni uguali a zero. In questo caso, è necessario che nella memoria RAM non venga scritto nulla e che si ritorni allo stato iniziale, in modo che possa iniziare l'elaborazione di eventuali nuove immagini in ingresso.



- Immagini multiple: questo test verifica il corretto funzionamento del componente a fronte di una serie di immagini in input. Una volta conclusa l'elaborazione della prima e dopo essere tornato nelle condizioni di partenza, il modulo deve poter ricevere l'immagine successiva ed iniziare la sua elaborazione.

- Valori "di frontiera" di delta_value: anche questo test bench verifica l'elaborazione di più immagini, come il precedente. Vengono però testate delle configurazioni particolari: dato che i valori di shift_level sono quantizzati, abbiamo eseguito almeno due elaborazioni per ognuno dei valori di shift_level, uno per ogni estremo del range dei valori che può assumere delta_value (si veda la tabella nella sezione 2.4.5).



- Reset asincrono: in una situazione in cui il segnale di reset venga portato al valore alto nel mezzo dell'elaborazione, bisogna assicurarsi che il componente si fermi e riparta dallo stato iniziale, ripetendo da capo l'operazione ed ottenendo comunque il risultato aspettato.

- Immagine con grandezza massima (128x128): abbiamo scritto anche un test che richiede l'elaborazione di un'immagine avente la massima dimensione possibile e controlla la correttezza dei limiti.

- Immagine con grandezza minima (1x1): semplice test speculare al precedente, che testa la più piccola immagine che può essere fornita in input, il cui valore dell'unico pixel verrà sempre trasformato in '0', come conseguenza del valore '0' di delta_value.

5 Conclusioni

Il componente che abbiamo descritto ha superato tutti i casi di test proposti e si è dimostrato affidabile e preciso nello svolgimento del compito a cui è adibito. L'istogramma delle immagini fornite in ingresso viene ricalibrato correttamente secondo l'algoritmo di equalizzazione sia nelle situazioni generiche sia nei casi più particolari.

Riteniamo quindi di aver rispettato la specifica fornitaci, ottenendo un buon risultato, conforme alla richiesta.