Listing 2.5 (cont.) vga_flash_n2_top.vhd

```
      clr <= btn(3);
      -- Constant outputs
      FlashCE_L <= '0';       -- Enable flash
      flashRp_L <= '1';
      CE_L <= '1';            -- Disable RAM
      OE_L <= '0';
      WE_L <= '1';
      -- flash stores data little endian
      data0(15 downto 8) <= DQ(7 downto 0);
      data0(7 downto 0) <= DQ(15 downto 8);

U1 : clkdiv
      port map(mclk => mclk, clr => clr, clk40 => clk40);

U2 : vga_flash_n2
      port map(clk40 => clk40, clr => clr, vidon => vidon,
            hc => hc, vc => vc, data0 => data0,
            addr0 => A, red => red, green => green,
            blue => blue);

U3 : vga_640x480
      port map(clk => clk40, clr => clr, hsync => hsync,
            vsync => vsync, hc => hc, vc => vc,
            vidon => vidon);
end vga_flash_n2_top;
```

Example 15: Moving Sprites with Flash Background

In this example, we will have the loons shown in Fig. 2.13b swim across the water following the red arrows. When *btn*(0) is pressed, the loons will move along the bottom dashed line and then change their orientation when getting to the left side of the image. When *btn*(1) is pressed, the loons will move along the top dashed line, stop at their original position, and then change their orientation again.

The background image will be stored in the external flash memory using the method described in Example 14. The image on the left in Fig. 2.13a is the original image of a cove on Lake Winnipesaukee, New Hampshire, and the image on the right in Fig. 2.13b is the resulting 8-bit color image resulting from running the Matlab program in Listing 2.1. The sprite image of the two loons in Fig. 2.13b is the same 100×100 loon image we used in Example 9.

The top-level design used to move the loons in Fig. 2.13b is shown in Fig. 2.14. The background image stored in the flash memory is constantly being displayed on the screen using the same state machine we used in Example 14. This occurs in the module *vga_flash_sprite* in Fig. 2.14, the VHDL code of which is given in Listing 2.6. The *loons100x100* block ROM module displays the loon sprite on the screen with its upper left corner at the screen location (*C1*, *R1*) using the same technique as in Example 9. To move the loon image, we just need to change the values of (*C1*, *R1*) similar to the way we did in the screen saver program in Example 6. The module *move_loons* in Fig. 2.14 will

provide these values of (*C1*, *R1*). The VHDL code for this *move_loons* module is given in Listing 2.7.

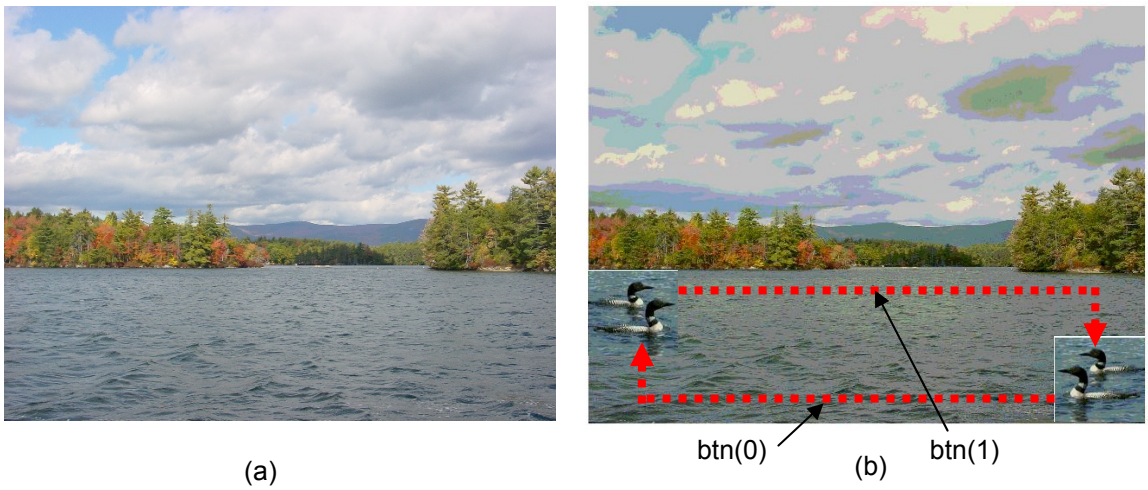

Figure 2.13 (a) Original 640x480 image; (b) 8-bit image showing motion of the loons

