Prova Finale - Progetto Reti Logiche

Prof. W. Fornaciari - Anno 2020/2021

Simone Buranti (Codice Persona 10687994 - Matricola 912036)
Alessandro Bagnacani (Codice Persona 10662176 - Matricola 911395)

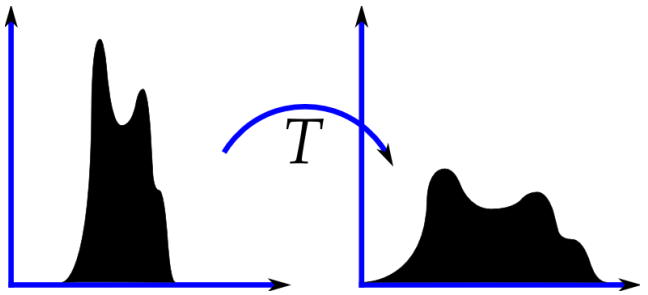
Indice

1 Introduzione

1.1 Scopo del Progetto

Lo scopo del progetto consiste nella realizzazione di un componente hardware, descritto in VHDL, che data in ingresso un'immagine in scala di grigi e di dimensioni variabili restituisce la corrispondente immagine equalizzata. L'algoritmo implementato dal componente si ispira al metodo di equalizzazione dell'istogramma, un metodo di elaborazione digitale delle immagini con cui è possibile regolarne il contrasto basandosi sull'istogramma dell'immagine. L'istogramma di un'immagine, infatti, descrive graficamente la distribuzione tonale di un'immagine digitale tracciando il numero di pixel per ogni valore tonale.

Le immagini il cui istogramma è distribuito su un ristretto intervallo di gradazioni risultano di difficile leggibilità dovuta al basso livello di contrasto. Una corretta equalizzazione provvede quindi a ricalibrare le rispettive gradazioni di colore dei pixel con lo scopo di mantenerne il reciproco dislivello ma proiettandolo su un intervallo di tonalità più ampio.



Istogramma di un'immagine - circoscritto in un ristretto intervallo di tonalità	*Istogramma dell'immagine equalizzata - distribuito su un intervallo di maggiore ampiezza*

Esempio dell'effetto del processo di equalizzazione sull'istogramma di un'immagine

1.2 Specifica del Progetto

La versione sviluppata rappresenta una versione semplificata dell'algoritmo standard. L'algoritmo di equalizzazione è applicato solo ad immagini in scala di grigi a 256 livelli di dimensioni non maggiori di 128x128 e trasforma i pixel nel modo seguente:

1. DELTA_VALUE = MAX_PIXEL_VALUE – MIN_PIXEL_VALUE
2. SHIFT_LEVEL = (8 – FLOOR(LOG2(DELTA_VALUE +1))))
3. TEMP_PIXEL = (CURRENT_PIXEL_VALUE - MIN_PIXEL_VALUE) << SHIFT_LEVEL
4. NEW_PIXEL_VALUE = MIN(255 , TEMP_PIXEL)

Dove al primo passaggio viene calcolato il range tonale dell'immagine in ingresso; questo viene poi utilizzato per ricavare il coefficiente di scala necessario per sfruttare l'intero range di grigi a disposizione. Ogni pixel viene in seguito gestito in modo tale da aumentare il contrasto dell'immagine.

Low Contrast	High Contrast	Low Contrast	High Contrast

Esempi di immagini pre e post equalizzazione (sorgente Google Immagini)

Il modulo è stato progettato in modo tale da codificare più immagini sequenzialmente, ossia è possibile elaborare una quantità indefinita di immagini purché ognuna di esse non cambi durante la computazione.
Il modulo inoltre necessita di un segnale di reset al solo avvio dello stesso, mentre ogni successiva esecuzione ne è completamente svincolata.

