Prova Finale (Progetto di Reti Logiche)

Prof. William Fornaciari - Anno 2020/2021

Tommaso Brumani (Codice Persona 10611219 - Matricola 908891)

1. Introduzione

Il progetto prevede l'implementazione in VHDL di una versione semplificata del metodo di equalizzazione dell'istogramma di un'immagine, ovvero un algoritmo che ha l'obiettivo di aumentare il contrasto di un'immagine in modo da occupare, se possibile, l'intero intervallo di intensità permesso dalla codifica dei suoi pixel.

Nella versione richiesta, l'algoritmo deve essere applicabile a immagini in scala di grigi con 256 livelli e aventi una dimensione massima di 128x128 pixel.

Figura 1: Esempi di immagini prima e dopo equalizzazione (fonte: Wikipedia).

L'algoritmo utilizzato per l'equalizzazione è qui riportato:

```
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` e `MIN_PIXEL_VALUE` sono, rispettivamente, i pixel dell'immagine con valore massimo e minimo, `CURRENT_PIXEL_VALUE` è il valore del pixel che si desidera alterare, e `NEW_PIXEL_VALUE` è il suo valore equalizzato da memorizzare.

L'immagine è memorizzata sequenzialmente in una memoria avente celle di dimensione 1 byte. I primi due di indirizzi della memoria (indirizzi 0 e 1) contengono, rispettivamente, le dimensioni di colonna e di riga dell'immagine, espresse in numero di pixel.

A partire dall'indirizzo 2 sono memorizzati i singoli pixel dell'immagine, in ordine sequenziale e in maniera contigua, mentre è richiesto che l'immagine equalizzata sia memorizzata a partire dall'indirizzo `2 +(num_colonne * num_righe)`, ovvero immediatamente dopo l'immagine originale.

2	Indirizzo 0:	Numero colonne	2
2	Indirizzo 1:	Numero righe	2
153	Indirizzo 2:	Pixel 1	153
128	Indirizzo 3:	Pixel 2	128
24	Indirizzo 4:	Pixel 3	24
86	Indirizzo 5:	Pixel 4	86
0	Indirizzo 6:	Pixel 1 (equalizzato)	255
0	Indirizzo 7:	Pixel 2 (equalizzato)	208
0	Indirizzo 8:	Pixel 3 (equalizzato)	0
0	Indirizzo 9:	Pixel 4 (equalizzato)	124
Pre-Equalizzazione			Post-Equalizzazione

(delta_value = 129) (shift_value = 1)

Figura 2: Esempio di contenuto della memoria prima e dopo l'equalizzazione (in decimale).

Il componente da realizzare ha la seguente interfaccia:

```
entity project_reti_logiche is
    port (
        i_clk       : in std_logic;
        i_rst       : in std_logic;
        i_start     : 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;
```

Dove:

- **`i_clk`** è il segnale di *clock* in ingresso al circuito, generato dai testbench;
- **`i_rst`** è il segnale di *reset* in ingresso al circuito, usato per l'inizializzazione;
- **`i_start`** è il segnale inviato in ingresso al circuito per indicare l'inizio di un'elaborazione;
- **`i_data`** è il segnale in ingresso a 8 bit che trasporta il contenuto della cella di memoria letta;
- **`o_address`** è il segnale in uscita a 16 bit che specifica l'indirizzo di memoria da cui leggere;
- **`o_done`** è il segnale in uscita che dichiara la fine dell'elaborazione e della scrittura del risultato;
- **`o_en`** è il segnale di *enable* in uscita, da inviare alla memoria per poter interagire con essa;
- **`o_we`** è il segnale di *write enable* in uscita, che va posto a valore '1' per scrivere in memoria, a valore '0' per leggere da essa;
- **`o_data`** è il segnale in uscita a 8 bit che trasporta il dato elaborato da scrivere in memoria.

Il segnale di *start* indica l'inizio dell'elaborazione, e rimane a valore '1' fino all'emissione del segnale *done*. Una nuova elaborazione può iniziare solo dopo che il segnale *done* sia stato riportato a valore '0'. La prima elaborazione è sempre preceduta da un *reset*.

2. Architettura

Il circuito è suddiviso in due moduli, uno rappresentante la *macchina a stati* e uno rappresentante il *data path* del circuito.

