PROVA FINALE DI RETI LOGICHE

Filippo Caliò (907675) - Cod.Persona: 10628126
Giovanni Caleffi (907455) - Cod.Persona: 10665233

Prof. William Fornaciari - AA: 2020/2021

Contents

1 Introduzione

1.1 Scopo del progetto

Lo scopo del progetto è la realizzazione di un componente hardware, scritto in VHDL. Esso riceve in ingresso un'immagine in scala di grigi a 256 livelli e, dopo aver applicato un algoritmo di equalizzazione a ciascun pixel, scrive in output l'immagine equalizzata.
Di seguito, un esempio di un'immagine 2x2 equalizzata (l'indirizzo dei dati in memoria verrà spiegato nel paragrafo 1.4).

0	1	2	3	4	5	6	7	8	9
2	**2**	46	131	62	89	0	255	64	172

1.2 Specifiche generali

L'algoritmo usato per l'equalizzazione delle immagini è una versione semplificata rispetto all'algoritmo standard. Esso può essere applicato solo a immagini in scala di grigi e per trasformare ogni pixel dell'immagine, esegue le seguenti operazioni:

$$\text{DELTA\_VALUE} = \text{MAX\_PIXEL\_VALUE} - \text{MIN\_PIXEL\_VALUE}$$
$$\text{SHIFT\_LEVEL} = (8 - \text{FLOOR}(\text{LOG2}(\text{DELTA\_VALUE} + 1)))$$
$$\text{TEMP\_PIXEL} = (\text{CURRENT\_PIXEL\_VALUE - MIN\_PIXEL\_VALUE}) << \text{SHIFT\_LEVEL}$$
$$\text{NEW\_PIXEL\_VALUE} = \text{MIN}(255\ ,\ \text{TEMP\_PIXEL})$$

MAX_PIXEL_VALUE e MIN_PIXEL_VALUE rappresentano rispettivamente il massimo e il minimo valore dei pixel dell'immagine, CURRENT_PIXEL_VALUE rappresenta il valore del pixel da trasformare e NEW_PIXEL_VALUE rappresenta il valore del nuovo pixel in output.

Il componente hardware è inoltre progettato per poter codificare più immagini, una dopo l'altra. Prima di codificare l'immagine successiva, però, l'algoritmo di equalizzazione deve essere stato applicato prima a tutti i pixel dell'immagine precedente.

1.3 Interfaccia del componente

L'interfaccia del componente, così come presentata nelle specifiche, è la seguente:

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

1.4 Dati e descrizione memoria

Le dimensioni dell'immagine, ciascuna di dimensione di 8 bit, sono memorizzati in una memoria con indirizzamento al Byte:

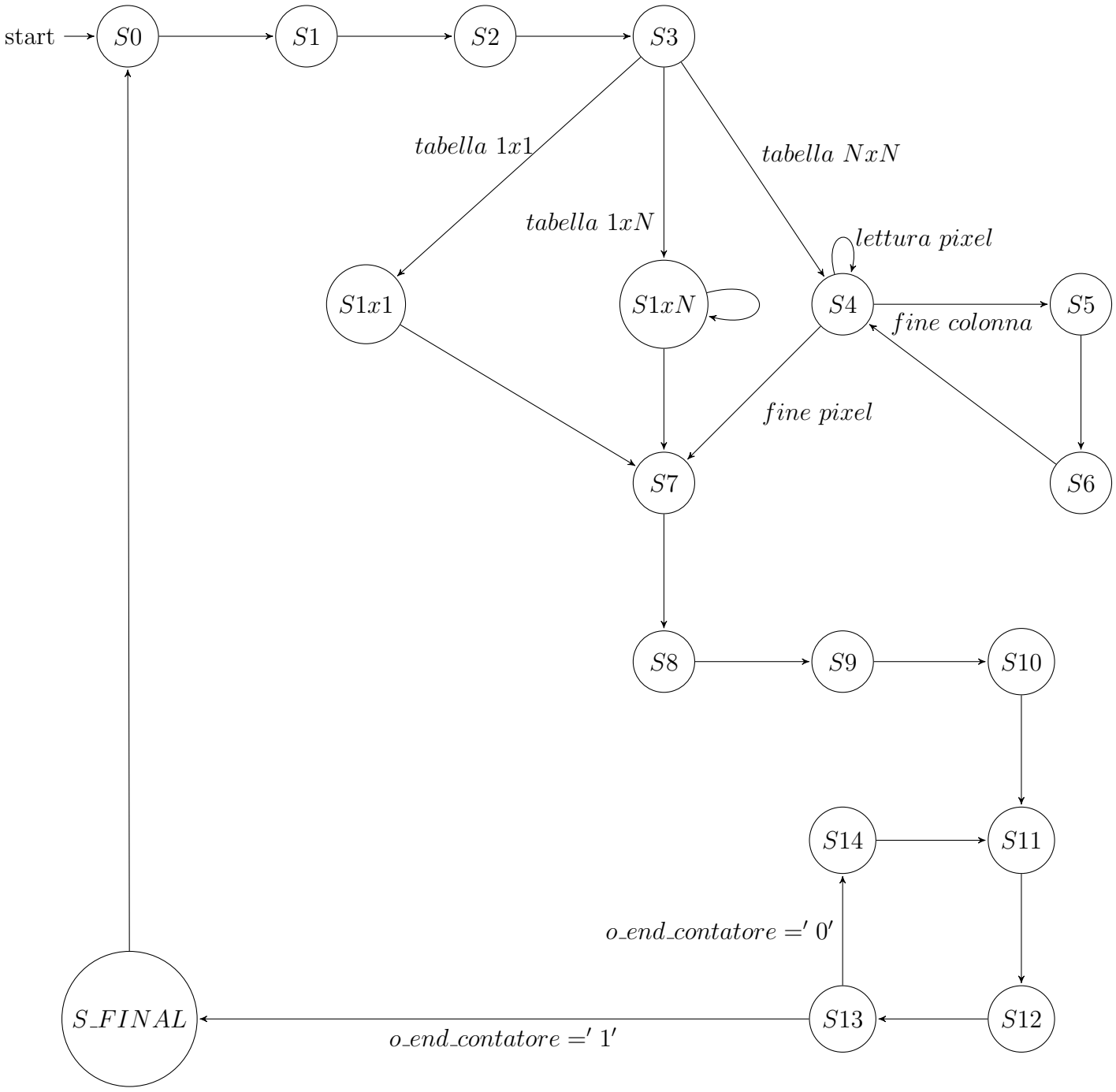
- Nell'indirizzo 0 viene salvato il numero di colonne (N-COL) dell'immagine.
- Nell'indirizzo 1 viene salvato il numero di righe (N-RIG) dell'immagine.
- A partire dall'indirizzo 2 vengono memorizzati i pixel dell'immagine, ciascuno di 8 bit.
- A partire dall'indirizzo 2+(N-COL*N-RIG) vengono memorizzati i pixel dell'immagine equalizzata.

N_COLONNE	Indirizzo 0
N_RIGHE	Indirizzo 1
PIXEL_1	Indirizzo 2
...	
PIXEL_N	
NEW_PIXEL_1	Indirizzo 2+(N_COL*N_RIGHE)
...	...
NEW_PIXEL_N	Indirizzo 1+2*(N_COL*N_RIGHE)

La dimensione massima dell'immagine è 128x128 pixel.

2 Design

La macchina è composta da 18 stati. Qui di seguito è fornita una descrizione dei vari processi.


start
S0
S1
S2
S3
tabella 1x1
tabella 1xN
tabella NxN
lettura pixel
S1x1
S1xN
S4
S5
fine colonna
fine pixel
S7
S6
S8
S9
S10
S14
S11
o_end_contatore =' 0'
S_FINAL
S13
S12
o_end_contatore =' 1'

2.1 Gestione dell'o_address, dell'enable, dell'o_done e del caricamento di o_data:

Specificare che l'oaddress è uguale a mux definitivo che se uguale a 0 prende il valore di mux roaddr e se uguale a 1 prende il valore di new o roAddr, usando il segnale di mux definitivo sel.