1.3 Interfaccia del componente e memoria

Il componente realizzato espone 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;
```

In particolare:

- i_clk è il segnale di CLOCK in ingresso generato dal TestBench;
- i_rst è il segnale di RESET che inizializza la macchina pronta per ricevere il primo segnale di START;
- i_start è il segnale di START generato dal Test Bench;
- i_data è il segnale (vettore) che arriva dalla memoria in seguito ad una richiesta di lettura;
- o_address è il segnale (vettore) di uscita che manda l'indirizzo alla memoria;
- o_done è il segnale di uscita che comunica la fine dell'elaborazione e il dato di uscita scritto in memoria;
- o_en è il segnale di ENABLE da dover mandare alla memoria per poter comunicare (sia in lettura che in scrittura);
- o_we è il segnale di WRITE ENABLE da dover mandare alla memoria (=1) per poter scriverci. Per leggere da memoria esso deve essere 0;
- o_data è il segnale (vettore) di uscita dal componente verso la memoria.

Per quanto riguarda l'immagine da processare, essa è salvata in una memoria con indirizzamento al Byte a partire dall'indirizzo di posizione 0. In questo modo i primi due indirizzi fanno riferimento alle dimensioni dell'immagine, mentre i successivi contengono gli effettivi valori rappresentanti i livelli di colorazione. I successivi indirizzi saranno quindi riservati ai pixel elaborati dell'immagine equalizzata.

Di seguito una rappresentazione grafica dell'organizzazione interna della memoria:

Numero Colonne	Indirizzo 0
Numero Righe	Indirizzo 1
Primo pixel input	Indirizzo 2
...	...
...	...
Ultimo pixel input	Indirizzo N-Col*N-Rig +1
Primo pixel output	Indirizzo N-Col*N-Rig +2
...	...
...	...
Ultimo pixel output	Indirizzo 2*N-Col*N-Rig +1

2. Design

2.1 Scelte progettuali

Il componente realizzato si divide principalmente di due moduli fisicamente e concettualmente distinti: il "datapath" e la "finite state machine".
Il primo rappresenta il componente circuitale adibito al calcolo effettivo dei valori richiesti ed è a sua volta costituito di moduli funzionali subordinati.
