****

Progetto di Reti Logiche

Anno Accademico 2020-2021

Equalizzazione dell’istogramma di una immagine in scala di grigi

*Autori:*

Simone Giampà matricola 909739

Francesco Leone matricola 910196

**Introduzione e Specifica del Progetto**

Il metodo dell’equalizzazione dell’istogramma è utilizzato nell’ambito del processamento delle immagini, per l’ottenimento di immagini con un contrasto migliorato. Il metodo incrementa il contrasto globale di molte immagini, e risulta utile specialmente quando i dati delle immagini sono rappresentati da valori numerici molto ravvicinati, quindi immagini a basso contrasto globale. Attraverso questa modifica, l’intensità può risultare meglio distribuita sull’istogramma complessivo. In questo modo le aree a basso contrasto locale guadagnano un contrasto più elevato. Il metodo dell’equalizzazione dell’istogramma riesce ad ottenere questi risultati in modo abbastanza efficace estendendo i valori di intensità più frequenti.

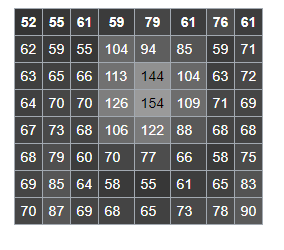
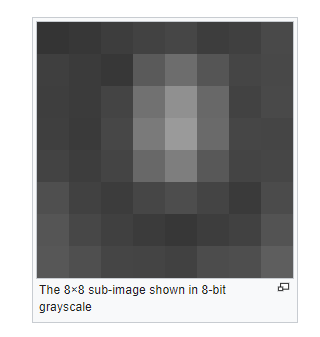


Immagine in scala di grigi a 8 bit, e la sua corrispondente rappresentazione numerica decimale

Ogni immagine è rappresentata in una scala di grigi a 8 bit, quindi ogni pixel assume un valore numerico compreso tra 0 e 255, dove allo 0 corrisponde un pixel nero, e al 255 un pixel bianco. Ogni pixel viene rielaborato e viene prodotto un nuovo valore numerico a partire dal precedente, sempre nella stessa scala di grigi.

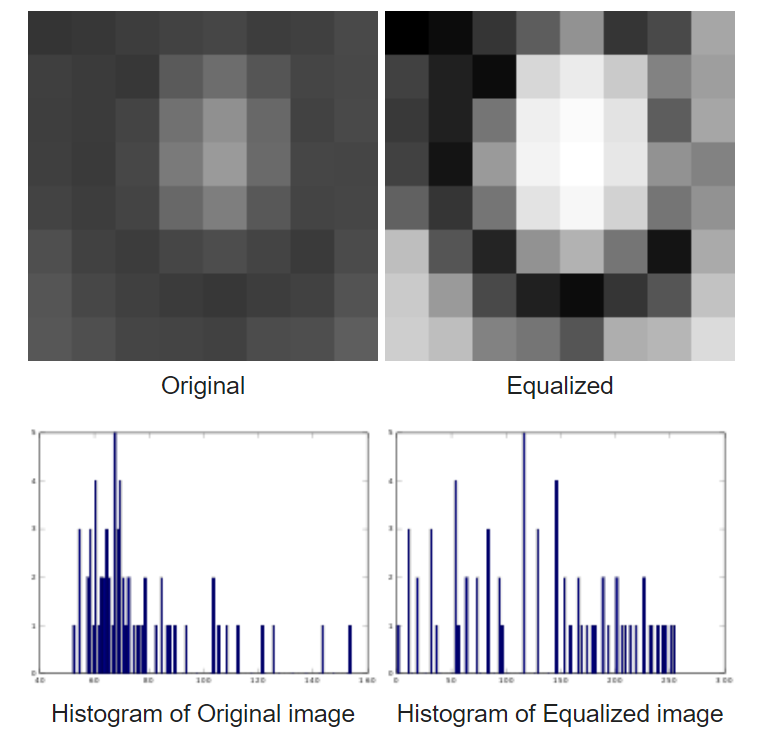
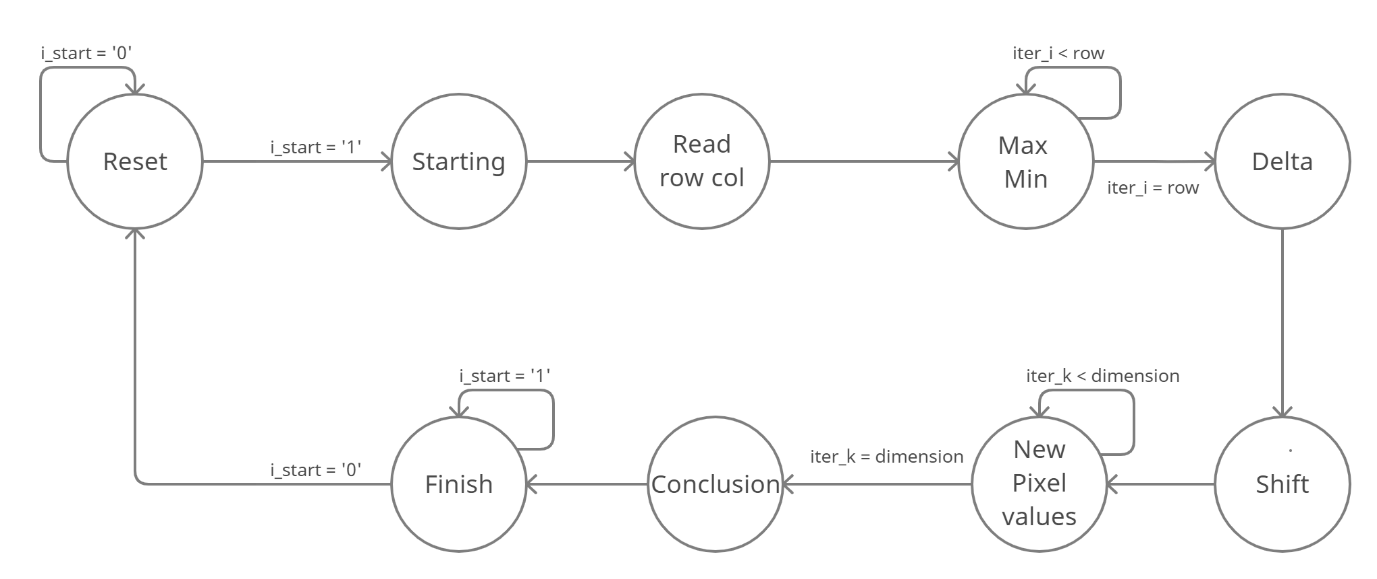