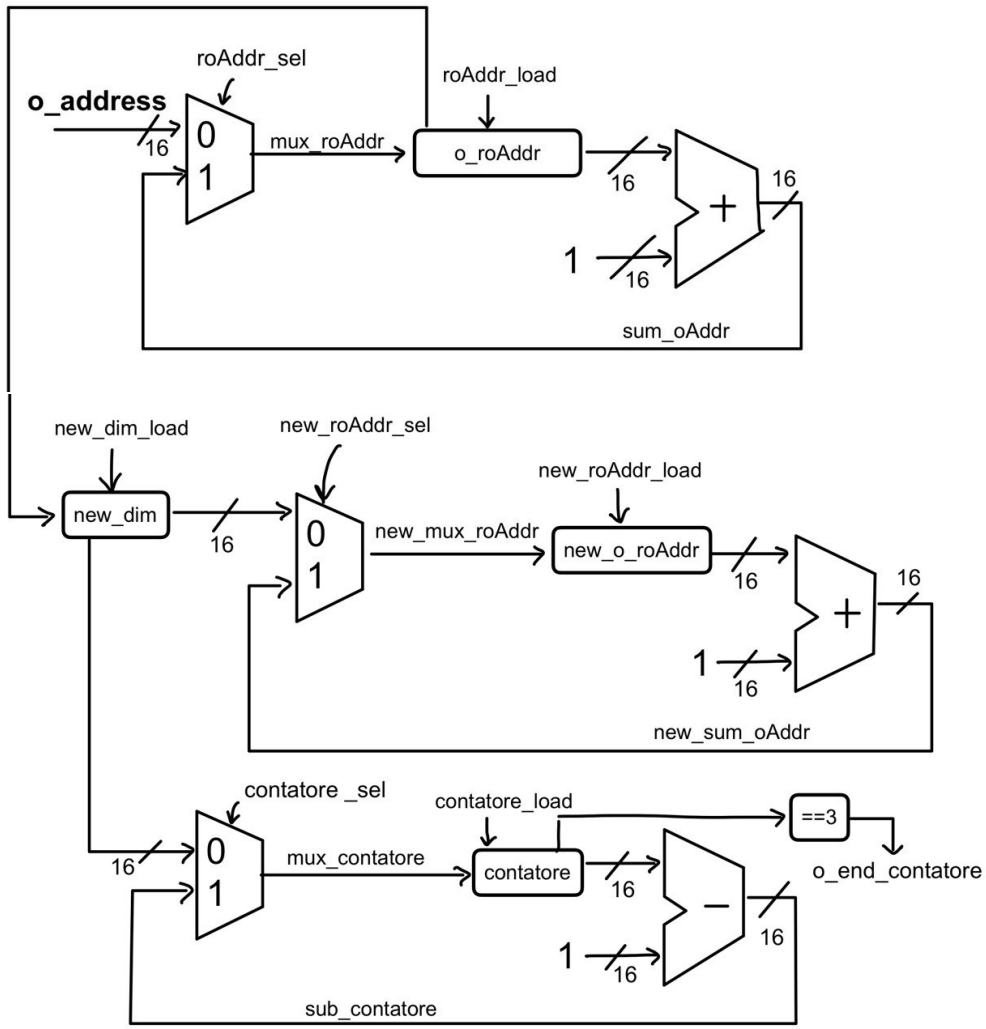


Figure 1: Gestione o_address

2.2 Lettura dei pixel

Processo per la gestione del ciclo dedicato alla lettura di tutti i pixel tramite l'uso del numero di righe e colonne:

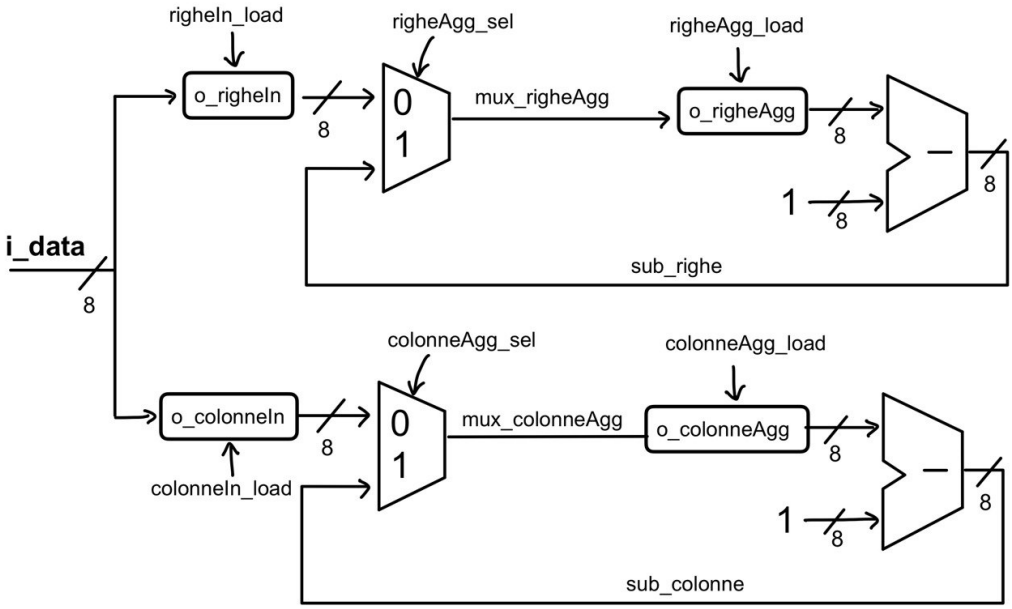


Figure 2: Gestione o_address

2.3 Calcolo pixel massimo e minimo e applicazione algoritmo per trovare valore nuovo pixel

Processo dedicato alla determinazione del pixel con valore massimo e minimo, del delta_value e dello shift_level e del nuovo valore del pixel

