# Prova finale – Progetto di reti logiche

Giovanni De Novellis

# 1. Introduzione

## 1.1 Scopo del progetto

Lo scopo del progetto è di implementare un componente, in linguaggio VHDL, che effettui una versione semplificata dell'algoritmo per il metodo di equalizzazione dell'istogramma di immagini. Il metodo di equalizzazione dell'istogramma serve a ricalibrare il contrasto di un'immagine, distribuendo i valori di intensità dei suoi pixel su tutto l'intervallo possibile. In particolare, l'algoritmo sviluppato seleziona il valore massimo e il valore minimo tra i pixel dell'immagine, calcola la loro differenza e shifta ogni pixel del logaritmo della differenza trovata in precedenza.

## 1.2 Specifiche

I dati, ciascuno di dimensione 8 bit, sono memorizzati in una memoria con indirizzamento al byte.

Il numero di indirizzi da leggere dipende dalla dimensione dell'immagine: i primi due, 0 e 1, contengono il numero di righe e di colonne dell'immagine, ed i successivi indirizzi contengono i pixel dell'immagine, che possono avere valore massimo 255. L'ultimo indirizzo da leggere avrà perciò valore num_righe *

num_colonne + 1 e dall'indirizzo successivo andranno scritti tutti i pixel dell'immagine equalizzata.

Di seguito è riportato un semplice esempio di immagine 2x3 per chiarire le specifiche appena spiegate.

## Esempio:

| Indirizzo memoria | Valore | Commento |
|---|---|---|
| 0 | 2 | Numero colonne |
| 1 | 3 | Numero righe |
| 2 | 55 | Primo byte originale |
| 3 | 255 | |
| 4 | 22 | |
| 5 | 80 | |
| 6 | 30 | |
| 7 | 180 | Ultimo byte originale |
| 8 | 66 | Primo byte equalizzato |
| 9 | 255 | |
| 10 | 0 | |
| 11 | 116 | |
| 12 | 16 | |
| 13 | 255 | Ultimo byte equalizzato |

### 1.3 Interfaccia del componente

Il componente da descrivere ha un'interfaccia così definita:

```
entity project_reti_logiche is
```

```
Port (
i_clk : in std_logic;
i_start : in std_logic;
i_rst : in std_logic;
i_data : in std_logic_vector(7 downto 0);
o_address : out std_logic_vector(15 downto 0);
o_done : out std_logic;
o_en : out std_logic;
o_we : out std_logic;
o_data : out std_logic_vector(7 downto 0)
);
end project_reti_logiche;
```

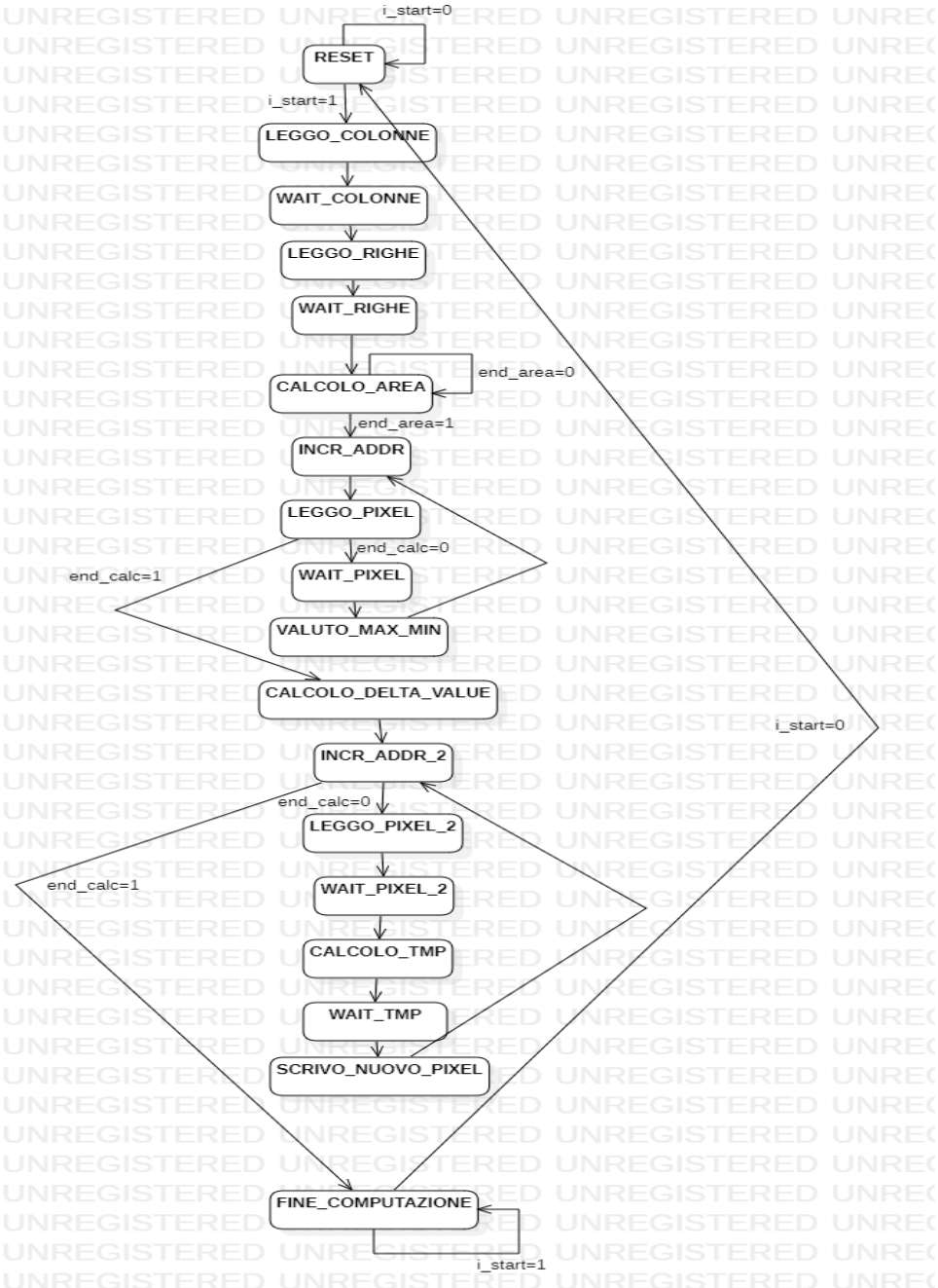
Nella quale:

- i_clk è il segnale del clock, è generato dal test bench, ed è richiesto che il progetto funzioni con un valore di almeno 100ns.
- i_start è il segnale di start, generato dal test bench, che il componente deve attendere abbia valore alto prima di poter iniziare a elaborare un'immagine.
- i_rst è il segnale di reset, usato per inizializzare il componente prima del segnale di start e per interrompere un'elaborazione e riportare il componente in attesa di una nuova immagine.
- i_data è il vettore di dimensione 1 byte che viene usato dalla memoria per propagare i dati al componente dopo una richiesta di lettura.
- o_address è il vettore di dimensione 1 byte che usa il componente per comunicare alla memoria l'indirizzo a cui accedere in lettura o scrittura.

- o_done è il segnale che il componente usa per comunicare al testbench che l'elaborazione è terminata e tutti i dati sono stati scritti in memoria.
- o_en è il segnale che il componente deve alzare per poter interagire con la memoria sia in lettura che in scrittura.
- o_we è il segnale che il componente deve alzare per poter interagire con la memoria in scrittura.
- o_data è il vettore di dimensione 1 byte che viene usato dal componente per propagare i dati alla memoria dopo una richiesta di scrittura.

# 2. Design del componente

## 2.1 Macchina a stati




### 2.1.1 RESET

Stato iniziale della macchina, in cui si attende che i_start venga alzato. In caso venga alzato il segnale i_rst, in un qualsiasi momento della computazione, si torna in questo stato.

### 2.1.2 LEGGO_COLONNE

Stato in cui viene richiesto alla memoria il valore delle colonne alzando o_en e settando o_address all'indirizzo 0 tramite un segnale di comando del datapath.

### 2.1.3 WAIT_COLONNE

Stato in cui viene caricato nel registro apposito, tramite un comando di load al datapath, il numero di colonne fornito dalla memoria.

### 2.1.4 LEGGO_RIGHE

Stato in cui viene richiesto alla memoria il valore delle righe alzando o_en e settando o_address all'indirizzo 1 tramite un segnale di comando del datapath.

### 2.1.5 WAIT_RIGHE

Stato in cui viene caricato nel registro apposito, tramite un comando di load al datapath, il numero di righe fornito dalla memoria.

