Progetto finale di reti logiche

Si vuole implementare un componente HW descritto in VHDL che, data un'immagine in scala di grigi in un formato descritto successivamente, calcoli l'area del rettangolo minimo che circoscrive totalmente una figura di interesse presente nell'immagine stessa. Il termine *circoscrive totalmente* indica il fatto che il rettangolo deve essere il più piccolo tale che tutti i pixel facenti parte della figura di interesse siano interni o appartenenti al perimetro del rettangolo.

Formato Immagine

Il formato usato per codificare l'immagine è composto da 2 parti. La prima parte è un **header** che descrive la struttura, la seconda parte invece descrive il **contenuto** dell'immagine.

L'**header** è composto da 3 campi, ognuno della dimensione di un byte. I campi sono i seguenti:

N_COLONNE
N_RIGHE
SOGLIA

I valori di questi tre campi rappresentano:

- N_COLONNE - larghezza dell'immagine (in pixel)
- N_RIGHE - altezza dell'immagine (in pixel)
- SOGLIA - valore di soglia per la figura di interesse

Il **contenuto** dell'immagine è codificato come una matrice (di dimensione N_RIGHE x N_COLONNE) in cui ogni elemento della matrice rappresenta il valore in scala di grigi di un singolo pixel. Ogni pixel occupa un byte e può avere valori tra 0 e 255. Se il valore di un pixel è maggiore o uguale al valore di SOGLIA, tale pixel fa parte della figura di interesse, altrimenti è considerato di sfondo.

N.B.: tutti i valori coinvolti, sia nell'header che nel contenuto dell'immagine, **sono compresi tra 0 e 255 e occupano un byte**.

ESEMPIO:

La seguente sequenza di numeri mostra un esempio del formato in plain text di una immagine 24 x 7 con valore di soglia per la figura di interesse pari a 2.

```
24 7
2
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  3  3  3  3  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15  0  0  0  0
0  3  0  0  0  0  0  7  0  0  0  0  0 31  0  0  0  0  0 25  0  0  0  0
0  3  3  3  0  0  0  7  7  7  0  0  0 31 31 11  0  0  0 25  0  0  0  0
0  3  0  0  0  0  0  7  0  0  0  0  0 31  0  0  0  0  0 25  0  0  0  0
0  3  0  0  0  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15 15 15 15  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

In memoria (la descrizione della memoria è al termine di questo documento) questa immagine è però memorizzata in maniera sequenziale dove ogni valore (dimensioni, soglia e pixel) è memorizzato in un byte separato. Il contenuto dell'immagine è memorizzato per righe. Gli spazi bianchi e gli a capo presenti nell'esempio ovviamente non sono memorizzati poiché il formato è puramente binario.

Indirizzo Memoria	Valore	Commento
N	**24**	Dimensione X dell'immagine (N_COLONNE)
N+1	**7**	Dimensione Y dell'immagine (N_RIGHE)
N+2	**2**	Valore di SOGLIA per la figura di interesse
N+3	**0**	valore del pixel <0,0> (riga 0, colonna 0)
N+4	**0**	Valore del pixel <0,1> (riga 0, colonna 1)
[...]	**[...]**	

Nota: il VALORE (colonna in **grassetto***) è ovviamente l'unica parte memorizzata ed è rappresentata in memoria come cifra binaria, cioè 24=00011000 7=00000111, 2=00000010, 0=00000000 etc...*

Prova d’esame

Compito dello studente è quello di descrivere in VHDL e sintetizzare il componente HW che implementa la precedente specifica, interfacciandosi con una memoria dove è memorizzata l’immagine e nel quale andrà riscritto indietro il valore calcolato per l’area. Allo studente verrà fornito un Test Bench (che include la memoria) di esempio per validare il funzionamento corretto del modulo implementato.

- Strumento di sintesi da usare è XILINX VIVADO WEBPACK e la target FPGA target può essere qualunque. Per uniformità si inserisca xc7a200tfbg484-1 durante la sintesi
- Un componente descritto e simulabile correttamente in pre-sintesi viene valutato fino ad un massimo di 24.
- Un componente sintetizzabile e correttamente simulabile in post-sintesi ottiene una valutazione superiore a 24.
- Lo studente deve allegare al progetto anche della documentazione che descriva le scelte progettuali fatte e che dimostri anche i test effettuati per validare il componente.
- La consegna del progetto può essere fatta 1 volta sola nell’arco dell’anno accademico.

Note Ulteriori sulla specifica

1) L’immagine nel formato opportuno è posizionata in memoria a partire dal byte 2;
2) Il valore dell’area del rettangolo che circoscrive la figura deve essere posizionato in memoria nei byte 1 e 0. Nel byte 1 ci dovrà essere la parte più significativa del valore di area mentre in 0 la parte meno significativa;
3) Il modulo partirà nella elaborazione quando un segnale START in ingresso verrà portato a 1 per un ciclo di clock; Al termine della computazione (e una volta scritto il risultato in memoria), il modulo deve alzare (portare a 1) un segnale DONE che notifica la fine per 1 ciclo di clock. Un nuovo segnale start non può essere dato fin tanto che DONE non è stato riportato a zero.
4) Per chiarire ulteriormente la specifica usiamo l’esempio precedente che mostra il formato dell’immagine e determiniamo l’area del rettangolo al variare della soglia:
    a) soglia = 0 => area del rettangolo che circoscrive la figura è 168 pixels;
    b) soglia <4 => area del rettangolo che circoscrive la figura è 110 pixels;
    c) 3 < soglia < 8 => area del rettangolo che circoscrive la figura è 80 pixels;
    d) 7 < soglia < 12 => area del rettangolo che circoscrive la figura è 50 pixels;
    e) 11 < soglia < 16 => area del rettangolo che circoscrive la figura è 50 pixels;
    f) 15 < soglia < 26 => area del rettangolo che circoscrive la figura è 21 pixels;
    g) 25 < soglia < 32 => area del rettangolo che circoscrive la figura è 6 pixels;
    h) soglia > 31 => area del rettangolo che circoscrive la figura è 0 pixels;

Interfaccia del Componente

Il componente da descrivere deve avere 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;
```

In particolare:

- i_clk è il segnale di CLOCK in ingresso generato dal TestBench;
- i_start è il segnale di START generato dal Test Bench;
- i_rst è il segnale di RESET che inzializza la macchina pronta per per ricevere il primo segnale di START;
- 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 comunicarci (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.

APPENDIX: Descrizione Memoria
!!!La memoria è già instanziata all'interno del Test Bench e non va sintetizzata!!!

La memoria e il suo protocollo può essere estratto dalla seguente descrizione VHDL che fa parte del test bench e che è derivata dalla User guide di VIVADO disponibile al seguente link:
https://www.xilinx.com/support/documentation/sw_manuals/xilinx2017_3/ug901-vivado-synthesis.pdf

```
-- Single-Port Block RAM Write-First Mode (recommended template)
--
-- File: rams_02.vhd
--
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity rams_sp_wf is
port(
  clk  : in  std_logic;
  we   : in  std_logic;
  en   : in  std_logic;
  addr : in  std_logic_vector(15 downto 0);
  di   : in  std_logic_vector(7 downto 0);
  do   : out std_logic_vector(7 downto 0)
);
end rams_sp_wf;

architecture syn of rams_sp_wf is
type ram_type is array (65535 downto 0) of std_logic_vector(7 downto 0);
signal RAM : ram_type;
begin
  process(clk)
    begin
    if clk'event and clk = '1' then
      if en = '1' then
        if we = '1' then
          RAM(conv_integer(addr)) <= di;
          do                      <= di;
        else
          do <= RAM(conv_integer(addr));
        end if;
      end if;
    end if;
  end process;
end syn;
```