Example 37

Sprites in Block ROM

In Example 36 we made a sprite by storing the bit map of three initials in a VHDL ROM. To make a larger sprite we could use the Core Generator to store the sprite image in either distributed or block RAM/ROM. Block ROM will allow larger images to be stored and in this example we will use a block ROM to store the image of the two loons shown in Fig. 6.11. The size of this image is 240 x 160 pixels and is available at www.lbebooks.com as the image file *loons240x160.jpg*.

In Example 30 we showed how to create a block ROM using the Core Generator where the initial data is stored in a *.coe* file. We can convert the JPEG image in the file *loons240x160.jpg* to a corresponding *.coe* file called *loons240x160.coe* by using the Matlab function *IMG2coe8*(*imgfile*, *outfile*) shown in Listing 6.7. This function will work with other standard image formats other than JPEG such as *bmp*, *gif*, and *tif*. Note that the *.coe* file produced by this function contains an 8-bit byte for each image pixel with the format

color byte = [R2,R1,R0,G2,G1,G0,B1,B0] (6-3)

The original image read into the Matlab function in Listing 6.7 will contain 8-bits of red, 8-bits of green, and 8-bits of blue. The 8-bit color byte stored in the *.coe* file will contain only the upper 3 bits of red, the upper 3 bits of green, and the upper 2 bits of blue. We need to do this because as we have seen the Nexys-2 board supports only 8-bit VGA colors. The resulting 8-bit color image is called *img2* in Listing 6.7 and will be of reduced quality from the original image as can be seen in Fig. 6.12.

Figure 6.11 Loon photo by Edie Haskell

Figure 6.12
The image *img2* produced by Listing 6.7

Listing 6.7 IMG2coe8.m

```
function img2 = IMG2coe8(imgfile, outfile)
% Create .coe file from .jpg image
% .coe file contains 8-bit words (bytes)
% each byte contains one 8-bit pixel
% color byte: [R2,R1,R0,G2,G1,G0,B1,B0]
% img2 = IMG2coe8(imgfile, outfile)
% img2 is 8-bit color image
% imgfile = input .jpg file
% outfile = output .coe file
% Example:
% img2 = IMG2coe8('loons240x160.jpg', 'loons240x160.coe');

img = imread(imgfile);
height = size(img, 1);
width = size(img, 2);
s = fopen(outfile,'wb');  %opens the output file
fprintf(s,'%s\n','; VGA Memory Map ');
fprintf(s,'%s\n','; .COE file with hex coefficients ');
fprintf(s,'; Height: %d, Width: %d\n\n', height, width);
fprintf(s,'%s\n','memory_initialization_radix=16;');
fprintf(s,'%s\n','memory_initialization_vector=');
cnt = 0;
img2 = img;
for r=1:height
    for c=1:width
        cnt = cnt + 1;
        R = img(r,c,1);
        G = img(r,c,2);
        B = img(r,c,3);
        Rb = dec2bin(R,8);
        Gb = dec2bin(G,8);
        Bb = dec2bin(B,8);
        img2(r,c,1) = bin2dec([Rb(1:3) '00000']);
        img2(r,c,2) = bin2dec([Gb(1:3) '00000']);
        img2(r,c,3) = bin2dec([Bb(1:2) '000000']);
        Outbyte = [ Rb(1:3) Gb(1:3) Bb(1:2) ];
        if (Outbyte(1:4) == '0000')
            fprintf(s,'0%X',bin2dec(Outbyte));
        else
            fprintf(s,'%X',bin2dec(Outbyte));
        end
        if ((c == width) && (r == height))
            fprintf(s,'%c',';');
        else
            if (mod(cnt,32) == 0)
                fprintf(s,'%c\n',',');
            else
                    fprintf(s,'%c',',');
            end
        end
    end
end
fclose(s);
```

The *.coe* file produced by Listing 6.7 will contain $240 \times 160 = 38,400$ bytes. Following the procedure described in Example 30 we can use the Core Generator to create a read only block memory component called *loons240x160*, which is 8 bits wide with a depth of 38400 as shown in Fig. 6.13. This component would be part of the top-level design shown in Fig. 6.14.