2.1. Data Path

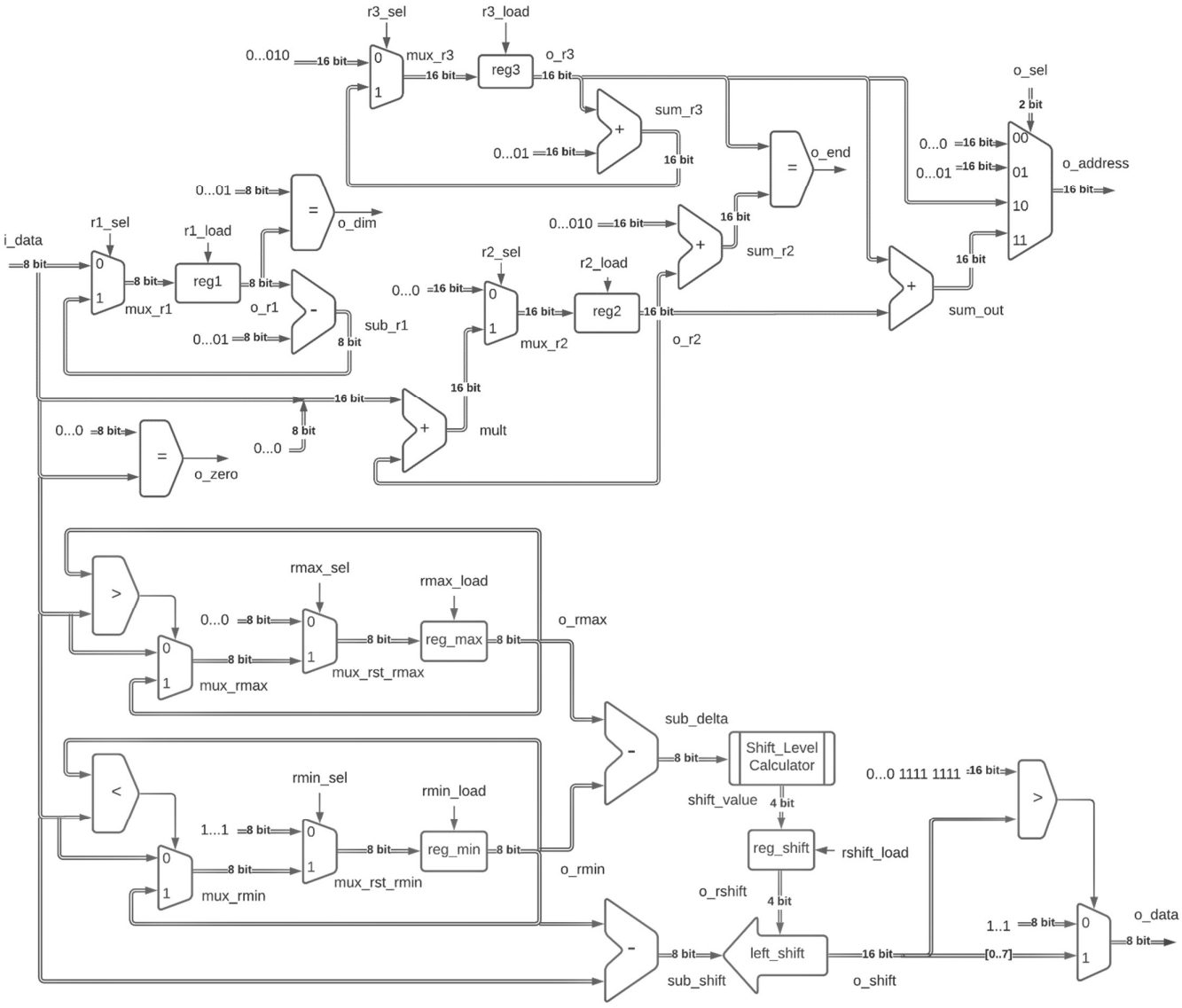

Figura 3: Rappresentazione del datapath del circuito (prima della sintesi).