### 2.1.6 CALCOLO_AREA

Stato in cui viene calcolata l'area dell'immagine da elaborare, attraverso un comando di load al datapath, utilizzando un vettore area che viene aggiornato di num_colonne ogni ciclo di clock e un contatore che viene aggiornato di 1 ogni ciclo di clock e segnala, alzando il segnale end_area, la fine del calcolo quando raggiunge il valore num_righe.

### 2.1.7 INCR_ADDR

Stato in cui viene aumentato il contatore dell'indirizzo corrente tramite un comando al datapath.

### 2.1.8 LEGGO_PIXEL

Stato in cui viene richiesto alla memoria il valore del pixel corrente alzando o_en e settando o_address al valore del contatore dell'indirizzo tramite un segnale di comando del datapath.

### 2.1.9 WAIT_PIXEL

Stato in cui viene caricato nel registro apposito, tramite un comando di load al datapath, il valore del pixel corrente fornito dalla memoria.

### 2.1.10 VALUTO_MAX_MIN

Stato in cui, tramite un segnale di load al datapath, viene valutato il pixel corrente ed eventualmente aggiornato il valore del pixel massimo o del pixel minimo.

### 2.1.11 CALCOLO_DELTA_VALUE

Stato in cui, dopo aver valutato tutti i pixel dell'immagine e trovato il valore massimo e minimo, viene calcolato e salvato nel datapath il loro delta value. Dopo aver calcolato il delta value il datapath, attraverso altri due process, calcola autonomamente anche floor e shift level.

### 2.1.12 INCR_ADDR_2

Stato in cui viene aumentato il contatore dell'indirizzo corrente tramite un comando al datapath.

### 2.1.13 LEGGO_PIXEL_2

Stato in cui viene richiesto alla memoria il valore del pixel corrente alzando o_en e settando o_address al valore del contatore dell'indirizzo tramite un segnale di comando del datapath.

### 2.1.14 WAIT_PIXEL_2

Stato in cui viene caricato nel registro apposito, tramite un comando di load al datapath, il valore del pixel corrente fornito dalla memoria.

### 2.1.15 CALCOLO_TMP

Stato in cui, a partire dal current_pixel, del min pixel e dello shift level, con un comando al datapath viene calcolato e salvato nel registro apposito il tmp_pixel.

### 2.1.16 WAIT_TMP

Stato in cui abilito la scrittura del nuovo pixel, che avrà valore min (255, tmp_pixel), nel registro apposito.

### 2.1.17 SCRIVO_NUOVO_PIXEL

Stato in cui abilito la scrittura del nuovo pixel nel corrispondente indirizzo in memoria, alzando o_en, o_we e selezionando il giusto indirizzo di scrittura attraverso il segnale o_addr_sel per il datapath.

### 2.1.18 FINE_COMPUTAZIONE

Stato in cui arriva il componente dopo che, avendo scritto l'ultimo pixel dell'immagine equalizzata in memoria, il datapath alza il segnale end_calc. Il componente segnala al testbench che la computazione è terminata alzando il segnale o_done e resta in questo stato fino a quando il testbench non abbassa il segnale i_start, per poi tornare allo stato di reset in attesa di una nuova immagine.

## 2.2 Data path

Il datapath è composto dai seguenti moduli:

### 2.2.1 REGISTRO RIGHE

È il registro in cui viene salvato il numero di righe dell'immagine.

### 2.2.2 REGISTRO COLONNE

È il registro in cui viene salvato il numero di colonne dell'immagine.

### 2.2.3 REGISTRO PIXEL

È il registro in cui viene salvato il pixel considerato quando si stanno valutando massimo e minimo valore dell'immagine.

### 2.2.4 REGISTRO PIXEL 2

È il registro in cui viene salvato il pixel considerato quando si sta calcolando il temp_pixel per l'immagine equalizzata.

### 2.2.5 MODULO CALCOLO AREA

Modulo nel quale, attraverso un contatore che viene aggiornato ogni ciclo, viene calcolata l'area dell'immagine. Quando il calcolo è completo il modulo segnalerà la fine del calcolo alzando il segnale end_area.

### 2.2.6 REGISTRO VALORE MASSIMO

È il registro dove viene salvato il valore massimo tra i pixel dell'immagine originale.

### 2.2.7 REGISTRO VALORE MINIMO

È il registro dove viene salvato il valore minimo tra i pixel dell'immagine originale.