Il secondo al contrario si occupa di gestire il primo assicurando la computazione dei dati corretti al momento opportuno.

2.2 Macchina a Stati Finiti e Datapath

La macchina a stati costruita è una macchina di Moore composta da 27 stati. Di seguito sono riportate una descrizione integrale della macchina a stati in cui si evidenziano i gruppi funzionali principali e una rappresentazione grafica dei componenti con cui essa comunica.

2.2.1 Gruppo Bianco

Gli stati contraddistinti dal colore bianco rappresentano le attività che riportano alla fase di inizio esecuzione.

2.2.2 Gruppo Azzurro

Il gruppo funzionale composto dagli stati marcati in azzurro è deputato alla lettura delle due dimensioni dell'immagine. Nel caso in cui una delle due sia nulla il sistema pone a 1 il segnale corrispondente e la computazione termina istantaneamente riportandosi in uno stato reset.

2.2.3 Gruppo Giallo

L'insieme degli stati marcati col colore giallo costituisce la parte della macchina adibita al calcolo della dimensione totale dell'immagine. A livello di datapath si è scelto di produrre tale risultato mediante somme successive. Una volta determinato tale risultato, esso viene salvato nei tre registri coinvolti nella gestione degli indirizzi: R1, R2, R3.

2.2.4 Gruppo Verde

Il gruppo degli stati verdi rappresenta i passi da compiere per trovare il massimo e il minimo dei valori tonali dell'immagine in ingresso. La parte del datapath che si occupa di questo obiettivo è ovviamente situata nella regione dei registri di massimo e minimo. Il registro R1 e i componenti legati ad esso, si occupano della lettura dei pixel dell'immagine salvata in memoria.

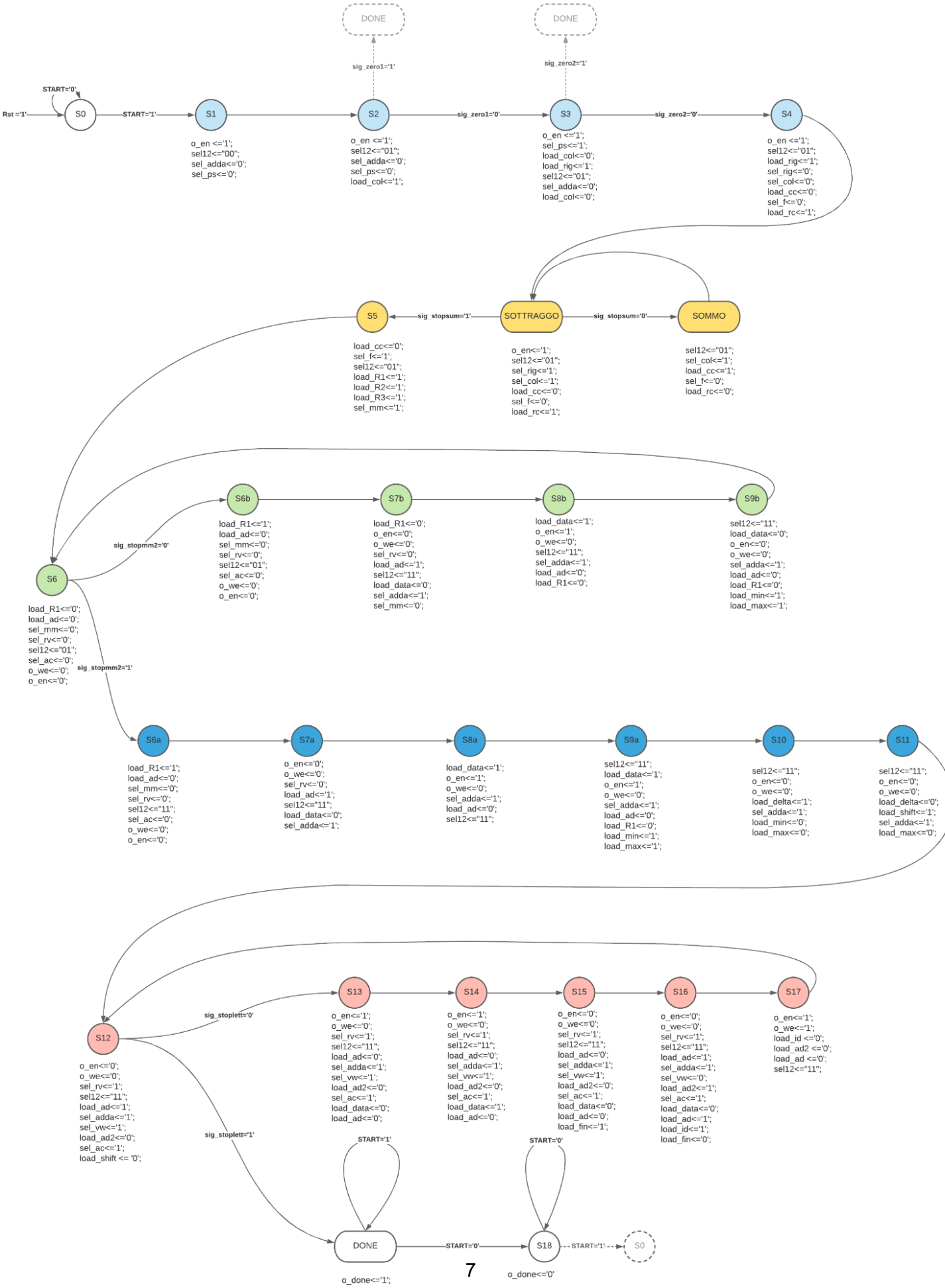
2.2.5 Gruppo Ciano

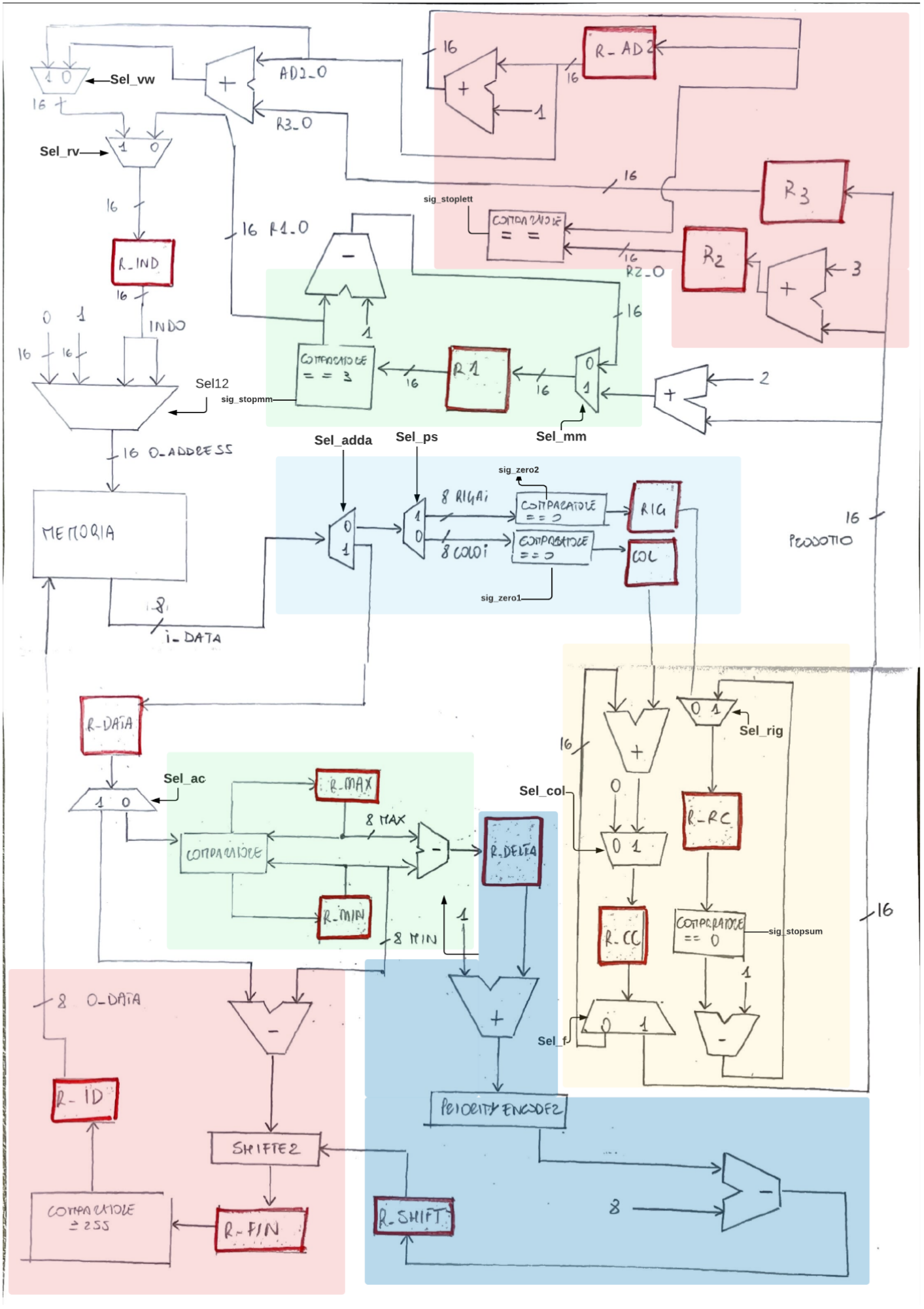
Gli stati colorati in ciano sono adibiti al calcolo dello SHIFT che verrà applicato ad ogni pixel dell'immagine.

2.2.6 Gruppo Rosso

