Politecnico di Milano
Dipartimento di Elettronica, Informazione e Bioingegneria

Progetto di Reti Logiche 2020/21

Prof. Palermo Gianluca

Shalby Hazem Hesham Yousef (Codice Persona: 10596243, Matricola: 910871)
Perego Niccolò (Codice Persona: 10628782, Matricola: 895468)

1 Aprile 2021

POLITECNICO
MILANO 1863

1 Requisiti di progetto

1.1 Descrizione del problema

Si vuole realizzare un componente in grado di svolgere una versione semplificata del processo di equalizzazione dell'istogramma di un'immagine, ossia di ricalibrare il contrasto di quest'ultima, effettuando una ridistribuzione dei valori di intensità pixel per pixel.

Le immagini di cui è richiesta la manipolazione sono definite in scala di grigi a 256 livelli e hanno una dimensione massima di 128x128 pixel.

1.2 Interfaccia del componente

Il componente realizzato ha un'interfaccia così definita in liguaggio VHDL:

```
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, predisponendola alla ricezione del segnale di START. Può essere anche asincrono;
- `i_start` è il segnale di START generato dal Test Bench;
- `i_data` è il segnale (vettore) che arriva dalla memoria in seguito a 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 mandare alla memoria per poter comunicare (sia in lettura che in scrittura);
- `o_we` è il segnale di WRITE ENABLE da mandare alla memoria per comunicare quale operazione si vuole svolgere su di essa. Può assumere valori 0 e 1, rispettivamente per lettura e scrittura;
- `o_data` è il segnale (vettore) di uscita dal componente verso la memoria.

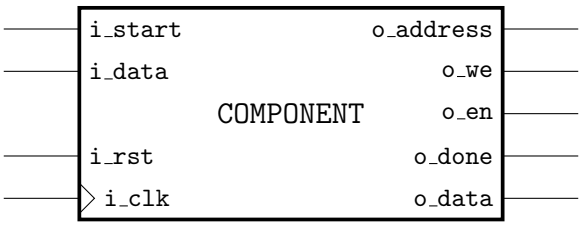


Figura 1: Schema del componente realizzato.

1.3 Descrizione della memoria e dell'interazione con il componente

Il modulo implementato dovrà dialogare in lettura e scrittura con una RAM, indirizzata al byte.

In particolare, l'algoritmo di equalizzazione sarà applicato a immagini pre-salvate in memoria, la cui grandezza effettiva (in pixel) sarà specificata dal prodotto del contenuto tra le celle a indirizzo `0` e `1` della RAM, contenenti rispettivamente il numero di colonne `n_col` e di righe `n_row` dell'immagine, entrambi di dimensione 8 bit.
Nei byte successivi, dall'indirizzo `2` all'indirizzo `n_col` $\cdot$ `n_row` $+$ `1`, sarà contenuta, pixel per pixel, sequenzialmente e in modo contiguo, l'immagine di cui è richiesta la trasformazione.
Infine, l'immagine ottenuta dal processo di equalizzazione svolto dal componente verrà salvata in memoria dall'indirizzo `n_col` $\cdot$ `n_row` $+$ `2` all'indirizzo `2` $\cdot$ `n_col` $\cdot$ `n_row` $+$ `1`.

Tabella 1: Schema generale del contenuto della memoria dopo l'elaborazione.

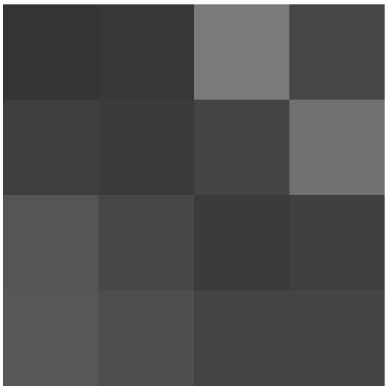
`n_col`	addr. 0
`n_row`	addr. 1
Primo pixel immagine da elaborare	addr. 2
⋮	⋮
Ultimo pixel immagine da elaborare	addr. `n_col` $\cdot$ `n_row` $+$ 1
Primo pixel immagine elaborata	addr. `n_col` $\cdot$ `n_row` $+$ 2
⋮	⋮
Ultimo pixel immagine elaborata	addr. 2 $\cdot$ `n_col` $\cdot$ `n_row` $+$ 1

1.4 Esempio di funzionamento

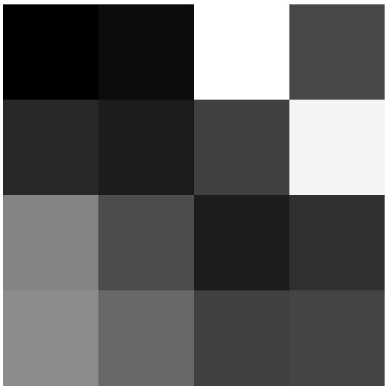
Si riporta in seguito un'esempio di elaborazione compiuta su un'immagine di test.

addr.	data	addr.	data
0	4	17	69
1	4	18	0
2	52	19	12
3	62	20	255
4	85	21	72
5	87	22	40
6	55	23	28
7	59	24	64
8	71	25	244
9	78	26	132
10	122	27	76
11	68	28	28
12	59	29	48
13	68	30	140
14	70	31	104
15	113	32	64
16	64	33	68

(a) Contenuto della memoria



(b) Immagine sorgente



(c) Immagine equalizzata

È evidente come l'immagine (c) presenti un contrasto maggiore rispetto alla (b). Questo è dato dall'ampliamento del range di valori assunti dai pixel dell'immagine, come evidenziato dalla tabella di memoria:

- Immagine sorgente, addr. 2 - 17: i pixel assumono valori da 52 a 122;
- Immagine equalizzata, addr. 18 - 33: i pixel assumono valori da 0 a 255.

Non avendo ancora affrontato appieno il processo svolto dal componente, si ritiene opportuno non approfondire in questa sezione l'insieme di passaggi che permettono la trasformazione dell'immagine riportata nell'esempio, che verranno ripresi in seguito.

Si rende disponibile un TestBench che replica l'elaborazione dell'immagine (b) a questo link.

2 Design del componente

Si è scelto di descrivere un modulo single-process tramite architettura behavioral (comportamentale) in linguaggio VHDL. Questo ha determinato la necessità di sviluppare un algoritmo adeguato allo svolgimento dell'operazione richiesta al componente, che può essere schematizzato secondo i seguenti passaggi chiave: