Prova Finale di Reti Logiche
2020-2021

Natalia Bagnoli
Matricola: 909725
Codice Persona: 10633002

Pasquale Castiglione
Matricola: 910188
Codice Persona: 10657816

Docente: William Fornaciari

POLITECNICO
MILANO 1863

Indice

1 Introduzione

1.1 Descrizione del Progetto

Il progetto richiesto consiste nell'implementazione in VHDL del metodo di equalizzazione dell'istogramma di un'immagine. Tale metodo permette di incrementare il contrasto di un'immagine distribuendo le intensità dei pixel su tutto l'istogramma.
In particolare al modulo è richiesto di:

1. Accedere ad una memoria RAM e leggerne il contenuto
2. Effettuare il calcolo del nuovo valore d'intensità per ogni pixel nel modo seguente:
```
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)
```
3. Scrivere il risultato nella RAM

L'implementazione deve essere poi sintetizzata con target FPGA xc7a200tfbg484-1.

1.2 Esempio

INDIRIZZO MEMORIA	VALORE	
0	2	Numero colonne
1	2	Numero righe
2	46	Primo pixel
3	131	Secondo pixel
4	62	Terzo pixel
5	89	Quarto e ultimo pixel
6	0	Primo pixel nuova immagine
7	255	Secondo pixel nuova immagine
8	64	Terzo pixel nuova immagine
9	172	Quarto e ultimo pixel nuova immagine

2 Implementazione

2.1 Scelte Progettuali

Si è scelto di implementare il metodo mediante l'utilizzo di una Macchina a Stati Finiti costituita da 21 stati, usando una architettura di tipo *Behavioral*. Tale *architecture* è costituita da due *process*:

- `Sych`: Si occupa della gestione del segnale asincrono di reset e dell'assegnazione del nuovo stato.
- `Comb`: Si occupa della logica combinatoria.

2.2 Tabella dei Signals

Durante la computazione il componente fa uso dei seguenti *signals*:

Nome	Tipo	Descrizione
n_col	std_logic_vector(7 downto 0)	Numero di colonne dell'immagine
n_row	std_logic_vector(7 downto 0)	Numero di righe dell'immagine
counter	std_logic_vector(7 downto 0)	Minimo tra n_col e n_row
size	std_logic_vector(15 downto 0)	Numero di pixel dell'immagine
temp_address	std_logic_vector(15 downto 0)	Indirizzo d'interesse per la lettura della memoria
current_pixel_value	std_logic_vector(7 downto 0)	Valore del pixel d'interesse letto dalla memoria
max_pixel_value	std_logic_vector(7 downto 0)	Valore massimo d'intensità dell'immagine
min_pixel_value	std_logic_vector(7 downto 0)	Valore minimo d'intensità dell'immagine
DELTA_VALUE	std_logic_vector(7 downto 0)	max_pixel_value - min_pixel_value
DELTA_VALUE_1	std_logic_vector(8 downto 0)	max_pixel_value - min_pixel_value + 1
floor_log2	std_logic_vector(3 downto 0)	Parte intera del logaritmo in base due del DELTA_VALUE_1
shift_level	std_logic_vector(3 downto 0)	Valore di shift necessario ai fini dell'elaborazione
diff_pixel_value	std_logic_vector(7 downto 0)	current_pixel_value - min_pixel_value
temp_pixel_value	std_logic_vector(15 downto 0)	diff_pixel_value << shift_level
new_pixel_value	std_logic_vector(7 downto 0)	Valore del pixel da salvare in memoria
State_type	Enumeration	Stati della Macchina a Stati Finiti
curr_state	State_type	Stato corrente della Macchina a Stati Finiti
next_state	State_type	Stato prossimo della Macchina a Stati Finiti

2.3 Macchina a Stati Finiti

La Macchina a Stati Finiti è costituita dai seguenti stati:

- RST: Stato di in cui vengono inizializzati i segnali.
- S0: Stato in cui vengono abilitati i segnali per la lettura della memoria e che si pone in attesa del segnale i_start.
- S1: Stato in cui viene letto il numero di colonne.
- S2: Stato in cui viene letto il numero di righe.
- S3: Stato in cui viene assegnato al segnale counter il valore di minimo tra n_col e n_row in modo da effettuare un numero minore di somme per il calcolo di size.
- S4: Stato in cui viene calcolata la dimensione dell'immagine (size) sommando un numero counter di volte n_row se counter = n_col oppure un numero counter di volte n_col se counter = n_row.
- S5, S6, S7: Stati attraverso i quali viene eseguito il ciclo che effettua la lettura di tutti i pixel dell'immagine al fine di trovare il valore d'intensità

minimo ed il valore di intensità massimo.
In particolare:

  - `S5`: Stato che verifica che siano stati letti tutti i pixel.
  - `S6`: Stato in cui viene letto il pixel i-esimo (`current_pixel_value`).
  - `S7`: Stato in cui viene comparato il pixel i-esimo con i valori di pixel massimo e di pixel minimo ed eventualmente avviene l'aggiornamento di quest'ultimi.

- `S8`: Stato in cui viene calcolato `DELTA_VALUE`.
- `S9`: Stato in cui viene calcolato `DELTA_VALUE + 1`.
- `S10`: Stato in cui viene calcolato `FLOOR(LOG2(DELTA_VALUE + 1))`.
- `S11`: Stato in cui viene calcolato `shift_level`.
- `S12`, `S13`, `S14`, `S15`, `S16`, `S17`: Stati corrispondenti al ciclo che calcola il nuovo valore del pixel i-esimo e salva in memoria il risultato.
  In particolare:

  - `S12`: Stato che verifica che siano stati letti tutti i pixel.
  - `S13`: Stato in cui viene letto il pixel i-esimo (`current_pixel_value`).
  - `S14`: Stato in cui viene calcolata la differenza (`diff_pixel_value`) tra il valore d'intensità del pixel i-esimo e il valore d'intensità minimo dell'immagine.
  - `S15`: Stato in cui viene effettuato lo shift del pixel i-esimo.
  - `S16`: Stato in cui viene effettuato il confronto tra il nuovo valore riguardante il pixel i-esimo e il valore `255` al fine di scegliere il minimo tra i due.
  - `S17`: Stato in cui viene scritto il risultato riguardante il pixel i-esimo in memoria.

- `S18`: Stato in cui viene portato alto il segnale `o_done` che notifica la fine dell'elaborazione.
- `S19`: Stato che mantiene alto il segnale `o_done` fintanto che `i_start=1`, diversamente la macchina viene riportata allo stato di RST.

2.3.1 Minimalità della FSM

E' possibile ottenere una soluzione più compatta in quanto la FSM non presenta il numero minimo di stati. Al fine di mantenere chiaro il codice, abbiamo ritenuto la nostra implementazione un giusto compromesso.