L'insieme degli stati in rosso si occupano, infine, dell'effettiva procedura di lettura dell'immagine di input e la successiva scrittura dei valori tonali dell'immagine equalizzata agli indirizzi corrispondenti. Per quanto riguarda il datapath, i registri che intervengono nella gestione degli indirizzi sono R_AD per quelli di lettura, a cui si somma il contenuto di R3 (offset) per ottenere quelli di scrittura mentre R2 funge da condizione di uscita dal ciclo.
Il valore di uscita è ottenuto sottraendo al valore di ingresso il minimo valore tonale; questo viene poi scalato del valore di SHIFT calcolato precedentemente e infine viene effettuato un confronto con il limite massimo di 255 prima di inviarlo in uscita, come da specifica.

Una volta completata anche questa fase il sistema alza un segnale di DONE che rimarrà alto finché il segnale di START rimane alto. Quando entrambi saranno riportati bassi il sistema resterà in attesa di un nuovo segnale di START per iniziare una nuova computazione.


DONE
DONE
sig_zero1='1'
sig_zero2='1'
START='0'
Rst ='1'
S0
START='1'
S1
S2
sig_zero1='0'
S3
sig_zero2='0'
S4
o_en <='1';
sel12<="00";
sel_adda<='0';
sel_ps<='0';
o_en <='1';
sel12<="01";
sel_adda<='0';
sel_ps<='0';
load_col<='1';
o_en <='1';
sel_ps<='1';
load_col<='0';
load_rig<='1';
sel12<="01";
sel_adda<='0';
load_col<='0';
o_en <='1';
sel12<="01";
load_rig<='1';
sel_rig<='0';
sel_col<='0';
load_cc<='0';
sel_f<='0';
load_rc<='1';
S5
sig_stopsum='1'
SOTTRAGGO
sig_stopsum='0'
SOMMO
load_cc<='0';
sel_f<='1';
sel12<="01";
load_R1<='1';
load_R2<='1';
load_R3<='1';
sel_mm<='1';
o_en<='1';
sel12<="01";
load_rig<='1';
sel_rig<='1';
sel_col<='1';
load_cc<='0';
sel_f<='0';
load_rc<='1';
sel12<="01";
sel_col<='1';
load_cc<='1';
sel_f<='0';
load_rc<='0';
S6
sig_stopmm2='0'
S6b
S7b
S8b
S9b
load_R1<='1';
load_ad<='0';
sel_mm<='0';
sel_rv<='0';
sel12<="01";
sel_ac<='0';
o_we<='0';
o_en<='0';
load_R1<='0';
o_en<='0';
o_we<='0';
sel_rv<='0';
load_ad<='1';
sel12<="11";
load_data<='0';
sel_adda<='1';
sel_mm<='0';
load_data<='1';
o_en<='1';
o_we<='0';
sel12<="11";
sel_adda<='1';
load_ad<='0';
load_R1<='0';
sel12<="11";
load_data<='0';
o_en<='0';
o_we<='0';
sel_adda<='1';
load_ad<='0';
load_R1<='0';
load_min<='1';