Immagine originale e relativo istogramma di rappresentazione a sinistra; accanto l’immagine equalizzata e il nuovo istogramma

Fonte: [Histogram Equalization - Wikipedia](https://en.wikipedia.org/wiki/Histogram_equalization)

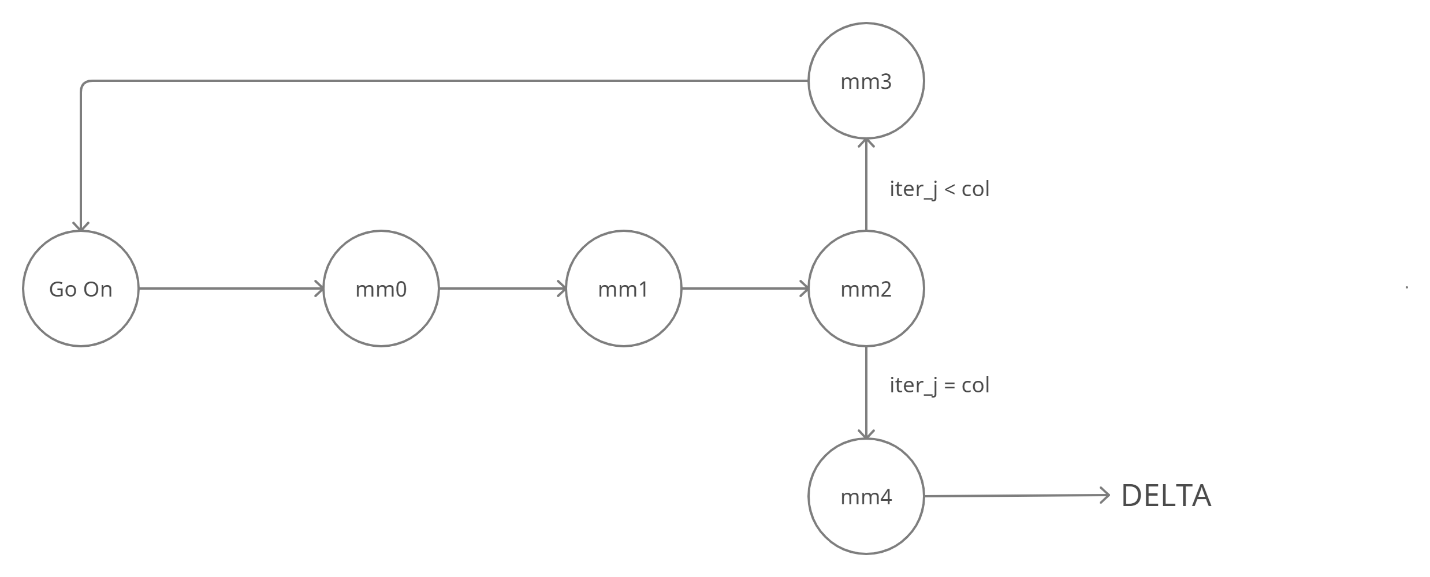
Descrizione Progettuale

L’approccio usato nel progetto è stato quello della macchina a stati. Nella figura che segue c’è una rappresentazione ad alto livello della macchina in cui alcuni stati sono stati implementati con più di uno stato. Il progetto è stato realizzato creando una sola architettura e una sola entity. La macchina a stati è sincronizzata sul fronte di salita del clock. Lo stato di reset può essere attivato alla fine dell’elaborazione di un immagine oppure in un qualunque momento dell’esecuzione in tal caso l’esecuzione si interrompe e la macchina torna nello stato di reset.



Descrizione degli stati

* **Starting***:*Viene caricato il primo indirizzo della memoria.
* **Read Row Col:**  Vengono letti i primi due indirizzi di memoria salvando i valori di righe e colonne.
* **Max Min:** Viene scansionata tutta la matrice alla ricerca di massimo e minimo che vengono salvati in due registri.
* **Delta:** Viene calcolata la differenza tra massimo e minimo e la dimensione in pixel della matrice.
* **Shift:** Viene calcolato lo shift value con dei controlli a soglia.