### 2.2.8 MODULO CALCOLO DELTA VALUE

Modulo sottrattore che calcola la differenza tra valore massimo e valore minimo dell'immagine.

### 2.2.9 MODULO CALCOLO FLOOR

Modulo che, attraverso dei controlli a soglia, calcola la parte intera del logaritmo in base 2 del delta value calcolato precedentemente.

### 2.2.10 MODULO CALCOLO SHIFT LEVEL

Modulo che calcola lo shift level da applicare ad ogni pixel dell'immagine originale come 8 - il risultato del modulo floor.

### 2.2.11 REGISTRO CONTATORE INDIRIZZO

Registro che aggiorna ad ogni ciclo di clock il valore dell'indirizzo corrente e segnala, attraverso il segnale end_calc, quando tutta l'immagine è stata valutata.

### 2.2.12 MULTIPLEXER INDIRIZZO USCITA

Multiplexer che, a seconda del valore di o_addr_sel, assegna all'indirizzo in uscita:

- 0 per leggere il numero di colonne dell'immagine.
- 1 per leggere il numero di righe dell'immagine.
- curr_address per leggere i pixel dell'immagine originale.
- curr_address + area per scrivere i pixel dell'immagine equalizzata.

### 2.2.13 MODULO CALCOLO TEMP PIXEL

Modulo che calcola il temp_pixel shiftando a sinistra di shift_level posizioni la differenza tra pixel corrente considerato e pixel minimo dell'immagine.

### 2.2.14 MODULO SCRITTURA NUOVO PIXEL IN MEMORIA

Modulo che assegna a o_data il nuovo pixel da scrivere in memoria come min(temp_pixel, 255).

# 3. Risultati sperimentali

## 3.1 Report di sintesi

Il componente è risultato sintetizzabile correttamente senza errori o warning. Il massimo delay dovuto al percorso critico è di 6.061ns, quindi il componente rispetta ampiamente il requisito di un periodo di clock di 100 ns e funzionerà correttamente fino a circa 7ns di clock. Per quanto l'area, attraverso il comando report_utilization vivado riporta l'utilizzo di 164 LUT e 140 Flip Flop, mentre non vengono inferiti Latch.

Si riporta lo schematic effettuato da Vivado relativamente alla post synthesis:

## 3.2 Test benches

Per verificare il corretto comportamento del componente e assicurare la copertura di percorsi più ampia possibile ho definito, oltre ad alcuni test di immagini con valori e dimensioni casuali, i seguenti test:

**Test per i corner case delle immagini possibili in input:**

- **Test con immagine con tutti pixel uguali:**
  Essendo tutti i pixel uguali, il delta value sarà 0 e quindi tutti i pixel shiftati a 255.
- **Test con immagine già equalizzata**

La memoria di input contiene un'immagine già equalizzata, quindi il delta value sarà 255 e lo shift level 0, e quindi l'immagine di output deve essere uguale.

- **Test con immagine di dimensione 0**
  Test in cui ho un'immagine di dimensione 0, verifico che il calcolo dell'area termini correttamente.
- **Test con immagine di dimensione 1**
  Test di immagine di 1 pixel, verifico che il pixel venga correttamente conteggiato sia come massimo che come minimo e che venga shiftato a 255.

**Test per testare reset asincroni e immagini diverse**

- **Test con reset asincrono senza cambio di immagine**
  Test dove, dopo aver dato il segnale di start, aspetto 100ns e poi alzo il segnale di reset. In seguito non cambio l'immagine e verifico che la computazione avvenga correttamente.
- **Test con reset asincrono e cambio di immagine dopo il reset**

Test dove, dopo aver dato il segnale di start, aspetto 100ns e poi alzo il segnale di reset. In seguito al reset l'immagine in input cambia e verifico che venga correttamente equalizzata resettando registri con massimo e minimo valore iniziale.

- **Test con immagini successive e reset asincroni in mezzo**

  Test generale dove ho 3 immagini: dopo aver correttamente equalizzato la prima passo alla seconda ma durante la seconda computazione resetto e cambio dando in input una terza immagine. Questo test vuole testare in modo esteso che il componente reagisca correttamente a reset asincroni casuali e cambi immagine.

# 4. Conclusioni

Il componente sintetizzato supera correttamente tutti i test specificati sia in Behavioral che in Post Synthesis Functional.