load_max<='1';
load_R1<='0';
load_ad<='0';
sel_mm<='0';
sel_rv<='0';
sel12<="01";
sel_ac<='0';
o_we<='0';
o_en<='0';
sig_stopmm2='1'
S6a
S7a
S8a
S9a
S10
S11
load_R1<='1';
load_ad<='0';
sel_mm<='0';
sel_rv<='0';
sel12<="11";
sel_ac<='0';
o_we<='0';
o_en<='0';
o_en<='0';
o_we<='0';
sel_rv<='0';
load_ad<='1';
sel12<="11";
load_data<='0';
sel_adda<='1';
load_data<='1';
o_en<='1';
o_we<='0';
sel_adda<='1';
load_ad<='0';
sel12<="11";
sel12<="11";
load_data<='1';
o_en<='1';
o_we<='0';
sel_adda<='1';
load_ad<='0';
load_R1<='0';
load_min<='0';
load_max<='1';
sel12<="11";
o_en<='0';
o_we<='0';
load_delta<='1';
sel_adda<='1';
load_min<='0';
load_max<='0';
sel12<="11";
o_en<='0';
o_we<='0';
load_delta<='0';
load_shift<='1';
sel_adda<='1';
load_max<='0';
S12
sig_stoplett='0'
S13
S14
S15
S16
S17
o_en<='0';
o_we<='0';
sel_rv<='1';
sel12<="11";
load_ad<='1';
sel_adda<='1';
sel_vw<='1';
load_ad2<='0';
sel_ac<='1';
load_shift <= '0';
o_en<='1';
o_we<='0';
sel_rv<='1';
sel12<="11";
load_ad<='0';
sel_adda<='1';
sel_vw<='1';
load_ad2<='0';
sel_ac<='1';
load_data<='0';
load_ad<='0';
o_en<='1';
o_we<='0';
sel_rv<='1';
sel12<="11";
load_ad<='0';
sel_adda<='1';
sel_vw<='1';
load_ad2<='0';
sel_ac<='1';
load_data<='1';
load_ad<='0';
o_en<='0';
o_we<='0';
sel_rv<='1';
sel12<="11";
load_ad<='0';
sel_adda<='1';
sel_vw<='1';
load_ad2<='0';
sel_ac<='1';
load_data<='0';
load_ad<='0';
load_fin<='1';
o_en<='0';
o_we<='0';
sel_rv<='1';
sel12<="11";
load_ad<='1';
sel_adda<='1';
sel_vw<='0';
load_ad2<='1';
sel_ac<='1';
load_data<='0';
load_ad<='1';
load_id<='1';
load_fin<='0';
o_en<='1';
o_we<='1';
load_id <='0';
load_ad2 <='0';
load_ad <='0';
sel12<="11";
sig_stoplett='1'
START='1'
START='0'
DONE
START='0'
S18
START='1'
S0
o_done<='1';
o_done<='0'


Sel_vw
AD2_O
R3_O
R_AD2
Sel_rv
R3
sig_stoplett
COMPARATORE
R2
R2_O
R_IND
R1_O
INDO
R1
Sel12
sig_stopmm
Sel_adda
Sel_ps
Sel_mm
16 O_ADDRESS
sig_zero2
8 RIGAi
8 COLOi
RIG
COL
sig_zero1
MEMORIA
i_DATA
PESOTTO
R_DATA
Sel_ac
R_MAX
8 MAX
R_MIN
8 MIN
R_DELTA
Sel_rig
Sel_col
R_RC
R_CC
sig_stopsum
8 O_DATA
R_ID
SHIFTE2
PRIORITY ENCODER