Il datapath del circuito è stato realizzato all'interno di un'unica entity ed è rappresentato nella Figura 3. Sono presenti sei registri:

- **reg1** è il registro utilizzato per contenere il numero di colonne dell'immagine;
- **reg2** è il registro che agisce da accumulatore per calcolare la dimensione totale dell'immagine;
- **reg3** è il registro che agisce da contatore per determinare l'indirizzo di memoria a cui accedere;
- **reg_max** è il registro che contiene il pixel di valore massimo dell'immagine;
- **reg_min** è il registro che contiene il pixel di valore minimo dell'immagine;
- **reg_shift** è il registro che contiene lo `SHIFT_LEVEL` dell'immagine;

Per tutti i registri esiste un segnale di `load` che determina quando sostituirne il valore memorizzato con quello del segnale in ingresso.

I segnali principali del circuito sono:

- **o_end** è il segnale in uscita che determina la lettura dell'ultimo pixel dell'immagine, ed è posto a valore '1' quando `o_r3` e `o_r2 + 2` hanno lo stesso valore;
- **o_dim** è il segnale in uscita che determina la fine (al ciclo di clock successivo) del calcolo della dimensione dell'immagine, che avviene sommando il numero di righe a se stesso un numero di volte pari al numero di colonne. Il segnale è posto a valore '1' quando `o_r1 = 1`;
- **o_zero** è il segnale in uscita usato per riconoscere immagini di dimensione zero. Viene posto a valore '1' quando `i_data = 0` o `o_r1 = 0`;
- **r1_sel** è il segnale utilizzato per pilotare il *mux* in ingresso a `reg1`: assume valore '0' per caricare il numero di colonne dell'immagine nel registro, mentre assume valore '1' durante il calcolo della dimensione totale, facendo in modo che il valore di `reg1` decrementi di 1 a ogni ciclo di *clock*;
- **r2_sel**, **r3_sel**, **rmax_sel** e **rmin_sel** sono i segnali che controllano i *mux* all'ingresso dei rispettivi registri, portandoli al valore di inizializzazione quando il segnale assume valore '0';
- **sub_delta** è il segnale ricavato dalla sottrazione di `reg_max` e `reg_min` e rappresenta il `DELTA_VALUE` dell'immagine;
- **mult** è il segnale ottenuto dalla somma del contenuto di `reg2` con `i_data`, effettuata `num_colonne` volte per calcolare la dimensione totale dell'immagine, simulando una moltiplicazione;
- **shift_value** è il segnale ottenuto dall'applicazione di soglie al segnale `sub_delta` (processo rappresentato nella Figura 3 come "`shift_level calculator`"), al fine di ottenere lo `SHIFT_LEVEL` dell'immagine;
