Prova Finale di Reti Logiche 2020/2021

Vincenzo Greco

May 16th, 2021

Matricola:	916368
Codice Persona:	10683567
Docente:	Salice Fabio

1 Requisiti del Progetto

Il progetto richiesto consiste nell'implementazione in VHDL dell metodo di **equalizzazione dell'istogramma di una immagine**.
Data un'immagine salvata in memoria viene chiesto al componente di:

1. Accedere ad una memoria RAM per recuperare il numero di righe e colonne di pixel di cui è composta l'immagine
2. Iterare nella memoria per trovare minimo e massimo dei valori dei pixel
3. Applicare l'equalizzazione ad ogni pixel dell'immagine
4. Salvare in memoria ogni pixel equalizzato a partire dalla posizione successiva rispetto all'ultimo pixel della matrice fornita

La massima grandezza della matrice dell' immagine è $128 * 128$.
Inoltre, l'implementazione deve essere in grado di gestire un segnale di Reset.
L'implementazione deve essere sintetizzata con target FPGA xc7a200tfbg484-1.

2 Implementazione

2.1 Descrizione ad Alto Livello

Da un'ottica di alto livello, l'implementazione esegue i seguenti passi:

1. Carica il numero di righe e colonne della matrice dei pixel dalla RAM
2. Esegue la ricerca di massimi e minimi:
3. Calcola il valore dello shift.
4. Per ogni pixel in memoria calcola il valore del pixel equalizzato e lo salva in RAM
5. Ad operazione conclusa il componente comunica il suo stato di done fin quando non verrà indicato che la prossima immagine è disponibile all' equalizzazione

2.2 Interpretazioni personali della specifica

- Nel caso fosse letto un valore maggiore di righe e/o colonne rispetto a quello consentito, questo verrà salvato ugualmente come il massimo consentito dalla specifica (128). Nel caso di presenza di valori in RAM oltre l' indice "righe x colonne + 2", questi verranno totalmente ignorati e, nel caso, sovrascritti.
- Per l'implementazione si è scelto di supporre il Reset asincrono rispetto al segnale di clock.

2.3 Macchina a Stati Finiti

La FSM è stata realizzata con specifica behavioural mediante due processi: `STATE` e `DELTA_LAMBDA`:

- Il processo `STATE` è il processo che si occupa di cambiare lo stato sul fronte di salita del clock e di portare la macchina nello stato di reset se il segnale `I_RST` fosse rilevato.
- Il processo `DELTA_LAMBDA` è invece il processo combinatorio che calcola le uscite e lo stato validi per il prossimo fronte di salita del clock.
- L'implementazione dello shift è stata fatta tramite controllo di soglia in modo da poter risparmiare sulla complessità delle operazioni effettuate avendo anche un controllo maggiore sul risultato.

Gli stati della FSM sono:

- `RST`: Stato in cui la macchina si prepara che il segnale `i_start` venga portato alto per poter iniziare l'esecuzione. Questo stato si occupa anche di portare i segnali interni in uno stato consistente per l'esecuzione. In particolare il massimo e il minimo verranno inizializzati rispettivamente al valore minimo e massimo così da poter essere direttamente confrontati con il valore del primo pixel letto da RAM.

- `SAVE_COLUMN`: Stato in cui si legge dalla RAM l'indirizzo 0 della memoria che corrisponde al numero di colonne che compongono l'immagine. Il valore salvato sarà il minimo fra quello fronito da `i_data` e 128. Nel caso il valore fosse 0 il processo si porterà nello stato di `DONE`.

- `SAVE_ROW`: Stato in cui si legge dalla RAM l'indirizzo 1 della memoria che corrisponde al numero di righe che compongono l'immagine. Il valore salvato sarà il minimo fra quello fronito da `i_data` e 128. Nel caso il valore fosse 0 il processo si porterà nello stato di `DONE`.

- `SAVE_PIXEL`: Stato in cui si legge il valore del pixel puntato sulla RAM. Questo valore verrà confrontato con massimo e minimo attuali per, nel caso, aggiornarli.

- `LOOP_SAVE`: Stato in cui decide se siano stati letti tutti i valori della matrice di pixel dalla RAM.

  - Se non si è porata a termine la scansione di tutti pixel il processo tornerà nello stato `SAVE_PIXEL`.
  - Se tutti i pixel sono stati scansionati verrà calcolato il delta tramite l'operazione $\texttt{DELTA} = max - min$.

- `CALC_SHIFT`: Stato in cui si calcola $shift = 8 - \lfloor (LOG_2(\texttt{DELTA} + 1)) \rfloor$ tramite controllo di soglia.