COMPARATORE ≥ 255
R_FIN
R_SHIFT

Per ragioni di leggibilità si è scelto di riportare nella rappresentazione grafica del datapath solo i segnali di controllo principali. I segnali di gestione dei registri possono essere ricavati seguendo la nomenclatura "load_nomeregistro".

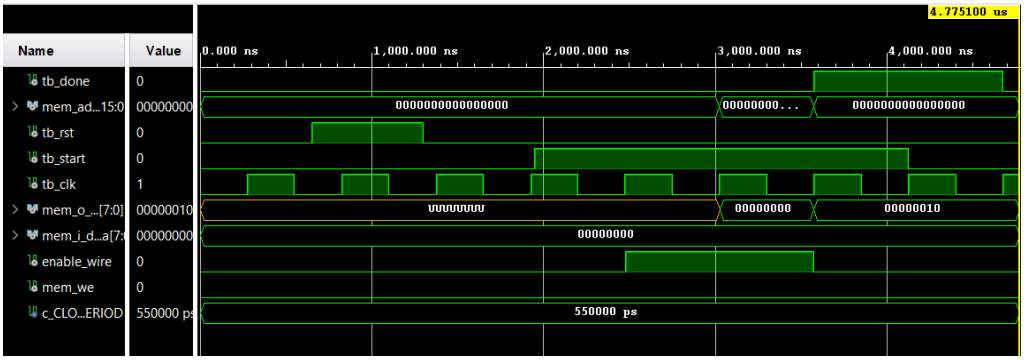
3. Test e risultati

3.1 Corner Cases

Al fine di coprire il maggior numero di percorsi potenzialmente critici si è scelto di stimolare preventivamente il componente realizzato con specifici e mirati test cases, per poi proseguire nell'attività di testing con testbenches più impegnativi e completi.

Test - 0 pixel -

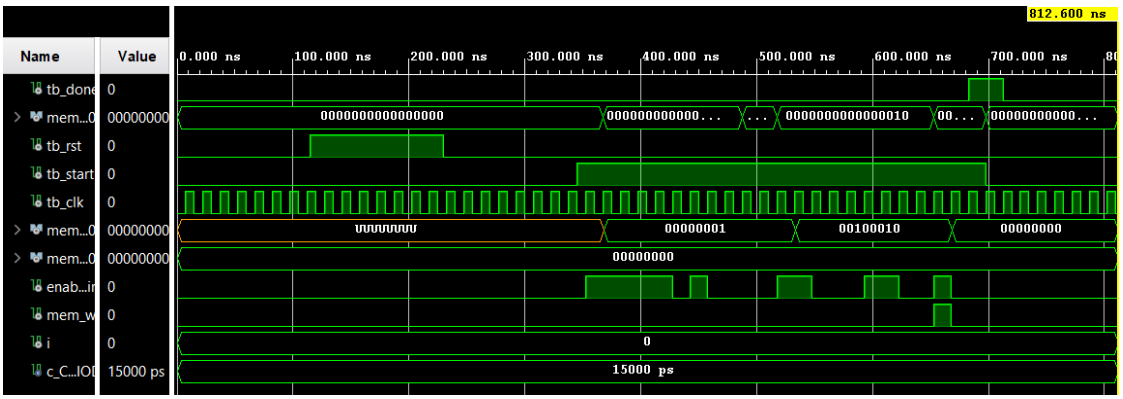
Questo primo test case porta l'attenzione sul semplice caso specifico in cui l'immagine ha almeno una dimensione nulla ed è quindi priva di pixel. Il componente ha risposto correttamente sia al caso in cui una sola delle due dimensioni fosse nulla sia al caso in cui lo sono entrambe non compiendo alcuna azione sulla memoria e portandosi in uno stato consistente di "ready" .



Waveform nel caso in cui la sola prima dimensione è nulla

Test - 1 pixel -

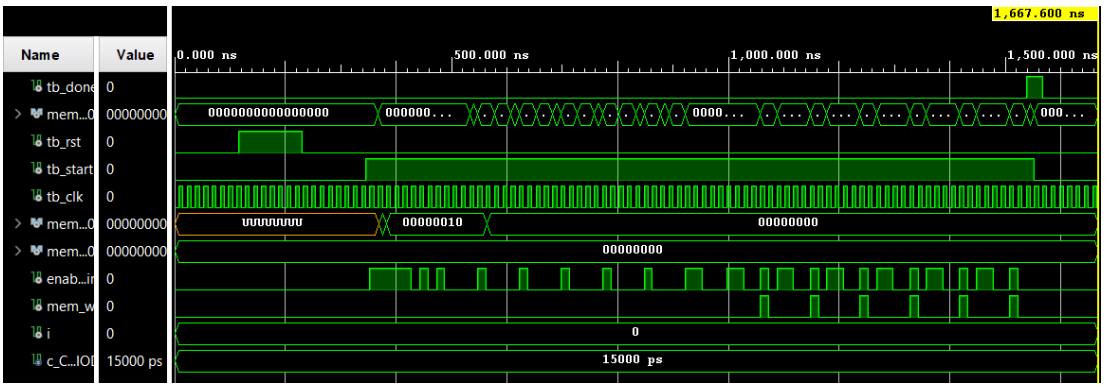
Nonostante l'analogia col precedente, questo test verifica il corretto funzionamento di tutti i cicli presenti all'interno del componente nello specifico caso in cui il pixel sia unico. Anche in questo caso la risposta ottenuta è stata positiva.



Waveform nel caso di 1 pixel

Test - Tutti 0 -

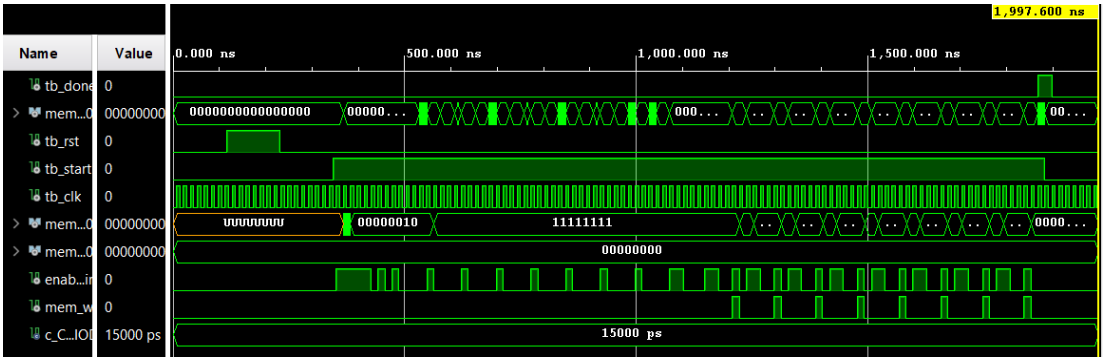
In questo testbench è stata inserita un'immagine "tinta unita" di piccole dimensioni. In particolare l'unico valore che figura nella memoria in questo caso è 0 e l'obiettivo è quello di coprire il caso particolare in cui nell'algoritmo di equalizzazione sia presente solo il valore 0.



Waveform nel caso in cui ogni pixel ha valore "00000000"

Test - Tutti 255 -

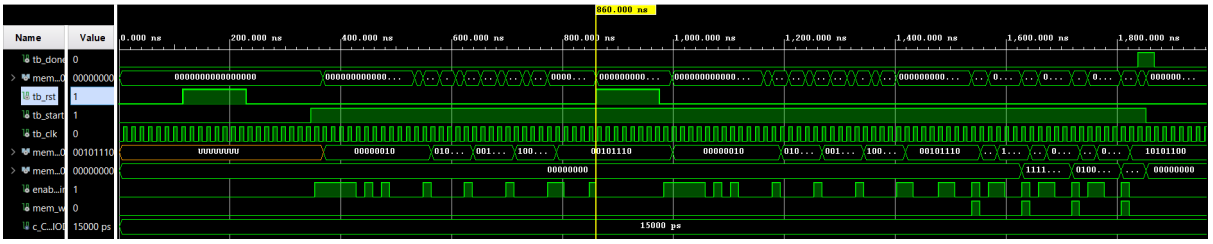
Caso opposto al precedente, è stata considerata un'immagine "tinta unita" di piccole dimensioni. L'unico valore che figura nella memoria in questo caso è 255 e l'obiettivo è quello di coprire il caso in cui nell'algoritmo di equalizzazione sia presente il valore massimo.