- **o_shift** è il segnale ottenuto dopo lo *shift sinistro* di (`CURRENT_PIXEL_VALUE - MIN_PIXEL_VALUE)` di un numero di bit pari a `SHIFT_LEVEL`: è confrontato con il numero 255, e il minore tra i due è portato in uscita su `o_data`.

Si specifica che, all'interno della Figura 3, i componenti che effettuano comparazioni tra segnali ("<" o ">") adottano la convenzione che il segnale in uscita sia posto a '1' qualora il segnale più "in alto" sia maggiore (per i componenti ">") o minore (per i componenti "<") di quello più "in basso".

2.2. Macchina a Stati

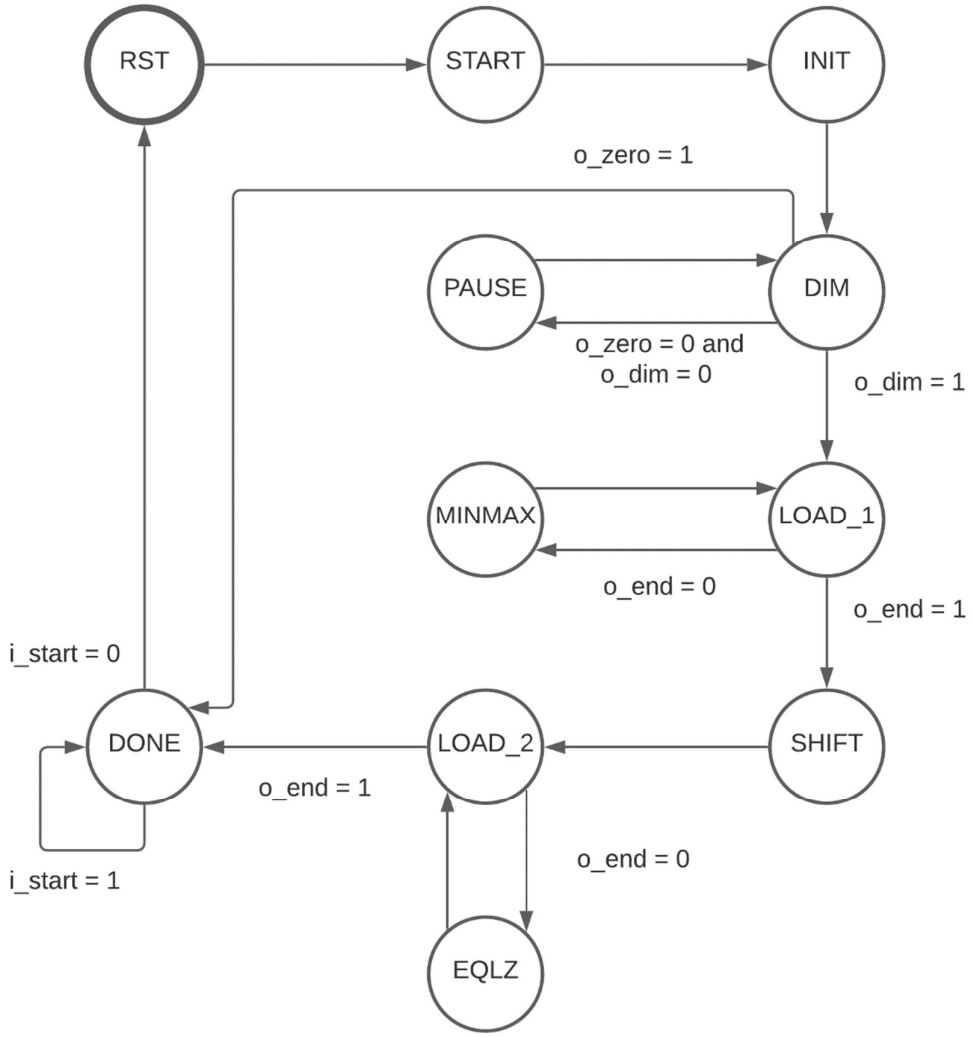

Figura 4: Rappresentazione della macchina a stati del circuito.

La macchina a stati del circuito è stata realizzata mediante due *process* principali: uno che descrive la transizione tra gli stati, e l'altro che descrive il funzionamento di ciascuno di essi.

Si specifica inoltre che la macchina a stati non è certamente minima, ma si è preferito adottare un approccio che favorisse maggiore semplicità e facilità di comprensione piuttosto che maggiore efficienza.

Gli 11 stati della macchina sono:

I. **RST:** lo stato iniziale e di reset del sistema, in cui `o_done` viene posto a '0' e il circuito è pronto per una nuova elaborazione.
II. **START:** lo stato di inizio dell'elaborazione in cui, dopo aver ricevuto il segnale di `i_start`, il circuito invia alla memoria il segnale di `o_en` e richiede la lettura dell'indirizzo 0.
III. **INIT:** lo stato in cui il circuito riporta al valore iniziale i registri `reg2` (0), `reg3` (2), `reg_max` (0) e `reg_min` (255), e carica il valore della cella di memoria a indirizzo 0 in `reg1`. Viene inoltre richiesta la lettura dell'indirizzo 1.
IV. **DIM:** lo stato in cui il circuito somma il valore della cella di memoria 1 al valore contenuto in `reg2`, per poi caricarlo sul medesimo registro: questo processo avviene un numero di volte pari a `num_colonne` al fine di calcolare la dimensione totale dell'immagine. Se però il `num_colonne` o `num_righe` sono uguali a 0, lo stato successivo diviene quello di fine elaborazione (`DONE`).
V. **PAUSE:** lo stato di attesa tra le somme utilizzate nel calcolo della dimensione totale dell'immagine, inserito per evitare alcune circostanze in cui la fine del calcolo (segnalata dall'assunzione del valore '1' da parte di `o_dim`) non veniva registrata dalla macchina a stati.
VI. **LOAD_1:** lo stato in cui il circuito richiede alla memoria la cella di indirizzo contenuto in `reg3` (ovvero uno dei pixel dell'immagine). Se però l'indirizzo richiesto è quello seguente all'ultimo pixel, lo stato successivo diviene `SHIFT`.