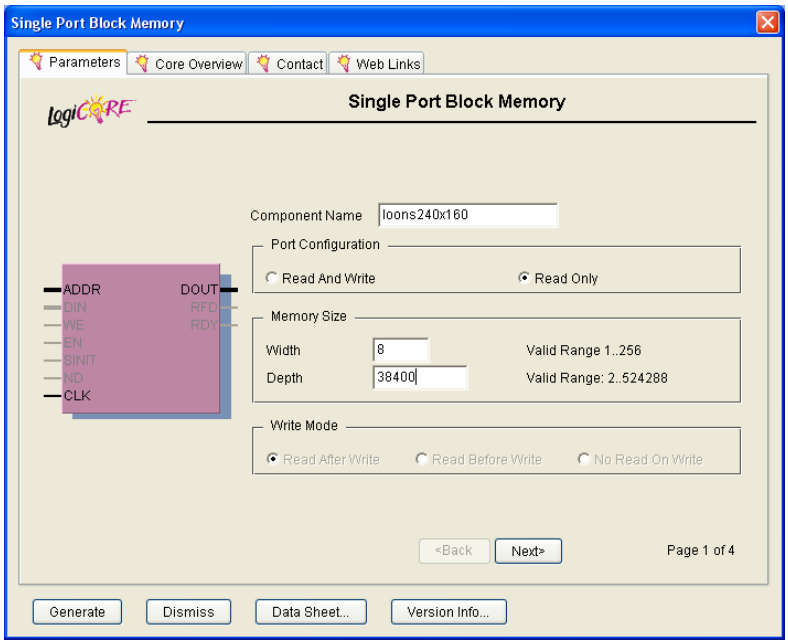

Figure 6.13 Generating the component *loons240x160*

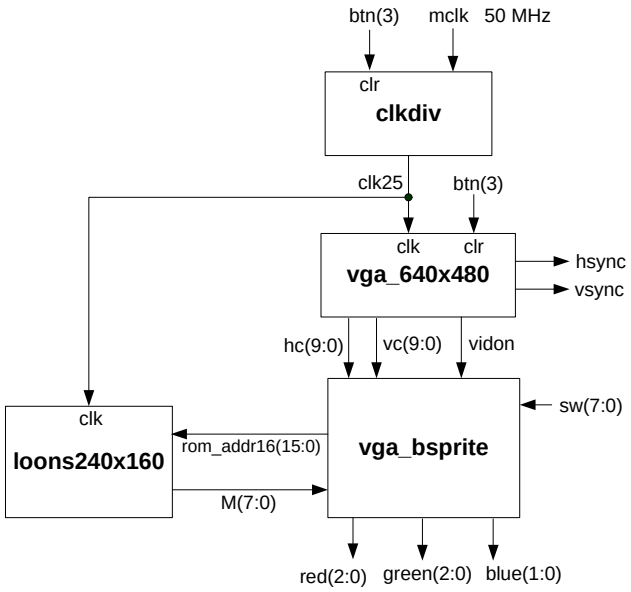

Figure 6.14 Top-level design to display the loon photo on the VGA screen

Listing 6.8 is the VHDL program for the component *vga_bsprite* shown in Fig. 6.14. It is similar to the component *vga_initials* shown in Fig. 6.9 of Example 36 and described by Listing 6.5. The main difference is that *rom_addr16*(15:0) is now calculated from the formula

$$rom\_addr16 = ypix \times 240 + xpix$$

where *xpix* and *ypix* are the local image coordinates within the loon image of the pixel whose color byte is at address *rom_addr16* in the block ROM *loons240x160*.