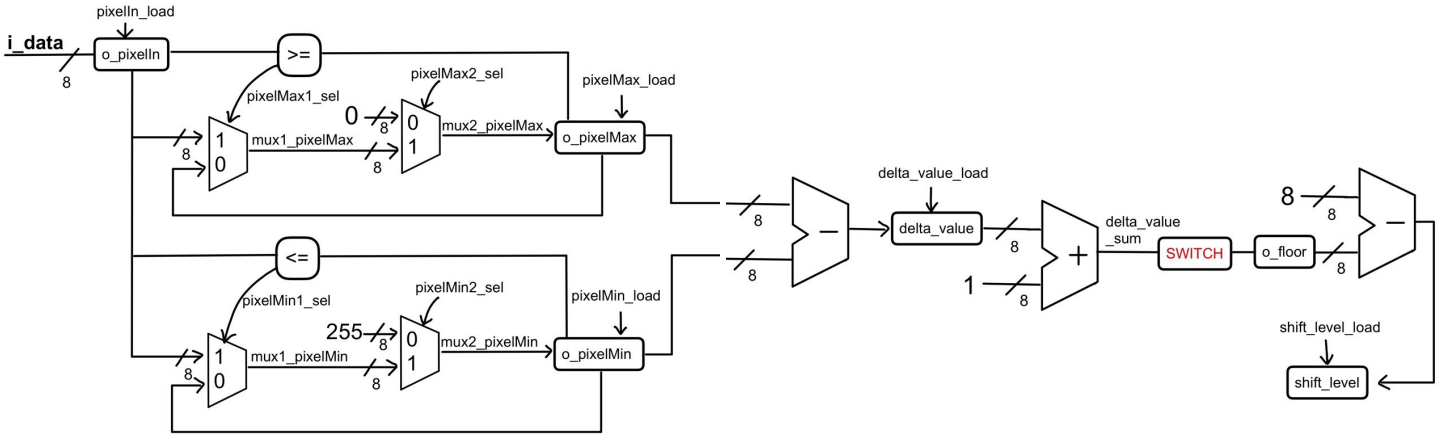


Figure 3: Gestione o_address

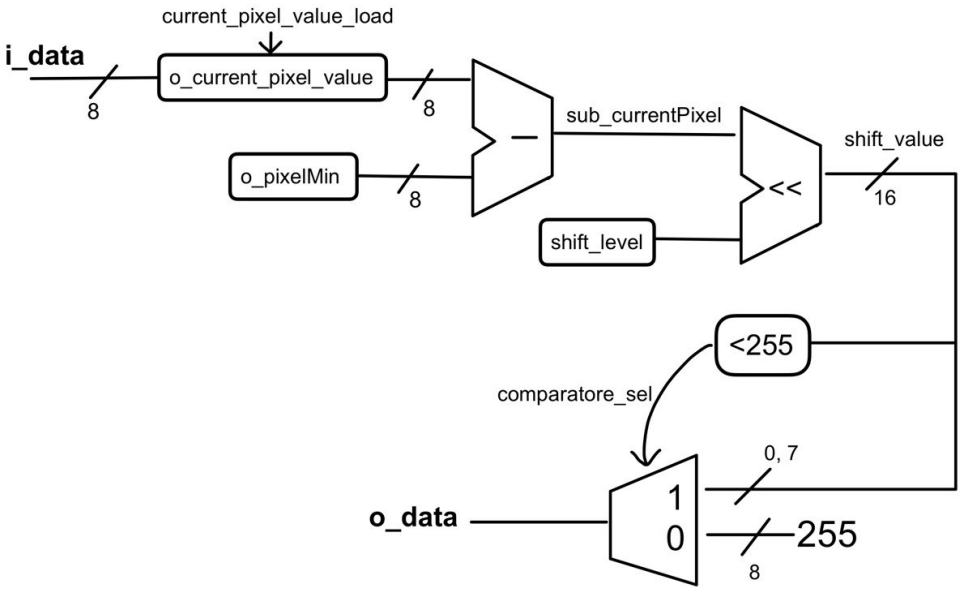


Figure 4: Gestione o_address

3 Risultati dei test

4 Conclusioni

4.1 Risultati della sintesi

4.2 Ottimizzazioni

Nel progettare il componente hardware, abbiamo prestato particolare attenzione nel rimuovere tutti i latch presenti.