VII. **MINMAX:** lo stato in cui il circuito riceve in ingresso su `i_data` la cella di memoria con indirizzo contenuto in `reg3`, il valore del quale è quindi incrementato di 1. Il valore del pixel letto viene invece confrontato con quelli contenuti in `reg_max` e `reg_min`, aggiornando questi ultimi se necessario.
VIII. **SHIFT:** lo stato in cui il circuito utilizza i contenuti di `reg_max` e `reg_min` per calcolare lo `SHIFT_LEVEL` dell'immagine e caricarlo in `reg_shift`. Viene inoltre riportato `reg3` al suo valore iniziale (2).
IX. **LOAD_2:** lo stato in cui il circuito richiede alla memoria la cella di indirizzo contenuto in `reg3` affinché il suo contenuto possa venire equalizzato. Se però l'indirizzo richiesto è quello seguente all'ultimo pixel, lo stato successivo diviene `DONE`.
X. **EQLZ:** lo stato in cui il circuito elabora il pixel in ingresso su `i_data` secondo il processo di equalizzazione dell'immagine, per poi scrivere il risultato all'indirizzo `(2 + indirizzo_pixel_letto + num_colonne * num_righe)`. Il contenuto di `reg3` è inoltre incrementato di 1.
XI. **DONE:** lo stato di fine elaborazione, in cui il circuito ha terminato il processo di equalizzazione (inclusa la scrittura di memoria) e pone a '1' il segnale `o_done`, per poi tornare allo stato iniziale una volta che il segnale `i_start` sia stato portato a '0'.

3. Risultati Sperimentali

3.1. Sintesi

```
Timing Report

Slack (MET) :             94.871ns  (required time - arrival time)
  Source:                 DATAPATH0/o_r2_reg[3]/C
                            (rising edge-triggered cell FDCE clocked by Clock  {rise@0.000ns fall@5.000ns period=100.000ns})
  Destination:            FSM_onehot_cur_state_reg[5]/D
                            (rising edge-triggered cell FDCE clocked by Clock  {rise@0.000ns fall@5.000ns period=100.000ns})
  Path Group:             Clock
  Path Type:              Setup (Max at Slow Process Corner)
  Requirement:            100.000ns  (Clock rise@100.000ns - Clock rise@0.000ns)
  Data Path Delay:        4.978ns  (logic 2.727ns (54.781%)  route 2.251ns (45.219%))
  Logic Levels:           7  (CARRY4=5 LUT2=1 LUT6=1)
  Clock Path Skew:        -0.145ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD):    2.100ns = ( 102.100 - 100.000 )
    Source Clock Delay      (SCD):    2.424ns
    Clock Pessimism Removal (CPR):    0.178ns
  Clock Uncertainty:      0.035ns  ((TSJ^2 + TIJ^2)^1/2 + DJ) / 2 + PE
    Total System Jitter     (TSJ):    0.071ns
    Total Input Jitter      (TIJ):    0.000ns
    Discrete Jitter          (DJ):    0.000ns
    Phase Error              (PE):    0.000ns
```

Figura 5: Timing Report.

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	185	0	134600	0.14
LUT as Logic	185	0	134600	0.14
LUT as Memory	0	0	46200	0.00
Slice Registers	67	0	269200	0.02
Register as Flip Flop	67	0	269200	0.02
Register as Latch	0	0	269200	0.00
F7 Muxes	0	0	67300	0.00
F8 Muxes	0	0	33650	0.00

Figura 6: Utilization Report.

Il componente risulta correttamente sintetizzabile e simulabile (sia pre-sintesi che post-sintesi) con ogni *test bench* utilizzato.

Il *timing report* (Figura 5) registra un *data path delay* massimo di 4,978ns, con un conseguente *slack* di 94,871ns rispetto al periodo di clock minimo richiesto dalla specifica (100ns). Il clock del circuito potrebbe quindi essere reso fino a 20 volte più rapido senza invalidare il corretto funzionamento del circuito.

L'*utilization report* (Figura 6) registra l'utilizzo di 185 *lookup tables* (lo 0,14% del totale disponibile) e di 67 *flip flop* (lo 0,02% del totale).

Si noti inoltre che il processo di sintesi non ha inferito alcun *latch*.

3.2. Simulazioni

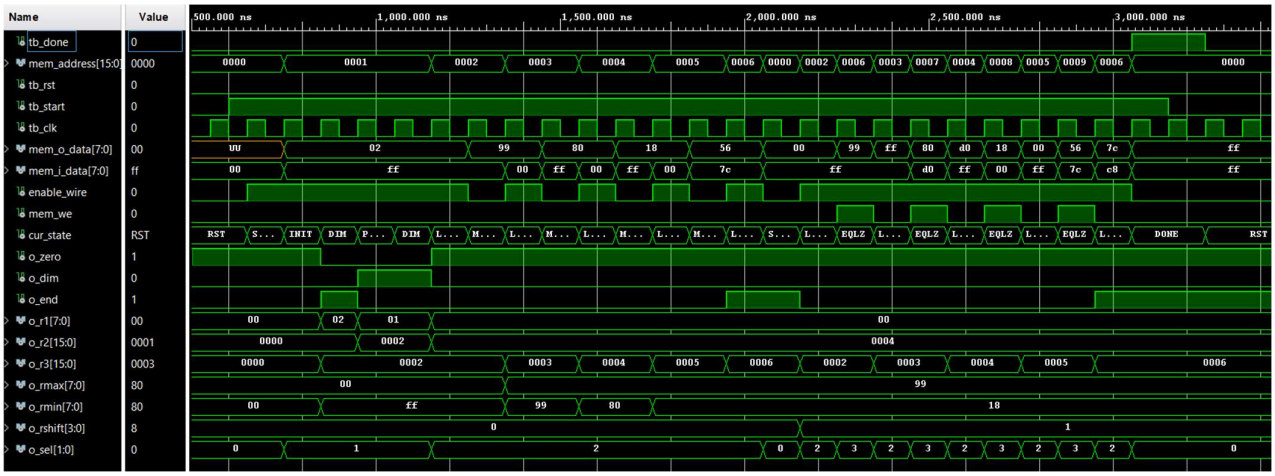

Figura 7: Simulazione del circuito per l'immagine di esempio in Figura 2.

Il circuito è stato messo alla prova attraverso diversi *test bench* per verificarne il corretto funzionamento. Questi sono:

I. Una serie di test costituiti da immagini generate casualmente, al fine di accertare il giusto comportamento del circuito in circostanze di utilizzo "comune". Nel corso di questi test sono state verificate anche le condizioni: immagini multiple consecutive, `SHIFT_LEVEL` da '0' a '8', `TEMP_PIXEL` < 255 e `TEMP_PIXEL` > 255.
II. Un test costituito da un'immagine avente `DELTA_VALUE` massimo, ovvero avente un pixel di valore '0' e un pixel di valore '255'.
III. Un test costituito da immagini aventi `DELTA_VALUE` minimo, in particolare aventi in un caso tutti i pixel con valore '0', e in un altro aventi tutti valore '255'.
IV. Un test costituito da un'immagine di dimensione massima (128 x 128) e una di dimensione minima (1 x 1).
V. Un test costituito da immagini aventi dimensione 0 (0 x 0, N x 0, 0 x N).
VI. Un test in cui l'elaborazione di un immagine è stata interrotta da un segnale di *reset*, per poi eseguire l'equalizzazione di un'altra immagine.

Le forme d'onda delle simulazioni non sono state incluse nel documento poiché la loro estensione ne renderebbe nella maggior parte dei casi impossibile una lettura accurata.

4. Conclusioni

Il circuito funziona come previsto in tutti i casi d'uso messi alla prova nei *test bench*, sia nelle simulazioni pre-sintesi che in quelle post-sintesi.

La durata massima dell'elaborazione (dall'istante in cui il segnale i_start viene portato a '1' a quello in cui o_done viene riportato a '0') calcolata con un *clock* di 100ns è di circa 6580µs, ed è riscontrata nell'equalizzazione di immagini con dimensione 128 x 128.

La durata dell'elaborazione di immagini di dimensione minima (1 x 1) è invece di circa 1,2µs, mentre quella di immagini di dimensione 0 è di circa 500ns per le immagini 0x0 e 0xN, e di circa 700ns per quelle Nx0.

Per quanto la velocità di elaborazione potrebbe essere aumentata (ad esempio diminuendo la dimensione della macchina a stati o integrando il processo di ricerca dei pixel di valore massimo e minimo con il calcolo della dimensione totale dell'immagine) la *performance* ottenuta è stata ritenuta sufficiente per le specifiche date, e un buon compromesso tra efficienza e semplicità.