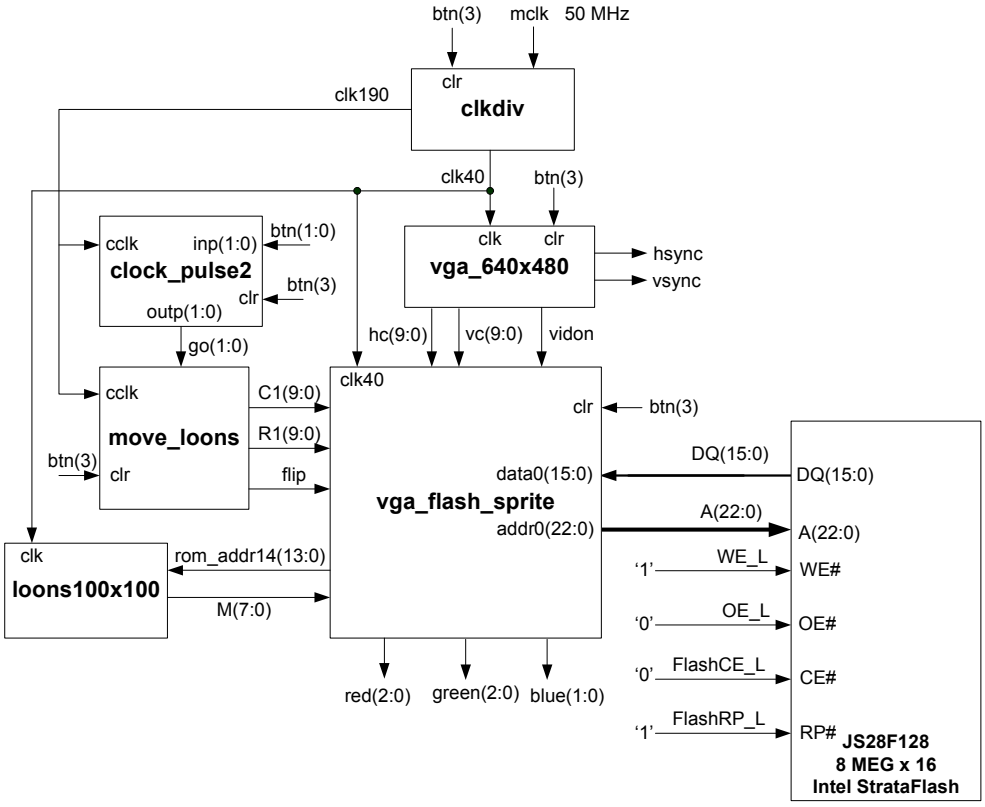

Figure 2.14 Top-level design for Example 15

Listing 2.6 vga_flash_sprite.vhd

```
-- Example 15a: vga_flash_sprite
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity vga_flash_sprite is
     port ( clk40, clr : in std_logic;
            vidon: in std_logic;
            flip: 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);
            C1, R1: in std_logic_vector(9 downto 0);
            data0 : in std_logic_vector(15 downto 0);
            rom_addr14: out std_logic_vector(13 downto 0);
            addr0 : out std_logic_vector(22 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_flash_sprite;

architecture vga_flash_sprite of vga_flash_sprite is
signal pixel: std_logic_vector(7 downto 0);
signal px1,px2,px3, px4: std_logic_vector(15 downto 0);
signal addr_count: std_logic_vector(19 downto 0);

     --counter for address bus
type state_type is (start, s1, s2, px1h0, px1l0, px1h, px1l,
                           px2h, px2l,px3h, px3l,px4h, px4l);
signal state: state_type;
constant addr_max: std_logic_vector(19 downto 0) := X"25800";
      --Max address = 320x480 = 153,600 = X"25800"
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 := 100;
constant h: integer := 100;
signal xpix, ypix: std_logic_vector(9 downto 0);
signal spriteon: std_logic;

begin
         ypix <= vc - vbp - R1;
loondir: process(hc)
         begin
            if flip = '1' then
                  xpix <= hbp + C1 + w - hc;
            else
                  xpix <= hc - hbp - C1;
            end if;
         end process loondir;
```

