Report Progetto di Reti Logiche 2021

Chiara Bianchimani

April 2021

Sommario

Descrizione e sintesi in VHDL di un componente che implementa il metodo di equalizzazione dell'istogramma di un immagine in scala di grigi a 256 livelli. La realizzazione è stata effettuata tramite una FSM di Mealy. Il componente viene testato in pre-sintesi e in post-sintesi.

1 Introduzione

La specifica del modulo richiede che nella RAM la dimensione dell'immagine venga specificata con indirizzamento al Byte dal byte 0. I primi due byte corrispondono alla grandezza dell'immagine, in righe e colonne. Dalla posizione 2 vengono specificati i valori dei pixel dell'immagine, ciascuno di 8 bit.I valori dell'immagine equalizzata sono specificati dall'indirizzo immediatamente successivo all'immagine di partenza.

La dimensione massima dell'immagine supportata dal modulo è di 128x128 pixel.

Il modulo inizia la codifica dell'immagine quando il segnale di `i_reset` viene portato a 1 e incomincia a ricodificare l'immagine quando `i_start` viene portato alto. Il modulo termina l'equalizzazione dell'immagine quando il segnale `o_done` viene portato a 1. Il modulo può equalizzare più di un'immagine, per far questo basta che `o_done` venga riportato a 0 e `i_start` torni ad 1.

Lo strumento utilizzato è Xilinx Vivado e la FPGA è *xc7a200tfbg484-1*.

L'algoritmo utilizzato per l'equalizzazione è il seguente:

```
delta_value = max_pixel_value - min_pixel_value
shift_level = (8-floor(log2(delta_value + 1)
temp_level = (current_pixel_value - min_pixel_value) << shift_level
new_pixel_value = min(255, temp_pixel)
```

Questi passaggi vengono effettuati dalla FSM come descritto in figura 1.

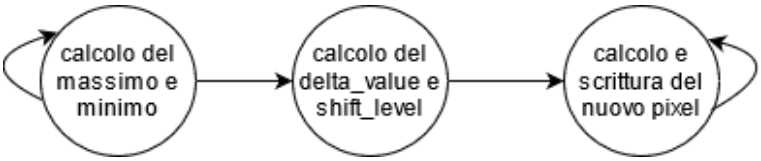

Figura 1: Il calcolo del massimo e minimo, come il calcolo e scrittura dei nuovi pixel, viene effettuati in cicli che leggono in memoria un pixel alla volta dell'immagine di partenza.

Un esempio di funzionamento del modulo di un'equalizzazione di un'immagine 4x3 si può vedere in figura 2.

Indirizzo della Ram	Contenuto	Commento	Indrizzo della RAM	Contenuto	Commento
0	4	numero di colonne	13	120	ultimo pixel dell'immagine
1	3	numero di righe	14	0	primo pixel dell'immagine equalizzata
2	0	primo pixel dell'immagine	15	40	
3	10		16	80	
4	20		17	120	
5	30		18	160	
6	40		19	200	
7	50		20	240	
8	60		21	255	
9	70		22	255	
10	80		23	255	
11	90		24	255	
12	100		25	255	ultimo pixel dell'immagine equalizzata

Figura 2: Esempio della memoria dopo il funzionamento del modulo

Figura 3: L'effetto dell'equalizzazione dell'immagine in figura 2. A sinistra, l'immagine prima di passare nel modulo e a destra quella equalizzata. Le immagini sono state create seguendo la convenzione che il pixel in alto a sinistra sia il primo bit dell'immagine e colorando i restanti proseguendo verso destra.

2 Architettura

2.1 Moduli

L'unità è composta da un unico modulo a sua volta composto da due processi. Un processo (`state_clock`) si occupa di gestire il reset e l'aggiornamento dei flip-flop al rising edge del clock. Il secondo processo (`state_comb`) si occupa della parte combinatoria e il funzionamento della macchina a stati. Sono stati impiegati i seguenti registri:

- `state` tiene traccia dello stato corrente della FMS;
- `done` tiene traccia di quando l'operazione di equalizzazione è terminata;
- `row` (di dimensione 8 bit) tiene traccia del numero di righe dell'immagine
- `col` (di dimensione 8 bit) tiene traccia del numero di colonne dell'immagine
- `currentaddress` (di dimensione 16 bit) tiene traccia dell' address corrente di lettura della RAM;
- `totpixel` (di dimensione 15 bit) numero totale di pixel presenti nell'immagine;
- `max` (di dimensione 8 bit) il valore in scala di grigi più grande nell'immagine di partenza;
- `min` (di dimensione 8 bit) il valore in scala di grigi più piccolo nell'immagine di partenza;
- `shiftlevel` (di dimensione 8 bit) risultato dell'operazione di shift richiesta dall'algoritmo di equalizzazione;
- `temppix` (di dimensione 16 bit)il valore del nuovo pixel equalizzato;

2.2 Macchina a stati finiti

La macchina a stati finiti è stata implementata con dodici stati: IDLE, READ_COL, READ_ROW, NUM_PIX, CALC_TOT, READ_MAX_MIN, CALC_MAX_MIN, INIT, READ_PIX, CALC_PIX, WRITE_PIX, DONE.

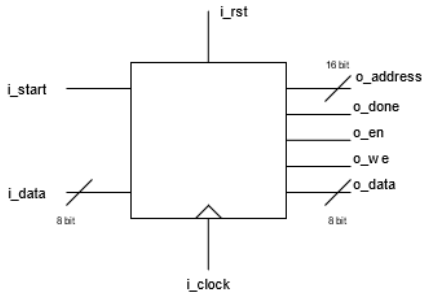

Figura 4: Entity del modulo

2.2.1 Diagramma

Si veda la figura 5 per uno schema della FSM.

IDLE Lo stato *IDLE* è lo stato iniziale della FSM ed è lo stato in cui la macchina viene portata in caso di reset. I registri vengono portati al valore di default e l'elaborazione dell'immagine incomincia quando `i_start` viene portato alto. Se c'è già stata almeno un'elaborazione di un'immagine allora lo stato aspetta che `o_done` venga portato a basso. La macchina rimane in questo stato finchè i segnali non hanno i comportamenti descritti sopra.

READ_COL Imposta le uscite (`o_en`, `o_address`) in modo da poter leggere dalla memoria il numero di colonne dell'immagine.

READ_ROW Viene letto il primo indirizzo in memoria, corrispondente al numero di colonne dell'immagine, e salvato nel registro `col`. Nel caso in cui il numero di colonne sia nullo, la macchina termina nello stato di *DONE*. Vengono poi impostate le uscite (`o_en`, `o_address`) per leggere il numero di righe.

NUM_PIX Viene salvato il numero di righe nel registro `row`. Nel caso in cui il numero di righe è nullo, la macchina va nello stato di *DONE*.

CALC_TOT Viene utilizzato un ciclo con la seguente condizione: `col = 0` then `next_state = CALC_TOT` per calcolare il numero totale di pixel nell'immagine. Il valore viene salvato nel registro `totpixel`.

READ_MAX_MIN Inizia l'operazione di lettura dei pixel per il calcolo del massimo e minimo. Si impostano le uscite `o_en` `o_address` per poter leggere un pixel alla volta.

CALC_MAX_MIN Si accede al valore del pixel, che viene confrontato con i valori precedenti dei registri `max` e `min`. Nel caso `min = 0` e `max = 255` il calcolo del massimo e del minimo si interrompe per andare nello stato successivo *INIT*. Altrimenti il ciclo $READ\_MAX\_MIN \Longleftrightarrow CALC\_MAX\_MIN$ continua fino alla fine dello scorrimento dell'immagine.

INIT In questo stato di inizializzazione vengono calcolati il delta tra il massimo e il minimo (`deltavalue`, una variabile) e lo `shiftlevel`. Il calcolo dello `shiftlevel` viene implementato con un case statement che tiene conto dei possibili risultati dell'operazione di log_2 con un `deltavalue` minimo di 1 e massimo di 256. Il registro `currentaddress` viene riportato all'inizio dell'immagine, cioè all'indirizzo 2.

READ_PIX Analogamente a quanto fatto nello stato *READ_MAX_MIN* vengono aggiornate le uscite `o_en`, `o_address` per la lettura dei pixel per svolgere le operazioni di equalizzazione.

CALC_PIX Viene letto il pixel corrente e tramite l'operazione di shift viene calcolato un valore temporaneo del nuovo pixel, `temppix`. Vengono impostate l'uscita `o_address = currentaddress + totpixel` per scrivere il pixel nella posizione corretta della memoria.