Waveform nel caso in cui ogni pixel ha valore "11111111"

Test - Reset asincrono -

In questo caso lo scopo del test è verificare il corretto comportamento del componente nel caso in cui venga fornito un segnale di reset durante una computazione. Come da specifica il segnale di reset risulta essere asincrono in quanto come si può evincere dall'immagine riportata di seguito in corrispondenza del fronte di salita del reset tutto il circuito azzera la computazione riportandosi allo stato iniziale; inoltre tutto questo avviene senza seguire il ciclo di clock confermando che tale comportamento è proprio di un segnale asincrono.

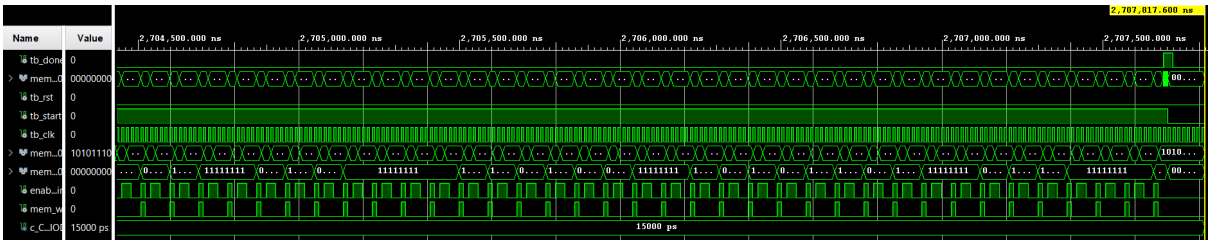


Waveform nel caso di reset durante la computazione

Test - Immagine 128x128 -

Se prima si sono verificati casi di piccole dimensioni ma specifici, lo scopo di questo test case è proprio quello di stressare il componente nel caso di un'immagine di dimensioni massime, ottenendo informazioni sulla robustezza dello stesso e sulla corretta gestione dell'elaborazione e dell'indirizzamento di immagini di massime dimensioni.

Nonostante la waveform sia troppo densa per poter distinguere i segnali, la si riporta per avere un confronto del tempo di simulazione rispetto ai casi precedenti.



Waveform nel caso di immagine 128x128

Tutti i test case precedentemente elencati fanno riferimento alla simulazione post-sintesi in quanto si è scelto, a discapito di una maggiore leggibilità dei segnali interni specificati nella macchina a stati, di riportare il comportamento tendenzialmente più soggetto a criticità.

In seguito a tali test, inoltre, si è proseguita l'attività di testing cercando di sollecitare il componente con immagini di dimensioni e valori variabili e talvolta tramite computazioni

successive. Il componente si è sempre mostrato solido e affidabile superando i test sottoposti sia in pre che in post sintesi.