Listing 2.6 (cont.) vga_flash_sprite.vhd

```
      --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';

loonaddr: process(xpix, ypix)
      variable  rom_addr1, rom_addr2: STD_LOGIC_VECTOR (16 downto 0);
      begin
        rom_addr1 := ('0' & ypix & "000000") + ("00" & ypix & "00000")
          + ("00000" & ypix & "00");      -- y*(64+32+4) = y*100
        rom_addr2 := rom_addr1 + ("0000000" & xpix);  -- y*100+x
        rom_addr14 <= rom_addr2(13 downto 0);
      end process loonaddr;


faddr: process(clk40, clr)
  begin
    if clr = '1' then
            state <= start;
            addr_count <= (others => '0');
            px1 <= (others => '0');
            px2 <= (others => '0');
            px3 <= (others => '0');
            px4 <= (others => '0');
    elsif (clk40'event and clk40 = '1') then
      case state is
            when start =>
               state <= s1;
            when s1 =>
               state <= s2;
            when s2 =>
               state <= px1h0;
               px1 <= data0;
               addr_count <= addr_count + 1; --Increment address
            when px1h0 =>
               if vidon = '1' then
                  state <= px1l;
                  px2 <= data0;
                  addr_count <= addr_count + 1; --Increment address
               else
                  state <= px1h0;
               end if;
            when px1l =>
               state <= px2h;
               px3 <= data0;
               if addr_count = addr_max - 1 then
                  --Reset when max address is reached
                  addr_count <= (others => '0');
               else
                  if vidon = '1' then
                     addr_count <= addr_count + 1;
                  end if;
               end if;
```

Listing 2.6 (cont.) vga_flash_sprite.vhd

```
            when px2h =>
               state <= px2l;
               px4 <= data0;
               if addr_count = addr_max - 1 then
                  --Reset when max address is reached
                  addr_count <= (others => '0');
               else
                  if vidon = '1' then
                     addr_count <= addr_count + 1;
                  end if;
               end if;
            when px2l =>
               state <= px3h;
            when px3h =>
               state <= px3l;
            when px3l =>
               state <= px4h;
            when px4h =>
               state <= px4l;
            when px4l =>
               state <= px1h;
               px1 <= data0;
               if addr_count = addr_max - 1 then
                  --Reset when max address is reached
                  addr_count <= (others => '0');
               else
                  if vidon = '1' then
                     addr_count <= addr_count + 1; --Increment address
                  end if;
               end if;
            when px1h =>
               state <= px1l;
               px2 <= data0;
               if addr_count = addr_max - 1 then
                  --Reset when max address is reached
                  addr_count <= (others => '0');
               else
                  if vidon = '1' then
                     addr_count <= addr_count + 1; --Increment address
                  end if;
               end if;
            when others =>
               null;
      end case;
    end if;
  end process faddr;


C2: process(state,px1, px2, px3, px4)
begin
   pixel <= (others => '0');
   case state is
      when px1h =>
            pixel <= px1(15 downto 8);
```

Listing 2.6 (cont.) vga_flash_sprite.vhd

```
      when px1l =>
            pixel <= px1(7 downto 0);
      when px2h =>
            pixel <= px2(15 downto 8);
      when px2l =>
            pixel <= px2(7 downto 0);
      when px3h =>
            pixel <= px3(15 downto 8);
      when px3l =>
            pixel <= px3(7 downto 0);
      when px4h =>
            pixel <= px4(15 downto 8);
      when px4l =>
            pixel <= px4(7 downto 0);
      when px1h0 =>
            pixel <= px1(15 downto 8);
      when others =>
            null;
   end case;
end process C2;

color: process(spriteon, vidon, M, pixel)
      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);
            elsif vidon = '1' then
                  red <= pixel(7 downto 5);
                  green <= pixel(4 downto 2);
                  blue <= pixel(1 downto 0);
            end if;
end process color;

addr0 <= "000" & addr_count;

end vga_flash_sprite;
```

