Prova Finale di Reti Logiche 2020/2021

Andrea Cerasani

13 Agosto 2021

Matricola:	XXXXXX
Codice Persona:	10680486
Docente:	William Fornaciari

1 Introduzione

Il progetto consiste nell'implementazione di una versione semplificata dell'algoritmo di equalizzazione dell'istogramma di un'immagine. Il modulo effettua l'equalizzazione di immagini in scala di grigi a 256 livelli e di dimensione massima 128x128 pixel, aumentandone il contrasto, come mostrato in figura 1.

(a) Prima (b) Dopo

Figura 1: Equalizzazione ottenuta con il modulo sviluppato

Ciascun pixel dell'immagine dev'essere trasformato nel modo seguente:

```
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)
```

Dove `MAX_PIXEL_VALUE` e `MIN_PIXEL_VALUE` sono il massimo e il minimo valore dei pixel dell'immagine, `CURRENT_PIXEL_VALUE` è il valore del pixel da trasformare e `NEW_PIXEL_VALUE` è il pixel trasformato.

Il modulo legge l'immagine da elaborare da una memoria in cui è memorizzata sequenzialmente e riga per riga. I primi due byte della memoria contengono la dimensione dell'immagine, ogni byte successivo contiene il valore dei pixel dell'immagine da elaborare. I valori dell'immagine equalizzata vengono scritti dal modulo a partire dall'indirizzo immediatamente successivo all'immagine originale.

Il modulo inizia l'elaborazione quando il segnale `i_start = 1`, il quale rimarrà alto finché `o_done` non è alto. `o_done` viene messo ad 1 quando il modulo termina la computazione e resta alto fino a che `i_start` non torna a 0. Quando accade il modulo torna allo stato iniziale pronto per una nuova codifica.

L'implementazione è stata sintetizzata con FPGA target la FPGA xc7a200tfbg484-1.

1.1 Esempio

In figura 2 è mostrato uno schema della memoria dopo aver equalizzato un'immagine di dimensione 3x2.

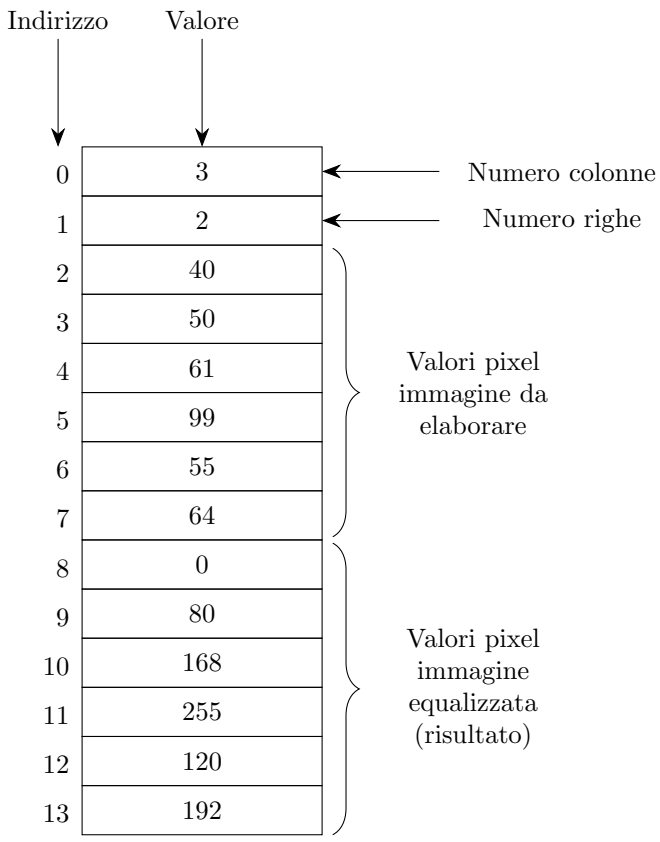

Figura 2: Schema della memoria

2 Architettura

Per lo sviluppo si è scelto un approccio bottom-up e modulare, separando il datapath dalla macchina a stati finiti e scomponendo ulteriormente il datapath in sottocomponenti, ciascuno con una specifica funzione. Ciò ha consentito di testare le singole sottocomponenti durante lo sviluppo prima di integrarle e testare il componente completo.