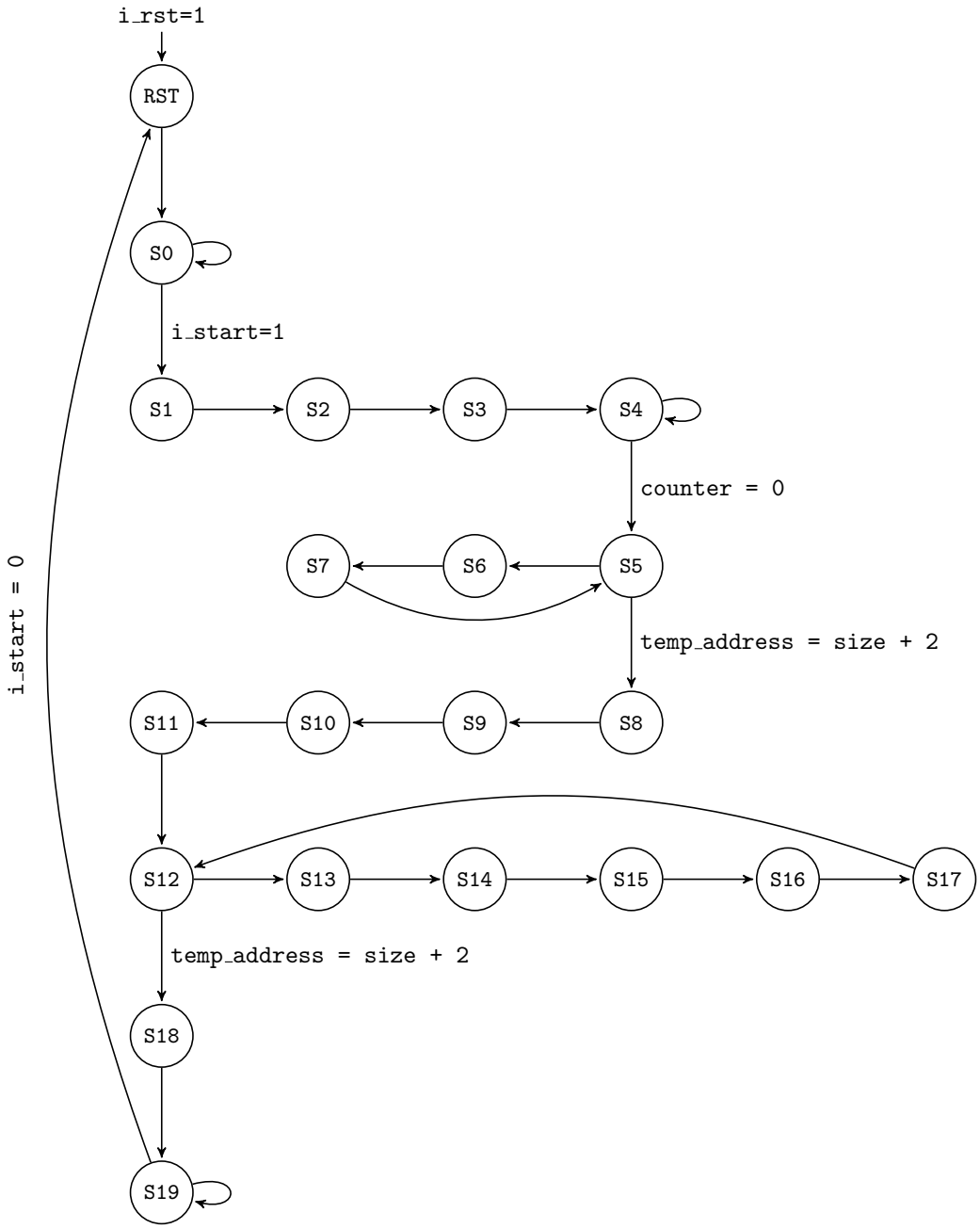

Figura 1: Diagramma della Macchina a Stati Finiti

3 Risultati

3.1 Report di Sintesi

Di seguito viene riportato parte del report sull'utilizzo delle risorse dell'FPGA generato nel processo di sintesi:

3.1.1 Utilization

A seguito del processo di sintesi sono risultati essere utilizzati

- Flip Flop: 204 (lo 0.07% del totale)
- LUT: 156 (lo 0.13% del totale)

Non è stato inferto alcun LATCH.

Resource	Utilization	Available	Utilization %
LUT	156	134600	0.12
FF	204	269200	0.08
IO	38	285	13.33

Figura 2: report_utilization

3.2 Risultati dei Test Bench

Per verificare il corretto funzionamento del componente abbiamo svolto un numero elevato di test: abbiamo generato questi ultimi sia in maniera casuale cercando di portare la macchina a compiere più cammini possibili differenti, sia in modo da testare alcuni corner case, utili per verificare che il componente non presentasse errori in situazioni limite. Vengono di seguito riportati i Test Bench considerati più rilevanti:

3.2.1 Test Bench Fornito

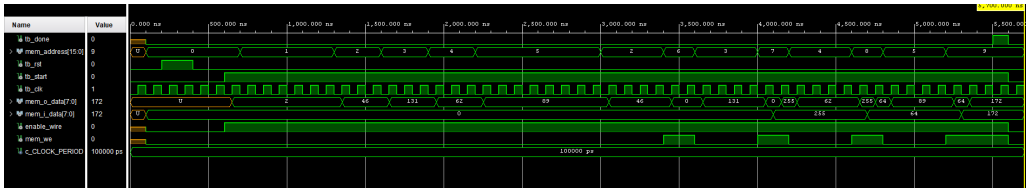

Figura 3: Test Bench Fornito

3.2.2 Reset Asincrono

In questo test viene verificato il caso in cui durante la computazione venga portato alto il segnale di reset. Il componente ritorna dunque allo stato RST reinizializzando i segnali e rieseguendo la computazione.

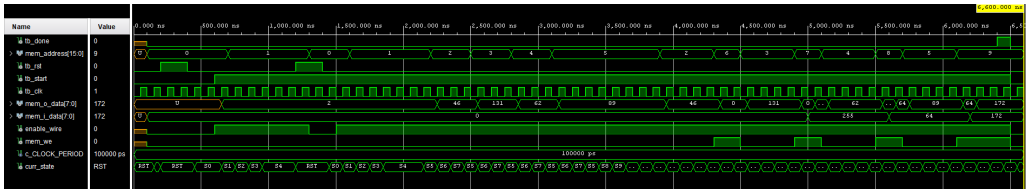

Figura 4: Reset Asincrono

3.2.3 Reset Asincrono e Cambio Immagine

In questo test viene verificato il corretto comportamento del modulo nel caso in cui, dopo un segnale di reset asincrono, venga cambiata l'immagine da modificare.

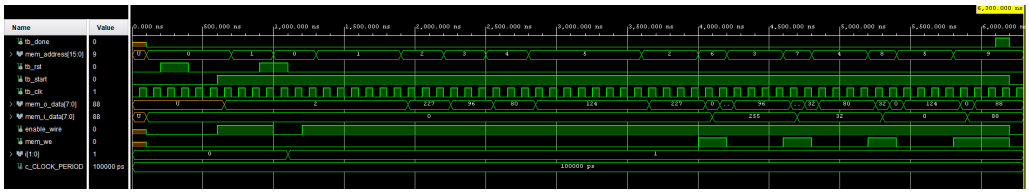

Figura 5: Reset Asincrono e Cambio Immagine

3.2.4 0 Pixel

Test che mostra come il componente gestisca correttamente il caso in cui il numero di colonne sia nullo.

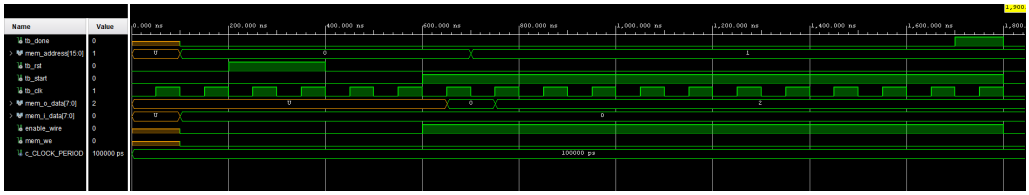

Figura 6: 0 pixel

3.2.5 Persistenza del segnale `i_start=1` alla fine della computazione

In questo test si verifica la coerenza del modulo rispetto la specifica, rilevando la persistenza del segnale `o_done` al livello logico alto fintanto che il segnale `i_start` non viene portato basso alla fine della computazione.

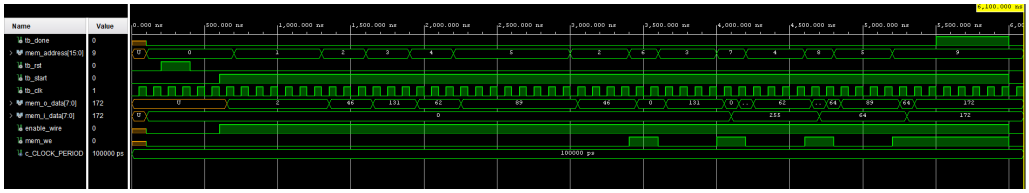

Figura 7: Persistenza `i_start=1`

4 Conclusioni

Il componente è risultato essere sintetizzabile ed in grado di superare correttamente tutti i test specificati nelle 3 simulazioni: *Behavioral*, *Post-Synthesis Functional* e *Post-Synthesis Timing*. Ci riteniamo dunque soddisfatti del lavoro compiuto che ci ha permesso di realizzare un componente funzionante che rispetti le specifiche forniteci.