In Listing 2.6, the process *loondir* will determine which direction the loons are facing depending on the value of the input *flip*. The value of *flip* is set in Listing 2.7. The initial value of flip is 0, corresponding to the loons facing west (left). This value of flip stays 0 when you press *btn*(0), corresponding to *go*(0) = '1'. When the loons reach the destination row, the value of *flip* is changed to 1, corresponding to the loons facing east (right). This value of *flip* stays 1 when you press *btn*(1), corresponding to *go*(1) = '1'. When the loons reach the starting row, the value of *flip* is changed back to 0, corresponding to the loons facing west again.

The process *loonaddr* in Listing 2.6 calculates the address of the 100×100 loon image stored in the block ROM l*oons100x100* as we did in Example 9. The processes *faddr* and *C2* in Listing 2.6 are used to read the background image from the flash memory as was done in Example 14. In the process *color* in Listing 2.6, note how displaying the loon sprite takes precedence over displaying the background image in the flash memory. This means that the loons will appear in front of the background image.

The method used in Listing 2.7 to move the loons is the same as we used in the screen saver program in Example 6. There are two *go* inputs in Listing 2.7 corresponding to pressing buttons 0 and 1. These *go* inputs must be single clock pulse signals and are generated from the module *clock_pulse2* in Fig. 2.14. The VHDL code for this module is given in Listing 2.8. When *btn*(0) is pressed, a single debounced pulse is generated on the output *go*(0). Similarly, when *btn*(1) is pressed, a single debounced pulse is generated on the output *go*(1). A detailed description of the *clock_pulse* circuit is given in Example 48 of our VHDL digital design book.[4]

The VHDL code for the top-level design in Fig. 2.14 is given in Listing 2.9.

Listing 2.7 move_loons.vhd

```
-- Example 15b: move_loons
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_arith.all;
use IEEE.STD_LOGIC_unsigned.all;

entity move_loons is
       port(
             cclk : in STD_LOGIC;
             clr : in STD_LOGIC;
             go : in STD_LOGIC_VECTOR(1 downto 0);
             flip : out std_logic;
             c1 : out STD_LOGIC_VECTOR(9 downto 0);
             r1 : out STD_LOGIC_VECTOR(9 downto 0)
           );
end move_loons;

architecture move_loons of move_loons is
constant c1s: integer := 535;
constant c1d: integer := 5;
constant r1s: integer := 380;
constant r1d: integer := 305;
begin
```

[4] Haskell, R. E. and D. M. Hanna, *Digital Design Using Digilent FPGA Boards – VHDL/Active-HDL Edition*, LBE Books, Auburn Hills, MI, 2nd Ed., 2014.

Listing 2.7 (cont.) move_loons.vhd

```
process(cclk, clr)
variable c1v, r1v: STD_LOGIC_VECTOR(9 downto 0);
variable calc: STD_LOGIC_VECTOR(1 downto 0);
begin
   if clr = '1' then
      c1v := conv_std_logic_vector(c1s,10);
      r1v := conv_std_logic_vector(r1s,10);
      calc := "00";
      flip <= '0';
   elsif cclk'event and cclk = '1' then
      if go(0) = '1' then
         calc(0) := '1';
         flip <= '0';
      end if;
      if calc(0) = '1' then
         if r1v = r1d then
            calc(0) := '0';
            flip <= '1';
         else
            if (c1v > c1d) then
               c1v := c1v - 1;
            elsif (r1v > r1d) then
               r1v := r1v - 1;
            end if;
         end if;
      end if;
      if go(1) = '1' then
         calc(1) := '1';
         flip <= '1';
      end if;
      if calc(1) = '1' then
         if r1v = r1s then
            calc(1) := '0';
            flip <= '0';
         else
            if (c1v < c1s) then
               c1v := c1v + 1;
            elsif (r1v < r1s) then
               r1v := r1v + 1;
            end if;
         end if;
      end if;
   end if;

   c1 <= c1v;
   r1 <= r1v;

end process;

end move_loons;
```