* **New pixel values:**  Viene calcolato il nuovo valore di ogni pixel e viene scritto in memoria. Le iterazioni terminano quando iter\_i diventa uguale alla dimensione dell’immagine.
* **Conclusion:** Viene impostato ad alto il segnale *o\_done*.
* **Finish:** Si attende che il segnale *i\_start* torni a 0 per processare un’altra immagine.

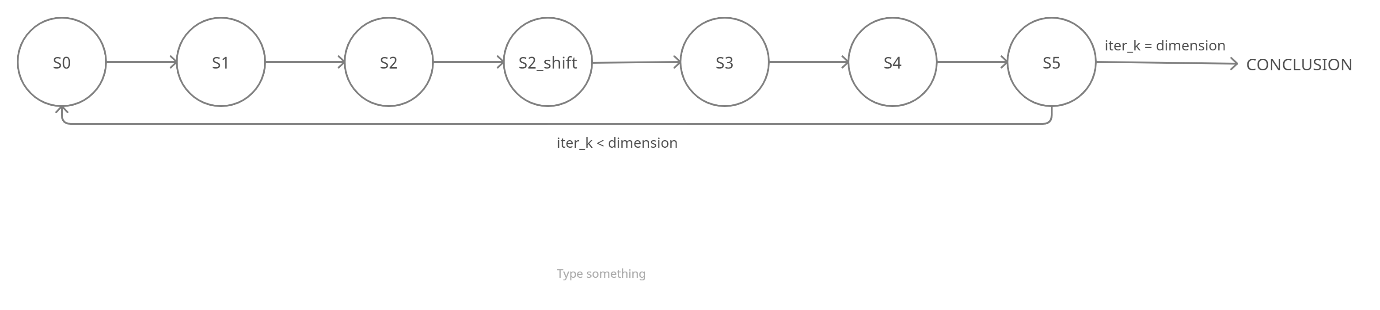
Ricerca di massimo e minimo

iter\_i = row

iter\_i < row

* **MM0:** Viene letto un nuovo pixel dalla memoria.
* **MM1:** Viene confrontato il pixel letto con massimo e minimo e se necessario si aggiornano i valori di massimo e minimo.
* **MM2**: Va in MM3 per leggere i pixel della stessa riga e in MM4 quando deve iniziare a leggere una riga nuova.
* **MM3:** Viene letto un nuovo pixel.
* MM**4**: Se è stata letta tutta l’immagine si procede nello stato DELTA altrimenti si inizia a leggere una nuova riga.

Scrittura in Memoria dell’immagine equalizzata



* **S0:** Viene letto un pixel.
* **S1:** Viene incrementato l’indirizzo di lettura.
* **S2:** Viene calcolata la differenza tra il pixel letto e il valore minimo.
* **S2\_shift:** Al pixel viene applicato lo shift e viene salvato il nuovo valore.
* **S3:** Viene abilitato il segnale *o\_we* e impostato l’indirizzo della memoria in cui scrivere il nuovo valore del pixel.
* **S5:** Viene scritto su *o\_data* il nuovo valore del pixel. Se l’immagine è stata completamente scritta si passa allo stato CONCLUSION altrimenti si torna in S0.