WRITE_PIX Vengono impostate le uscite `o_we`, `o_address` per scrivere in memoria il nuovo pixel, `temppix`. Se questo valore è maggiore di 255 viene impostato a valore massimo di 255. Se l'operazione di equalizzazione è finita (`currentaddress = currentaddress + totpixel +1`) allora la macchina ha finito l'elaborazione e va nello stato di *DONE*. Altrimenti il ciclo $READ\_PIX \longrightarrow CALC\_PIX \longrightarrow WRITE\_PIX$ continua fino a che l'equalizzazione non ha terminato.

DONE Il segnale `o_done` viene portato ad alto e la macchina va nello stato di *IDLE* in attesa di nuove immagini.

3 Risultati Sperimentali

La rete è stata sintetizzata con un periodo di clock di 100ns, ma è stata testata con successo anche utilizzando un periodo di clock di 15ns. Post-sintesi sono stati utilizzati 88 flip-flops (FF), 209 look-up tables (LUT) e 38 pin tra input e output. [1]

[1] In una versione alternativa del modulo, dove l'operazione di moltiplicazione per il calcolo totale del numero di pixel viene effettuata utilizzando l'operatore VHDL * i valori riscontrati sono di 72 FF, 157 LUT e 38 IO

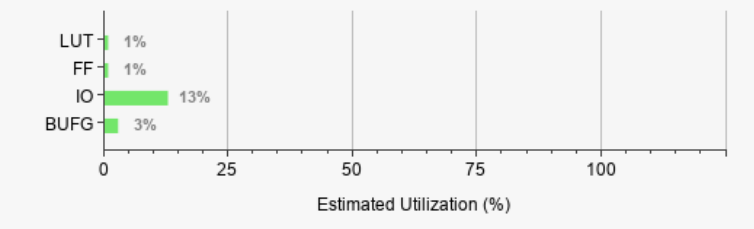

Figura 5: Risultati di sintesi

4 Simulazioni

Dai test effettuati il modulo funziona correttamente sia in pre-sintesi che in post-sintesi. I dati dei test si riferiscono ad un clock di 100ns.

4.1 Testbench base

Il primo caso di test effettuato è quello che è stato fornito, cioè un'immagine di dimensioni 2x2. Il tempo di esecuzione della simulazione Behavioral è di 3650ns, mentre in Post-Synthesis 3650,1 ns.

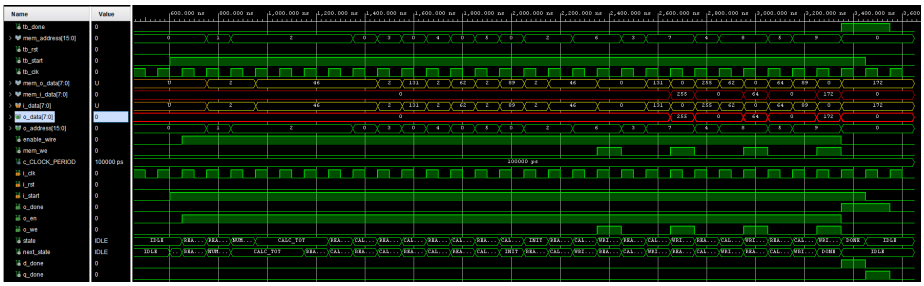

Figura 6: I valori dell'immagine di partenza sono 46,131,62,89 mentre quelli dell'immagine equalizzata sono 0,255,64,172.

4.2 Equalizzazione di un'immagine da 128x128

Viene fornita al modulo una sola immagine di dimensioni massime 128x128. Il tempo di esecuzione della simulazione Behavioral è di 752117.5ns, mentre in Post-Synthesis 752117.6ns.

4.3 Equalizzazione di più immagini consecutive

Vengono fornite al modulo in successione sedici immagini di dimensioni 2x3. Il tempo di esecuzione della simulazione Behavioral è di 67450 ns, mentre in Post-Synthesis 69050.1ns.

4.4 Numero di colonne e/o righe è pari a zero o maggiore di 128

Il modulo è stato testato per il corretto funzionamento nel caso in cui un'immagine abbia un numero di righe o colonne non valido.