Listing 2.8 clock_pulse.vhd

```
-- Example 15c: clock pulse
library IEEE;
use IEEE.std_logic_1164.all;

entity clock_pulse2 is
      port (
            cclk, clr: in std_logic;
            inp: in std_logic_vector(1 downto 0);
            outp: out std_logic_vector(1 downto 0)
      );
end clock_pulse2;

architecture clock_pulse2_arch of clock_pulse2 is
signal delay1, delay2, delay3: std_logic_vector(1 downto 0);
begin
    process(cclk, clr)
    begin
       if clr = '1' then
            delay1 <= "00";
            delay2 <= "00";
            delay3 <= "00";
       elsif cclk'event and cclk='1' then
            delay1 <= inp;
            delay2 <= delay1;
            delay3 <= delay2;
       end if;
    end process;
    outp <= delay1 and delay2 and (not delay3);
end clock_pulse2_arch;
```

Listing 2.9 vga_flash_move_loons_top.vhd

```
-- Example 15: vga_flash_move_loons_top
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.std_logic_unsigned.all;
use work.vga_components.all;

entity vga_flash_move_loons_top is
       port(
            mclk : in STD_LOGIC;
            btn : in STD_LOGIC_VECTOR(3 downto 0);
            hsync, 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);
            A : out STD_LOGIC_VECTOR(22 downto 0);
            DQ : in STD_LOGIC_VECTOR(15 downto 0);
            FlashCE_L : out STD_LOGIC;
            FlashRp_L: out STD_LOGIC;
            CE_L, WE_L, OE_L : out STD_LOGIC
          );
end vga_flash_move_loons_top;
```

Listing 2.9 (cont.) vga_flash_move_loons_top.vhd

```
architecture vga_flash_move_loons_top of vga_flash_move_loons_top is
signal clr, clk40, clk190, vidon, flip: std_logic;
signal hc, vc, C1, R1: std_logic_vector(9 downto 0);
signal M: std_logic_vector(7 downto 0);
signal data0: std_logic_vector(15 downto 0);
signal rom1_addr14: std_logic_vector(13 downto 0);
signal go: STD_LOGIC_VECTOR(1 downto 0);
begin
      clr <= btn(3);
      FlashCE_L <= '0';       -- Enable flash
      flashRp_L <= '1';
      CE_L <= '1';
      OE_L <= '0';
      WE_L <= '1';
      -- flash stores data little endian
      data0(15 downto 8) <= DQ(7 downto 0);
      data0(7 downto 0) <= DQ(15 downto 8);

U1 : clkdiv40190
      port map(
            mclk => mclk, clr => clr, clk190 => clk190, clk40 => clk40
      );

U2 : vga_flash_sprite
      port map(
            clk40 => clk40, clr => clr, vidon => vidon, flip => flip,
            hc => hc, vc => vc, M => M, C1 => C1, R1 => R1,
            rom1_addr14 => rom1_addr14, data0 => data0, addr0 => A,
            red => red, green => green, blue => blue
      );

U3 : vga_640x480
      port map(
            clk => clk40, clr => clr, hsync => hsync, vsync => vsync,
            hc => hc, vc => vc, vidon => vidon
      );

U4: loons100x100
            port map (
                  addr => rom1_addr14, clk => clk40, dout => M);

U5 : move_loons
      port map(
            cclk => clk190, clr => clr, flip => flip, go => go,
            c1 => c1, r1 => r1
      );

U6 : clock_pulse2
      port map(
            cclk => clk190, clr => clr,
            inp => btn(1 downto 0), outp => go
      );

end vga_flash_move_loons_top;
```