- `EQUALIZE_PIXEL`: Stato in cui si calcola il valore del pixel equalizzato come $valore\_pixel\_equalizzato = (valore\_pixel\_corrente - min) \ll shift$

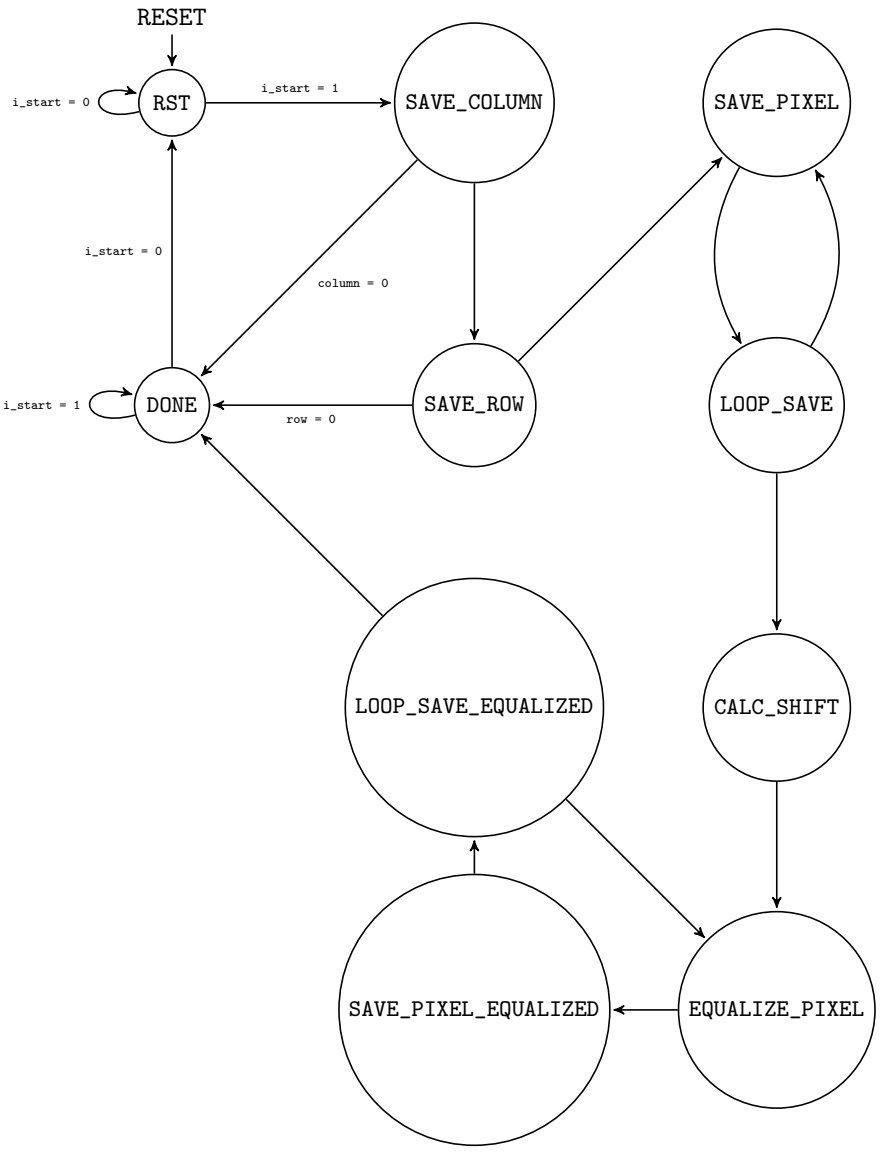
- `SAVE_PIXEL_EQUALIZED`: Stato in cui viene salvato in memoria il valore $nuovo\_valore = \min(255, valore\_pixel\_equalizzato)$ partendo dall' indirizzo $righe * colonne + 2$

- `LOOP_SAVE_EQUALIZED`: Stato in cui si decide se siano stati equalizzati tutti i pixel della matrice

  - Se non si è porata a termine l'equalizzazione di tutti i pixel il processo tornerà nello stato `EQUALIZE_PIXEL`.
  - Se tutti i pixel sono stati equalizzati verrà il processo si porterà nello stato di `DONE`.

- `DONE`: Stato in il segnale `O_DONE` verrà portato alto fin quando il segnale `i_start` non venga portato basso per tornare poi allo stato di `RST`.

Nella figura a seguito è riportato il disegno dell'automa.



3 Test Benches

I test sono stati effettutati allo scopo di trovari eventuali criticità del componente

- Segnale di `RESET` asincrono rispetto al clock
- Valore di righe e/o colonne a 0 controllando che quindi nessun valore in RAM venga modificato
- Matrice di pixel 1 x 1 e 128 x 128 per testare le dimensioni limite dell' immagine
- Altri test bench vari generati casualmente, mantenendo comunque la validità della specifica
- Testata la totale assenza di modifiche in memoria oltre quelle necessarie al implementazione della specifica richiesta

Per tutti i test svolti è stata effettuata la simulazione behavioural e successivamente la simulazione functional e timing post-synthesis, tutte con successo.

4 Risultati Sperimentali

4.1 Report di Sintesi

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

- LUT: 173 (0.13% del totale)
- FF: 112 (0.04% del totale)

4.2 Risultato dei Test Bench

Lavorando sul testing si è anche provato a variare il periodo di clock, confermando che, non solo l'implementazione mantiene il suo funzionamento anche a 1 ns nella simulazione Behavioural e 2 ns nella simulazione Post-Synthesis, ma anche che la macchina non presenti risultati inattesi o inconsistenti. Test inferiori a 1 ns non sono stati effettuati.

5 Conclusioni

Avendo passato la totalità dei test sia scritti manualmente, che generati, che ricevuti attraverso beep o altri canali di comunicazione, posso affermare con una certa confidenza che l'implementazione rispetti tutte le specifiche indicate.