Christopher Spadavecchia
Eli Shtindler
CPE 487

Lab 3

Project 1 - vgaball:

```
set_property PROBES.FILE {} [get_hw_devices xc7a100t_0]
set_property FULL_PROBES.FILE {} [get_hw_devices xc7a100t_0]
set_property PROGRAM.FILE {C:/Users/elism/Vivado Projects/Lab3/Lab3.runs/impl_1/vga_top.bit} [get_hw_devices xc7a100t_0]
program_hw_devices [get_hw_devices xc7a100t_0]
INFO: [Labtools 27-3164] End of startup status: HIGH
refresh_hw_device [lindex [get_hw_devices xc7a100t_0] 0]
INFO: [Labtools 27-1434] Device xc7a100t (JTAG device index = 0) is programmed with a design that has no supported debug core(s) in it.
```

After uploading the code to the board was successful (the aftermath of the 'Program Device' step), the following lines were shown in the console.

Original Ball Bouncing Vertically

After uploading the original code, the monitor displayed a red square ball bouncing vertically.

Modified Ball Bouncing like the DVD logo

After modifying the code, the monitor displayed a blue circle ball bouncing vertically and horizontally.

ball.vhd

Original:

```
ARCHITECTURE Behavioral OF ball IS
	CONSTANT size  : INTEGER := 8;
	SIGNAL ball_on : STD_LOGIC; -- indicates whether ball is over current pixel position
	-- current ball position - intitialized to center of screen
	SIGNAL ball_x  : STD_LOGIC_VECTOR(10 DOWNTO 0) := CONV_STD_LOGIC_VECTOR(400, 11);
	SIGNAL ball_y  : STD_LOGIC_VECTOR(10 DOWNTO 0) := CONV_STD_LOGIC_VECTOR(300, 11);
	-- current ball motion - initialized to +4 pixels/frame
	SIGNAL ball_y_motion : STD_LOGIC_VECTOR(10 DOWNTO 0) := "00000000100";
BEGIN
	red <= '1'; -- color setup for red ball on white background
	green <= NOT ball_on;
	blue  <= NOT ball_on;
	-- process to draw ball current pixel address is covered by ball position
	bdraw : PROCESS (ball_x, ball_y, pixel_row, pixel_col) IS
	BEGIN
```

```
            IF (pixel_col >= ball_x - size) AND
             (pixel_col <= ball_x + size) AND
                    (pixel_row >= ball_y - size) AND
                    (pixel_row <= ball_y + size) THEN
                          ball_on <= '1';
            ELSE
                   ball_on <= '0';
            END IF;
            END PROCESS;
            -- process to move ball once every frame (i.e. once every vsync pulse)
            mball : PROCESS
            BEGIN
                   WAIT UNTIL rising_edge(v_sync);
                   -- allow for bounce off top or bottom of screen
                   IF ball_y + size >= 600 THEN
                          ball_y_motion <= "11111111100"; -- -4 pixels
                   ELSIF ball_y <= size THEN
                          ball_y_motion <= "00000000100"; -- +4 pixels
                   END IF;
                   ball_y <= ball_y + ball_y_motion; -- compute next ball position
            END PROCESS;
END Behavioral;
```

Modified:

```
ARCHITECTURE Behavioral OF ball IS
       CONSTANT size  : INTEGER := 30;
       SIGNAL ball_on : STD_LOGIC; -- indicates whether ball is over current pixel position
       -- current ball position - intitialized to center of screen
       SIGNAL ball_x  : STD_LOGIC_VECTOR(10 DOWNTO 0) := CONV_STD_LOGIC_VECTOR(400, 11);
       SIGNAL ball_y  : STD_LOGIC_VECTOR(10 DOWNTO 0) := CONV_STD_LOGIC_VECTOR(300, 11);
       -- current ball motion - initialized to +4 pixels/frame
       SIGNAL ball_y_motion : STD_LOGIC_VECTOR(10 DOWNTO 0) := "00000000100";
       SIGNAL ball_x_motion : STD_LOGIC_VECTOR(10 DOWNTO 0) := "00000000100";

BEGIN
       red <= NOT ball_on; -- color setup for blue ball on white background
       green <= NOT ball_on;
       blue  <= '1';
       -- process to draw ball current pixel address is covered by ball position
       bdraw : PROCESS (ball_x, ball_y, pixel_row, pixel_col) IS
           VARIABLE diff_x, diff_y : INTEGER;
       BEGIN
           diff_x := conv_integer(ball_x) - conv_integer(pixel_col); -- difference between
ball x and pixel x
           diff_y := conv_integer(ball_y) - conv_integer(pixel_row); -- difference between
ball y and pixel y
           IF (diff_x*diff_x + diff_y*diff_y) <= (size*size) then -- distance formula
               ball_on <='1';
```

```
		ELSE
			ball_on <= '0';
		END IF;
		END PROCESS;
		-- process to move ball once every frame (i.e. once every vsync pulse)
		mball : PROCESS
		BEGIN
			WAIT UNTIL rising_edge(v_sync);
			-- allow for bounce off top or bottom of screen
			IF ball_y + size >= 600 THEN
				ball_y_motion <= "11111111100"; -- -4 pixels
			ELSIF ball_y <= size THEN
				ball_y_motion <= "00000000100"; -- +4 pixels
			END IF;
			-- allow for bounce off left or right of screen
			IF ball_x + size >= 800 THEN
				ball_x_motion <= "11111111100"; -- -4 pixels
			ELSIF ball_x <= size THEN
				ball_x_motion <= "00000000100"; -- +4 pixels
			END IF;

			ball_y <= ball_y + ball_y_motion; -- compute next ball position
			ball_x <= ball_x + ball_x_motion; -- compute next ball position
		END PROCESS;
END Behavioral;
```

In order to change the size of the ball, the size integer was increased. The color of the ball was changed by modifying the values of red, green, and blue. By setting blue to always be 1 instead of red, the color of the ball was changed to blue. In order to change the square ball to a round, the ball signal was set to 1 whenever the current pixel was within a certain distance of the ball's center. In order to do this additional calculations were required. First the difference between the current pixel address and the ball's center was found for the x and y directions. Distance formula was then used to see if the pixel was within the radius of the ball. In order to make the ball move in the x direction, the existing code that moved the ball in the y direction was replicated. The names of the variables were changed and 800 was used as the new bound because the display was 800 pixels wide as opposed to only 600 pixels tall.