2.1 Descrizione ad alto livello

Nel modulo sviluppato la computazione è divisa in diverse fasi:

1. Ricerca e salvataggio valore massimo e valore minimo tra i pixel dell'immagine da elaborare
2. Salvataggio dell'indirizzo di memoria a partire dal quale verrà scritta l'immagine equalizzata
3. Per ciascun pixel:
   (a) Calcolo del nuovo valore
   (b) Scrittura in memoria del nuovo valore
4. Reset e termine della computazione

2.2 Datapath

Il datapath è il componente del modulo che contiene la logica per:

- Contare i pixel
- Gestire gli indirizzi di memoria su cui leggere o scrivere

- Confrontare valori per la ricerca del valore massimo e minimo
- Calcolare i nuovi valori da scrivere

Esso è stato suddiviso in diversi sottocomponenti per svolgere ciascuna delle suddette funzioni.

2.2.1 Min Max Comparator

Figura 3: Schema del componente

Questo componente si occupa di comparare i valori in ingresso con due registri che memorizzano il minimo e il massimo attuale per aggiornarli in caso di nuovo massimo o minimo. Sono stati utilizzati due process per i registri che contengono il minimo e il massimo e una parte combinatoria per calcolare lo stato successivo.

2.2.2 Image Down Counter

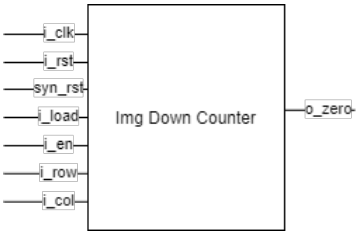

Figura 4: Schema del componente

L'Image Down Counter è usato per sapere quando sono stati processati tutti i pixel dell'immagine.
Il componente memorizza le righe e le colonne dell'immagine in due registri realizzati con process, dopo aver memorizzato righe e colonne quando il segnale i_en è alto decrementa il registro delle righe da i_row a 0 per i_col volte. Ogni volta che il registro delle righe viene resettato a i_row si decrementa il registro delle colonne. In questo modo, assumendo che i_en sia alto, il componente mette o_zero a 1 dopo i_row * i_col cicli di clock, quando anche i_col arriva a 0.

2.2.3 Address Increaser

Figura 5: Schema del componente

L'Address Increaser è usato per incrementare l'indirizzo della memoria e passare al pixel successivo dell'immagine.
Il componente contiene un registro che memorizza un valore di 16 bit, nel nostro caso l'indirizzo di memoria. Esso viene messo a zero di default dopo un reset ma settando i_load a 1 viene memorizzato il valore di i_data. Questa funzione viene usata per saltare i due indirizzi in cui sono contenute righe e colonne nella seconda parte della computazione del modulo, che verrà approfondita nel paragrafo

riguardante la FSM.
Con l'indirizzo desiderato nel registro interno del componente, settando i_en a 1, ad ogni fronte di salita del clock il componente aumenta di 1 l'indirizzo e porta il nuovo indirizzo in uscita su o_address.

2.2.4 New Value Logic

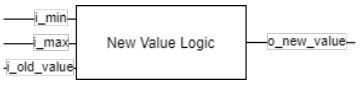

Figura 6: Schema del componente

Il New Value Logic è un circuito combinatorio usato per calcolare il valore di un nuovo pixel dell'immagine equalizzata, dati in ingresso tutti i valori necessari, secondo la formula fornita dalla specifica. Riceve in ingresso il valore minimo e massimo tra i pixel e il valore dell'immagine da equalizzare.
Inoltre per calcolare `FLOOR(LOG2(DELTA_VALUE + 1))`, necessario per computare il nuovo valore, si è scelto di usare un controllo a soglia.

Oltre a questi componenti, nel datapath sono presenti due registri, implementati con process nella entity del datapath, che memorizzano il numero di righe e di colonne dell'immagine da equalizzare e che sono usati per reimpostare il valore delle righe o delle colonne dell'Image Down Counter durante la computazione.