Risultati Sperimentali

Simulazioni del circuito sintetizzato

Sono stati effettuati vari test per stressare il circuito sintetizzato in vario modo. La FPGA utilizzata per la simulazione e la sintesi del circuito è xc7a200tfbg484-1, della famiglia Artix-7. Di seguito sono illustrati vari testbench e relativi risultati di simulazione, con particolare focus ai casi limite.

**Testing Principale**

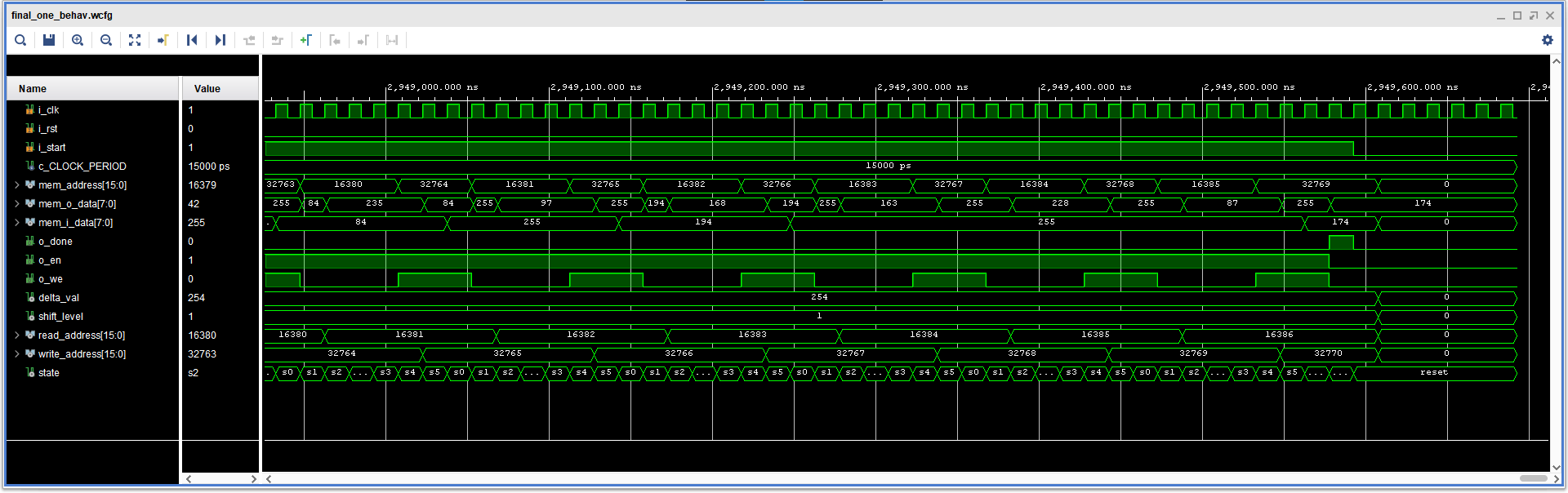
Il test 1 è un test bench generico e semplice anche da controllare manualmente, che dà in ingresso alla memoria una immagine di 4x3 pixels. Con un clock da 15 ns, il tempo totale di esecuzione risulta di circa 2.7 μs.

Graphical user interface

Description automatically generated

Segnali per la simulazione di un test bench generic, immagine 4x3

Il test 2 è un test volto a simulare il processamento di una immagine di risoluzione massima, ovvero 128 x 128 pixels. Il tempo di esecuzione è significativamente alto, comparato ad altre immagini di risoluzione minore, e si riporta solo la parte finale della simulazione.



Segnali per la simulazione di un test bench con memoria inizializzata con una matrice di 128 x 128 pixels

Il test 3 è opposto al test 2, in quanto il suo obiettivo è di testare il comportamento del circuito nel caso la matrice in ingresso alla memoria sia costituita da un singolo pixel (matrice 1x1).

Graphical user interface

Description automatically generated with medium confidence

Simulazione di un test bench con memoria inizializzata con una matrice 1 x 1 contenente un singolo pixel

Altri testbench che abbiamo simulato danno in ingresso alla memoria matrici costituite da valori numerici molto ravvicinati tra di loro (così da rappresentare immagini a basso contrasto globale), e matrici con valori numerici estremi, ovvero 0 oppure 255 (così da rappresentare immagini a massimo contrasto globale possibile, e già equalizzate). In tutti i testbenches simulati, i risultati sono corretti.