Una volta coperti anche i casi più comuni si è sottoposta una batteria di test cases di vario genere in numero pari a 10.000: anche a seguito di questa operazione non sono state verificate criticità.

4. Conclusioni

Il componente sintetizzato supera correttamente tutti i test specificati nelle 2 simulazioni: Behavioral e Post-Synthesis Functional.
Consultando alcuni dei report che la TCL Console mette a disposizione è inoltre possibile analizzare altri aspetti propri del progetto realizzato.

In particolare osservando le informazioni esposte dal “timing report” si nota che lo “slack” del circuito è estremamente ridotto rispetto ai limiti richiesti dalla specifica, in quanto il ritardo legato alla computazione del datapath ammonta a poco più di 5 ns. Il componente potrebbe di conseguenza processare lo stesso tipo di immagini ad una velocità pari a 1/(5,052ns), ossia ad una frequenza di circa 198 MHz.
Tale risultato è il frutto di scelte progettuali orientate a ridurre la lunghezza dei cammini critici nel datapath a discapito di un uso maggiore di registri di memoria.

```
------------------------------------------------------------------------------------

Timing Report

Slack (MET) :             5.052ns  (required time - arrival time)
  Source:                 DATAPATH0/ad2o_reg[4]/C
                            (rising edge-triggered cell FDCE clocked by clock  {rise@0.000ns fall@5.000ns period=10.000ns})
  Destination:            FSM_onehot_curr_state_reg[15]/D
                            (rising edge-triggered cell FDCE clocked by clock  {rise@0.000ns fall@5.000ns period=10.000ns})
  Path Group:             clock
  Path Type:              Setup (Max at Slow Process Corner)
  Requirement:            10.000ns  (clock rise@10.000ns - clock rise@0.000ns)
  Data Path Delay:        4.797ns  (logic 2.708ns (56.452%)  route 2.089ns (43.548%))
  Logic Levels:           7  (CARRY4=5 LUT6=2)
  Clock Path Skew:        -0.145ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD):    2.100ns = ( 12.100 - 10.000 )
    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
```