Progetto di Reti Logiche

Nome: **Bresciani Matteo**
Corso: **Reti Logiche**
Codice Persona: **10527150**
Matricola: **847342**

Descrizione funzionalità del componente

Il componente descritto con la seguente interfaccia

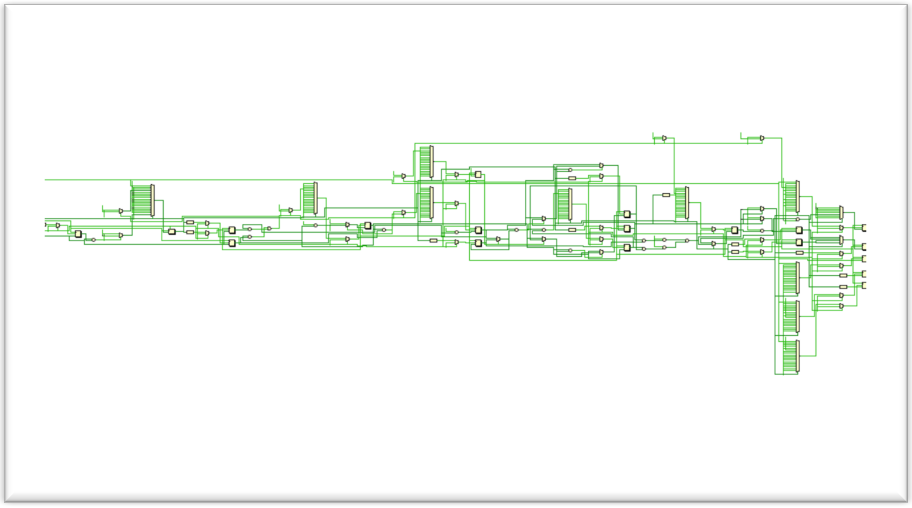
```
entity project_reti_logiche is
     port (

       i_clk: in  std_logic;
      i_start: in  std_logic;
      i_rst: 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;
```

ha la funzione, data un'immagine in scala di grigi formata da un byte (con valori compresi tra 0 e 255) , di calcolare l'area del rettangolo minimo che circoscrive i bit con valore maggiore o uguale ad un valore soglia.

Il componente legge da una memoria istanziata dal testbench[1] prima il numero di *colonne*, di *righe* e il *valore soglia*; successivamente acquisisce ogni valore appartenente all'immagine (le specifiche complete sono alla seguente pagina :

https://home.deib.polimi.it/zoni/material/Prova_finale_di_Reti_Logiche20180313_documento_di_specifica.pdf)



Design del componente sintetizzato

Realizzazione

Il componente, durante l'acquisizione, svolge tale funzione cercando gli indici più piccoli e più grandi delle colonne e delle righe dove il pixel è maggiore della soglia. Terminata la lettura di ogni pixel, il componente li sottrae e va moltiplicare *base* e *altezza* ottenuti.

```
a = i_ultimo - i_primo + 1
b = j_ultimo - j_primo + 1
result= a * b
```

Pseudocodice dell'operazione finale svolta dal componente

Scelta adattate

Si è scelto di utilizzare una **macchina a stati di Moore**, suddividendo quindi le varie operazioni in degli stati.

Stati Utilizzati

Il componente dispone di 17 stati, i quali si suddividono le varie operazioni di *acquisizione*, *confronto con il valore soglia* o la *scrittura (o di buffering):*

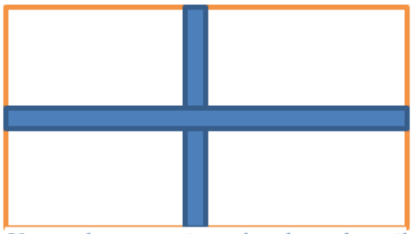
- **S0**: inizializza la macchina pronta a ricevere il primo byte con *o_en=1;*
- **S00**: stato di buffer;
- **S1**: acquisisce il numero di colonne;
- **S011**: stato di buffer;
- **S2**: acquisisce il numero di righe;
- **S22**: stato di buffer;
- **S3**: acquisisce il valore soglia;
- **S33**: stato di buffer;
- **S4**: acquisisce il pixel corrente;
- **S44**: stato di buffer;

- **S5-S6**: gestisce, facendo confronti, gli estremi dell'immagine;
- **S7**: calcola il risultato e lo scompone in due *signal* da 8 bit ciascuno;
- **S8**: scrive nella prima cella di memoria i bit meno significativi;
- **S88**: stato di buffer;
- **S9**: scrive nella seconda cella di memoria i bit meno significativi portando a 1 il segnale *done*;
- **S99**: stato di buffer;
- **S_done**: stato finale in cui *o_done=0;*

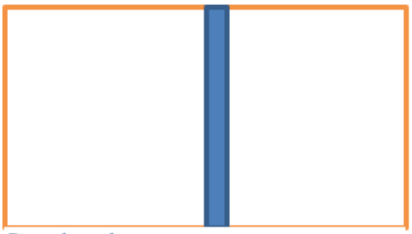
Test

Il componente esegue con successo tutti i testbench provati. Il componente è stato testato con 4 testbench (con e senza ritardo) che si trovano alla pagina del tutor Zoni (https://home.deib.polimi.it/zoni/tutor_rl_2017_2018.html). Inoltre sono stati provati altri casi test con immagini particolari come le seguenti:

Singola riga



Una colonna e riga che deve dare il prodotto della base e dell'altezza



Singola colonna

Base e/o altezza in memoria nulle

Primo e ultimo byte maggiori di soglia

Risultati dei test

Il componente è stato simulato in:

- **Pre-Sintesi** (*Behavioral*); √
- **Post-Sintesi** (*Functional* e *Timing*); √

  →**Post-Implementazione** (solo *Functional*) √

Si mostrano inoltre i tempi di computazione raccolti durante la simulazione dei 4 Testbench forniti con un ciclo di clock pari a:

15 ns:

Simulazione (functional +timing)	**Tempo**
TestBench1	7,927 us
TestBench2	10,447 us
TestBench3	8,587 us
TestBench4	8,047 us

10 ns:

Simulazione (functional +timing)	**Tempo**
TestBench1	5,315 us
TestBench2	6,995 us
TestBench3	5,775 us
TestBench4	5,395 us

5 ns:

Simulazione (functional +timing)	**Tempo**
TestBench1	2,707 us
TestBench2	3,547 us
TestBench3	2,927 us
TestBench4	2,747 us

1 ns:

Simulazione (functional)	**Tempo**
TestBench1	621 ns
TestBench2	782 ns
TestBench3	665 ns
TestBench4	629 ns