2.2.5 Schema del datapath

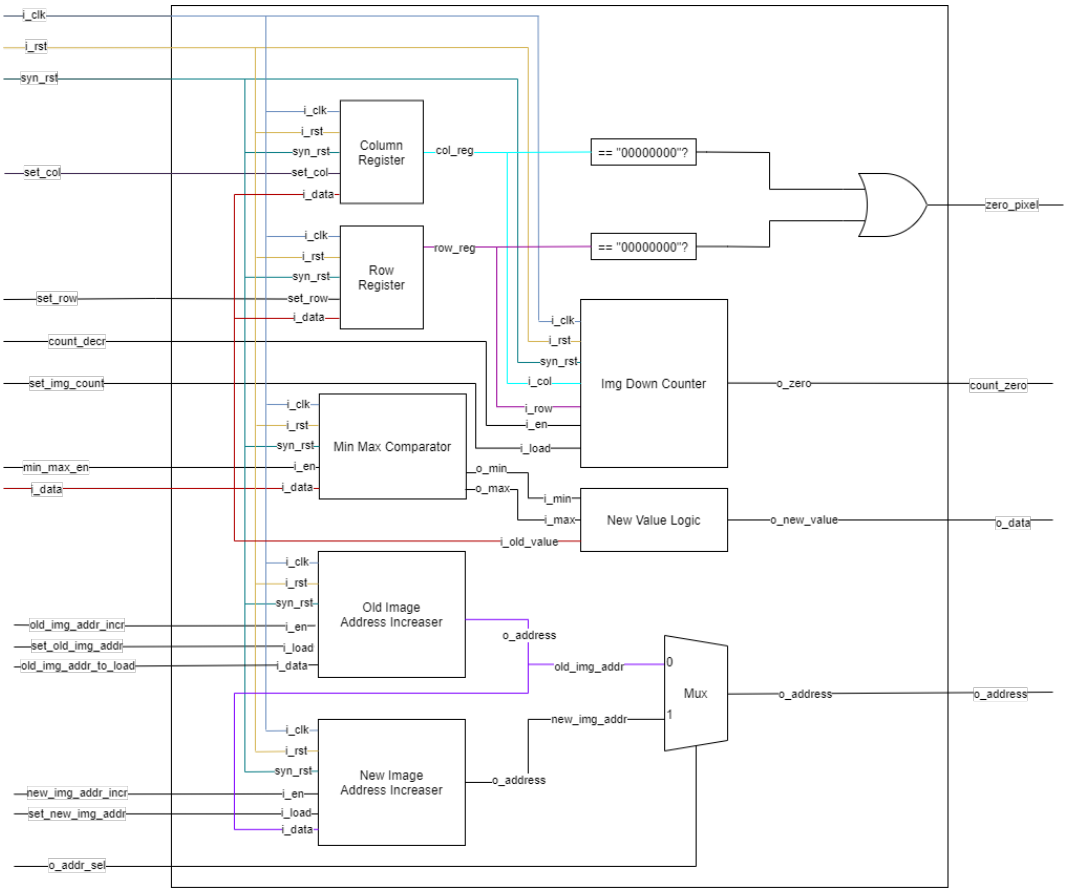

Figura 7: Schema del datapath

In figura 7 è mostrato uno schema del datapath, per una più facile lettura i segnali con più diramazioni sono stati colorati.
Si può notare l'aggiunta dei registri delle colonne e delle righe, sopra menzionati, del multiplexer per selezionare l'indirizzo di memoria dell'immagine da equalizzare o di quella equalizzata e il segnale zero_pixel, usato per ottimizzare la computazione nel caso di immagine vuota.
Si noti inoltre la presenza di un reset sincrono in tutti i componenti, che viene usato a fine computazione.

2.3 Macchina a Stati Finiti

La Macchina a Stati Finiti (FSM) è il componente che si occupa di gestire lo stato della computazione e che si interfaccia con la memoria, facendo uso dei dati prodotti dal datapath. È stata implementata con due process, dividendo:

- Parte combinatoria: si occupa di associare ad ogni stato i valori dei segnali che interagiscono con datapath e memoria
- Parte sequenziale: a seconda dello stato corrente e dei valori dei segnali della sensitivity list (cur_state, i_start, zero_pixel, count_zero) determina lo stato successivo e fa avanzare la computazione

La computazione inoltre può essere divisa concettualmente in due macrofasi che coincidono con due scansioni dell'immagine da equalizzare:

1. Ricerca massimo e minimo e calcolo dell'indirizzo di memoria dove scrivere l'immagine equalizzata
2. Calcolo valore dei pixel dell'immagine equalizzata e scrittura in memoria

Di seguito sono elencati gli stati della FSM con una breve descrizione della loro funzione:

- `START`: Stato di idle a cui si giunge dopo un reset o dopo la fine della computazione di un'immagine. Si attende che il segnale `i_start` venga messo ad 1 per passare allo stato successivo
- `WAITCOL`: Stato in cui si attende che la memoria restituisca il valore del numero di colonne
- `COL`: Stato in cui si memorizza nel registro del componente il numero di colonne
- `WAITROW`: Stato in cui si attende che la memoria restituisca il valore del numero di righe
- `ROW`: Stato in cui si memorizza nel registro del componente il numero di righe
- `WAITCOUNT`: Stato in cui si inizializza il componente Image Down Counter con i valori di righe e colonne memorizzati e da cui si va direttamente allo stato di `RESET` nel caso in cui l'immagine da elaborare sia vuota
- `MINMAX`: Stato in cui si confronta il valore del pixel in ingresso con l'attuale massimo e minimo e si aggiorna eventualmente il loro valore, si decrementa l'Image Down Counter e si aumenta l'indirizzo di memoria
- `WAITMINMAX`: Stato in cui si attende che la memoria restituisca il valore del pixel in ingresso e si verifica se l'Image Down Counter è arrivato a 0 per passare alla fase successiva della computazione
- `PHASE_2`: Stato in cui si salva l'indirizzo di memoria a partire da cui verrà memorizzata l'immagine equalizzata, si reinizializza il componente Image Down Counter e si reimposta l'indirizzo di memoria dell'immagine da equalizzare all'indirizzo del primo pixel
- `READ` : Stato in cui si legge il valore del pixel che viene usato dal componente New Value Logic del datapath per calcolare il valore del pixel corrispondente nell'immagine equalizzata, si decrementa l'Image Down Counter e si aumenta l'indirizzo di memoria salvato riferito all'immagine da equalizzare
- `WRITE`: Stato in cui si scrive il pixel calcolato in memoria e si aumenta l'indirizzo di memoria salvato riferito all'immagine equalizzata

- `RESET`: Stato in cui si giunge dopo aver calcolato e scritto tutti i valori dei pixel dell'immagine equalizzata. In questo stato si resetta con un reset sincrono il datapath
- `DONE`: Stato in cui viene settato `o_done` a 1 per segnalare la fine della computazione e in cui si attende che `i_start` sia messo a 0 per poter tornare allo stato `START`

In figura 8 è riportato uno schema della FSM.