4.5 Altri test

Sono stati effettuati anche i seguenti test:

- test con `min = max`
- l'immagine ha un solo pixel
- test con valori dei pixel limite (1 e 255)
- test con numero variabile di immagini e loro grandezza
- test dove viene fornito il segnale di `i_reset` durante l'esecuzione

5 Conclusioni

Il modulo funziona correttamente nel rispetto delle specifiche date. E' stata posta più attenzione alla leggibilità del codice del modulo piuttosto che ad un'ottimizzazione del numero degli stati, per favorire la comprensione della macchina a stati ed evitare possibili errori.

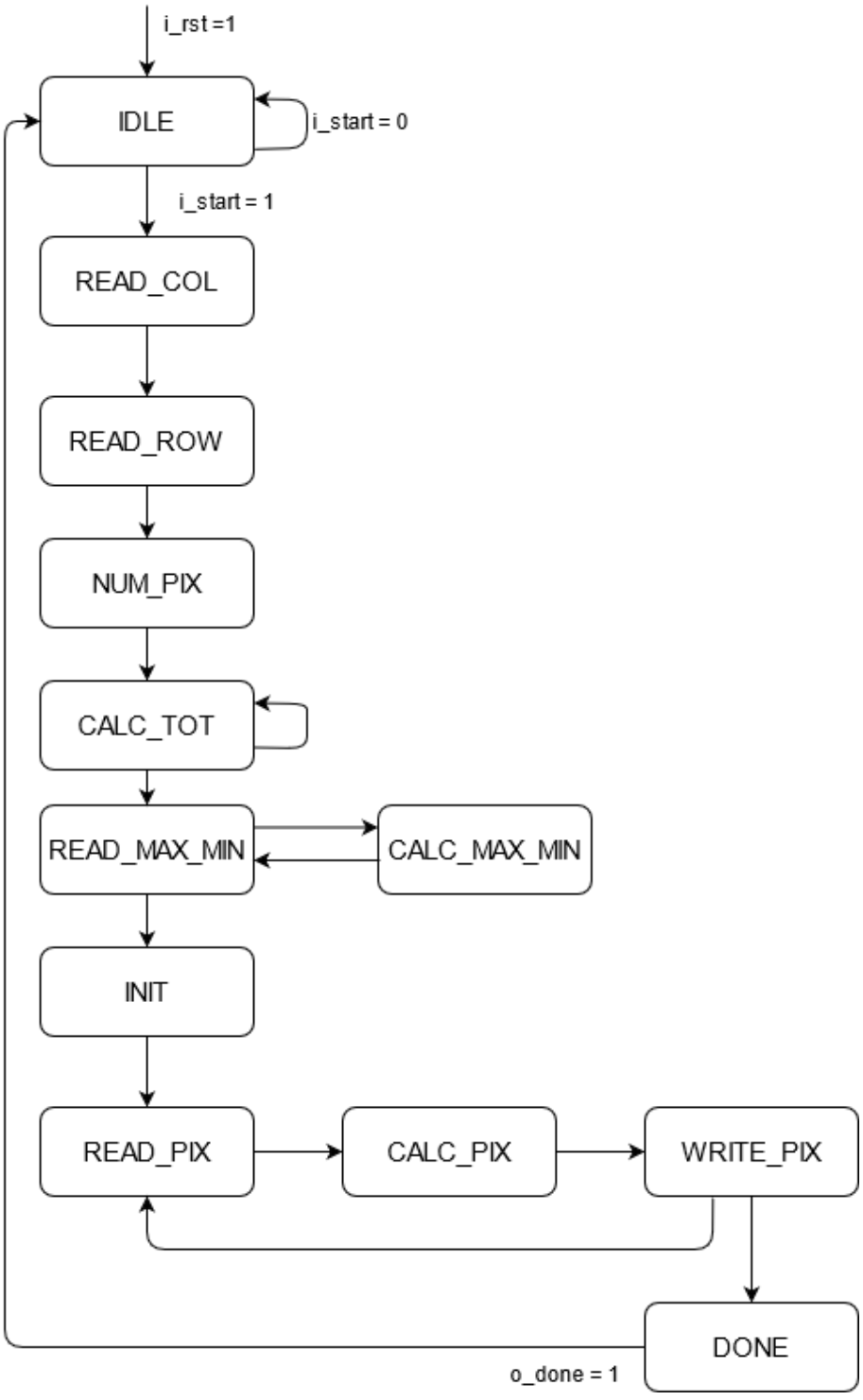

Figura 7: Schema della FSM