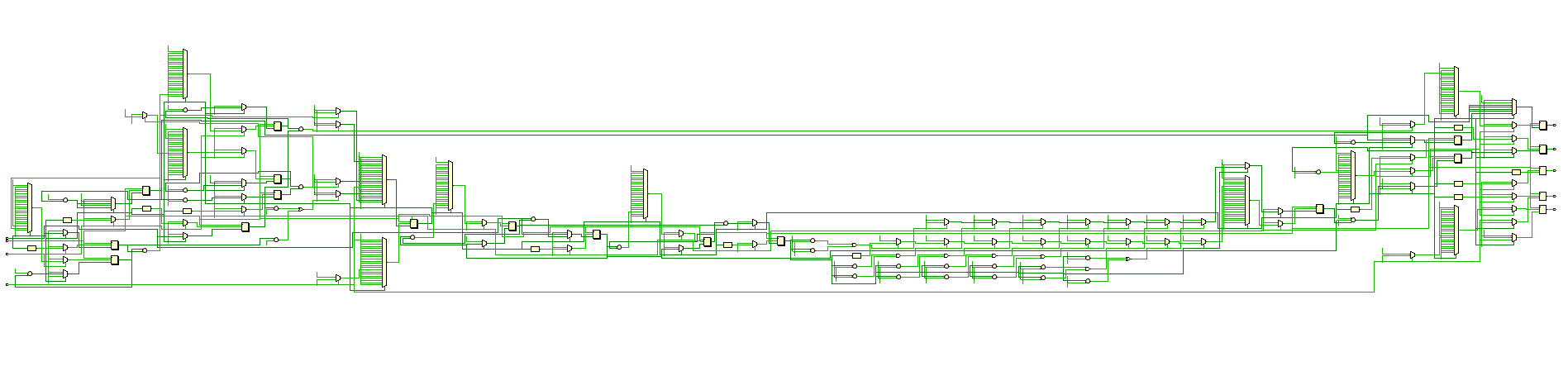
Stress Testing

Per mettere alla prova il circuito sintetizzato abbiamo sfruttato un generatore di file di test, messo a disposizione degli studenti del corso, per creare test benches molto più pesanti rispetto agli altri precedentemente descritti. Ogni test bench usato è stato simulato in *simulation – behavioral* e in *post synthesis – functional*. Di seguito illustriamo i casi più significativi.

* **Simulazione di 50 immagini consecutive**: abbiamo generato un file che istanzia la memoria con 50 diverse immagini. Ogni matrice ha una dimensione e dei valori generati in modo casuale. Ogni matrice è rappresentata da valori che variano da 0 a 255, e ogni valore è generato casualmente dal generatore. Il generatore stesso ha creato le istruzioni per il controllo dei valori scritti in memoria dal circuito sintetizzato. Il risultato finale è stato “test passato”, a indicare la totale assenza di errori, anche in post-synthesis.
* **Simulazione di 3 immagini consecutive con segnale di reset intermedio**: un testbench che istanzia 3 diverse matrici in memoria, i cui valori numerici sono generati in modo casuale. Inoltre per complicare la simulazione, il segnale di reset viene portato alto durante il processamento della prima matrice. La risposta del circuito è stata l’azzeramento e il ripristino dei valori dei segnali ai valori di default iniziali. Dopodichè l’esecuzione normale è ripartita con l’immagine successiva in modo corretto.

Report di Sintesi del Circuito

I risultati della sintesi del circuito sono stati calcolati a partire dal processo di *report utilization*. Il circuito è correttamente sintetizzabile e implementabile da Vivado. A seguito del processo di sintesi risultano utilizzati in totale 195 LUT (LookUp Tables) e 197 FF (Flip-Flops). I numeri risultato molto più bassi rispetto alle reali capacità della FPGA utilizzata, infatti il numero di FF è circa lo 0.07% del totale, mentre il numero di LUT è circa lo 0.14% del totale presenti sulla scheda.



Schematica del circuito sintetizzato

Conclusioni del Progetto

La progettazione del circuito attraverso una singola macchina a stati finiti (FSM) ci ha permesso di imparare ad usare il software Vivado e il linguaggio di programmazione hardware VHDL. E’ stato interessante progettare il circuito e verificare l’evoluzione dei segnali digitali nel tempo, come forma di debugging dei vari stati e il controllo della correttezza del circuito. L’algoritmo da sviluppare si è rivelato tanto impegnativo quanto stimolante per l’apprendimento del nuovo ambiente di sviluppo.