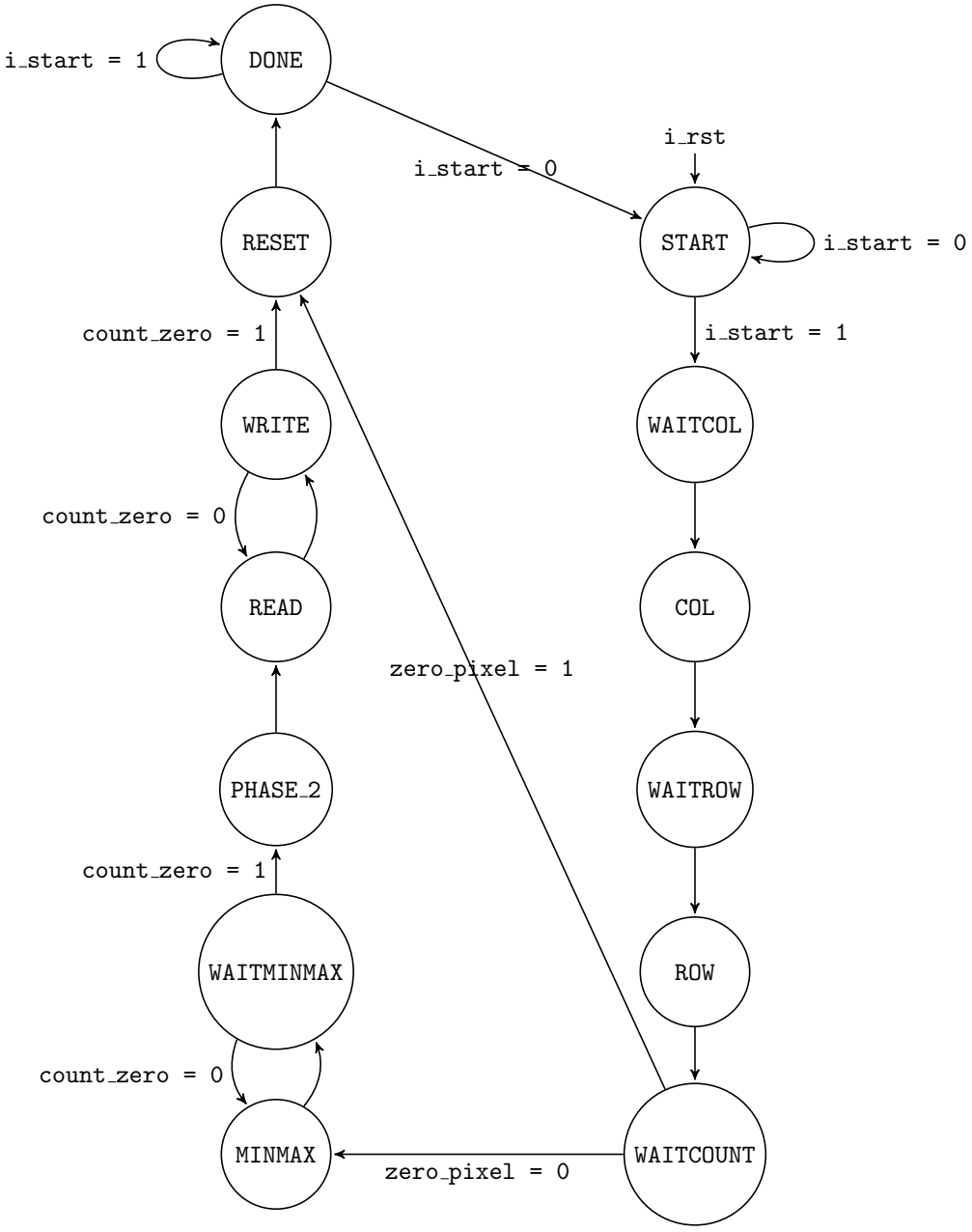

Figura 8: Rappresentazione della macchina a stati

3 Risultati Sperimentali

3.1 Report di sintesi

Dal punto di vista dell'area la sintesi riporta il seguente utilizzo dei componenti:

- LUT: 158 (0.12% del totale)
- FF: 93 (0.03% del totale)

Si è fatta particolare cautela nella scrittura del codice per evitare che venissero inferiti Latch.

Dal punto di vista del tempo, usando il comando *report_timing* e impostando il periodo di clock a 100ns tra i constraint, si è osservato uno slack di 95.570ns e dunque un ampio margine dal limite dei 100ns dettato dalla specifica.

3.2 Simulazioni

Per verificare il corretto funzionamento del modulo sono stati utilizzati diversi numerosi test durante le varie fasi di sviluppo del modulo.
Nelle fasi intermedie di sviluppo sono stati costruiti test per i singoli sottocomponenti del datapath per accertarsi del loro corretto funzionamento.
Successivamente a sviluppo completato si è fatto uso di un generatore di test casuale per verificare che il modulo fosse robusto e che equalizzasse correttamente più immagini in computazioni in successione.
Inoltre sono stati utilizzati test per verificare il corretto funzionamento del modulo nei casi limite, quali:

- Immagine con 0 pixel, quindi con il valore 0 in uno o entrambi i primi due indirizzi di memoria, per testare il caso limite dell'immagine vuota
- Immagine con valore dei pixel tutti a 0, per testare il valore minimo dei pixel e lo shift massimo
- Immagine con valore dei pixel tutti a 255, per testare il valore massimo dei pixel e lo shift massimo
- Un'immagine con `DELTA_VALUE = 127` e una con `DELTA_VALUE = 126` per testare il controllo a soglia implementato per calcolare `FLOOR(LOG2(DELTA_VALUE + 1))`
- Immagine avente come valori dei pixel solo 0 e 255, per testare lo shift minimo
- Immagine di dimensione 128x128, per testare la dimensione massima dell'immagine
- Immagine di dimensione 1x1 Per testare la dimensione minima di un'immagine non vuota

Il primo test in particolare ha suggerito l'ottimizzazione implementata per saltare gli stati intermedi e giungere a quello di `RESET`.

I test sono stati superati tutti con successo in pre e post-sintesi.

4 Conclusioni

Si ritiene, a seguito dei test effettuati, che il modulo sviluppato rispetti le specifiche. Dal punto di vista del design si è cercato di adottare un approccio modulare che ha permesso di scomporre il problema in sottoproblemi e ha facilitato sviluppo e testing.
L'architettura è stata inoltre pensata per ottimizzare temporalmente la computazione favorendo il parallelismo e limitandosi a due scansioni dell'immagine da equalizzare.