Prova Finale di Reti Logiche

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 e' composta l'immagine
2. Iterare nella memoria per trovare minimo e massimo dei valori dei pixel
3. Trovare il livello di shift da applicare ad ogni pixel dell'immagine
4. Calcolare il nuovo valore dei pixel e salvarli in memoria

La massima grandezza della matrice dell'immagine e' 128 x 128 come da specifica. Nel caso fosse letto un valore maggiore di righe o colonne questo verra' salvato come 128. Nel caso di presenza di valori oltre l' indice di righe x colonne, questi verranno, nel caso fossero raggiunti, sovrascritti dai pixel pixel equalizzati.
Inoltre, l'implementazione deve essere in grado di gestire un segnale di Reset. Per l'implementazione si è scelto di supporre il Reset asincrono rispetto al segnale di clock. L'implementazione deve essere poi sintetizzata con target a 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 verra' indicato che la prossima immagine e' disponibile all' equalizzazione

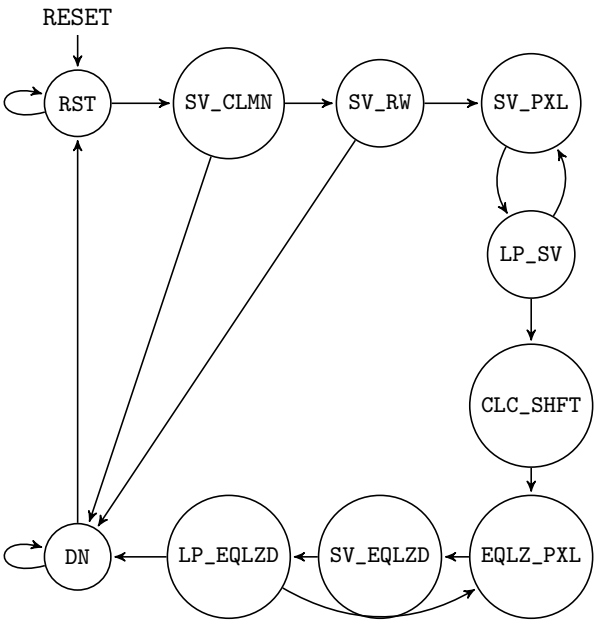
2.2 Macchina a Stati Finiti

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 questo 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.
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. Queseto stato si occupa anche di portare i segnali interni in uno stato consistente per l'esecuzione.
- `SAVE_COLUMN`: Stato in cui si legge dalla RAM l'indirizzo 1 della memoria che corrisponde al numero di colonne che compongono l'immagine. Il valore salvato sara' il minimo fra quello fronito da `i_data` e 128. Nel caso il valore fosse 0 il processo porta l'`o_done` alto.
- `SAVE_ROW`: Stato in cui si legge dalla RAM l'indirizzo 0 della memoria che corrisponde al numero di righe che compongono l'immagine. Il valore salvato sara' il minimo fra quello fronito da `i_data` e 128. Nel caso il valore fosse 0 il processo porta l'`o_done` alto.
- `SAVE_PIXEL`: Stato in cui si legge il valore del pixel e li confrontano con massimi e minimi.
- `LOOP_SAVE`: Stato in cui decide se siano stati letti tutti i valori della matrice di pixel
- `CALC_SHIFT` : Stato in cui si calcola lo shift

- `EQUALIZE_PIXEL`: Stato in cui si calcola il valore aggiornato di un pixel al seguito della equalizzazione
- `SAVE_PIXEL_EQUALIZED`: Stato in cui viene salvato il pixel in RAM
- `LOOP_SAVE_EQUALIZED` : Stato in cui si decide se siano stati equalizzati tutti i pixel della matrice
- `DONE`: Stato in cui si aspetta che `i_start` venga portato basso e riportandosi 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 criticita' 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 la validità della specifica

Per tutti i test riportati e i test successivi 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 Functional Post-Synthesis. Test inferiori a 1ns non sono stati effettuati.

5 Conclusioni

Avendo passato la totalita' 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.