Listing 6.8 vga_bsprite.vhd

```
-- Example 37a: vga_bsprite
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity vga_bsprite is
    port ( vidon: in std_logic;
           hc : in std_logic_vector(9 downto 0);
           vc : in std_logic_vector(9 downto 0);
           M: in std_logic_vector(7 downto 0);
           sw: in std_logic_vector(7 downto 0);
           rom_addr16: out std_logic_vector(15 downto 0);
           red : out std_logic_vector(2 downto 0);
           green : out std_logic_vector(2 downto 0);
           blue : out std_logic_vector(1 downto 0)
    );
end vga_bsprite;

architecture vga_bsprite of vga_bsprite is
constant hbp: std_logic_vector(9 downto 0) := "0010010000";
    --Horizontal back porch = 144 (128+16)
constant vbp: std_logic_vector(9 downto 0) := "0000011111";
    --Vertical back porch = 31 (2+29)
constant w: integer := 240;
constant h: integer := 160;
signal xpix, ypix: std_logic_vector(9 downto 0);
signal rom_addr : std_logic_vector(16 downto 0);
signal C1, R1: std_logic_vector(9 downto 0);
signal spriteon, R, G, B: std_logic;

begin
    --set C1 and R1 using switches
    C1 <= '0' & SW(3 downto 0) & "00001";
    R1 <= '0' & SW(7 downto 4) & "00001";
    ypix <= vc - vbp - R1;
    xpix <= hc - hbp - C1;

    --Enable sprite video out when within the sprite region
    spriteon <= '1' when (((hc >= C1 + hbp) and (hc < C1 + hbp + w))
          and ((vc >= R1 + vbp) and (vc < R1 + vbp + h))) else '0';
```

Listing 6.8 (cont.) vga_bsprite.vhd

```
    process(xpix, ypix)
    variable  rom_addr1, rom_addr2: STD_LOGIC_VECTOR (16 downto 0);
    begin
          rom_addr1 := (ypix & "0000000") + ('0' & ypix & "000000")
            + ("00" & ypix & "00000") + ("000" & ypix & "0000");
           -- y*(128+64+32+16) = y*240
          rom_addr2 := rom_addr1 + ("00000000" & xpix);   -- y*240+x
          rom_addr16 <= rom_addr2(15 downto 0);
    end process;

    process(spriteon, vidon, M)
          variable j: integer;
    begin
          red <= "000";
          green <= "000";
          blue <= "00";
          if spriteon = '1' and vidon = '1' then
                red <= M(7 downto 5);
                green <= M(4 downto 2);
                blue <= M(1 downto 0);
          end if;
    end process;

end vga_bsprite;
```

Notice how we multiply *ypix* by 240 using a shift and add method by recognizing that

$$ypix \times 240 = ypix \times (128 + 64 + 32 + 16)$$

The *red*, *green*, and *blue* outputs are obtained from the *loon240x160* ROM output *M*(7:0) according to the color byte format shown in Eq. (6-3).

Listing 6.9 is the VHDL program for the top-level design shown in Fig. 6.14.

Listing 6.9 vga_bsprite_top.vhd

```
-- Example 37b: vga_bsprite_top
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use work.vga_components.all;

entity vga_bsprite_top is
       port(
            mclk : in STD_LOGIC;
            btn : in STD_LOGIC_VECTOR(3 downto 0);
            sw : in STD_LOGIC_VECTOR(7 downto 0);
            hsync : out STD_LOGIC;
            vsync : out STD_LOGIC;
            red : out std_logic_vector(2 downto 0);
            green : out std_logic_vector(2 downto 0);
            blue : out std_logic_vector(1 downto 0)
           );
end vga_bsprite_top;

architecture vga_bsprite_top of vga_bsprite_top is
signal clr, clk25, vidon: std_logic;
signal hc, vc: std_logic_vector(9 downto 0);
signal M: std_logic_vector(7 downto 0);
signal rom_addr16: std_logic_vector(15 downto 0);
begin

      clr <= btn(3);

U1 : clkdiv
      port map(mclk => mclk, clr => clr, clk25 => clk25);

U2 : vga_640x480
      port map(clk => clk25, clr => clr, hsync => hsync,
      vsync => vsync, hc => hc, vc => vc, vidon => vidon);

U3 : vga_bsprite
      port map(vidon => vidon, hc => hc, vc => vc, M => M, sw => sw,
            rom_addr16 => rom_addr16, red => red, green => green,
            blue => blue);

U4 : loons240x160
      port map (addr => rom_addr16, clk => clk25, dout => M);

end vga_bsprite_top;
```