```
-- Message module
-- This module will get the signals from dice component and display the
-- constant message and some changeable message.
library IEEE;
use  IEEE.STD_LOGIC_1164.all;
use  IEEE.STD_LOGIC_ARITH.all;
use  IEEE.STD_LOGIC_UNSIGNED.all;
LIBRARY lpm;
USE lpm.lpm_components.ALL;
entity play_ms is
        PORT(
        clock                           : in STD_LOGIC;
        vclock                          : in STD_LOGIC;
        reset                           : in STD_LOGIC;
        piece1_wins, piece2_wins        : in boolean;
        piece1, piece2                  : in BOOLEAN;
        step_size                       : in STD_LOGIC_VECTOR(2 downto 0);
        vga_red, vga_green, vga_blue    : out BOOLEAN;
        countX,countY                   : in STD_LOGIC_VECTOR(9 downto 0)
        );
END play_ms;

architecture dice_message of play_ms is

-- Video Display Signals
-- Signals for Video ROM Memory for Pixel Data
signal rom_address: std_logic_vector(8 Downto 0);
signal rom_data: std_logic_vector(7 Downto 0);
signal col_address, row_address, con_row, con_col: std_logic_vector(5 Downto 0);
signal pixel_col_count, pixel_row_count: std_logic_vector(5 Downto 0);
signal rom_mux_output: std_logic;
signal format_address: std_logic_vector(5 downto 0);
signal format_data: std_logic_vector(5 downto 0);
signal red_data,green_data, blue_data: boolean;
signal in_screen, in_screen_vertical, in_screen_horizontal: boolean;

SIGNAL  piece1_vga_red,  piece1_vga_green,  piece1_vga_blue   : BOOLEAN;
SIGNAL  piece2_vga_red,  piece2_vga_green,  piece2_vga_blue   : BOOLEAN;

begin

 --Small 8 by 8 Character Genrator ROM for Video Display
 tiny_char_gen_rom: lpm_rom
      GENERIC MAP ( lpm_widthad => 9,
        lpm_numwords => "512",
        lpm_outdata => "UNREGISTERED",
        lpm_address_control => "UNREGISTERED",
-- Reads in mif file for character generator data
         lpm_file => "char_set.mif",
         lpm_width => 8)
      PORT MAP ( address => rom_address, q => rom_data);
-- Character Format ROM for Video Display
-- Displays constant format character data
-- on left side of Display area
format_rom: lpm_rom
      GENERIC MAP ( lpm_widthad => 6,
        lpm_numwords => "48",
        lpm_outdata => "UNREGISTERED",
        lpm_address_control => "UNREGISTERED",
-- Reads in mif file for data display format
        lpm_file => "dis_con2.mif",
         lpm_width => 6)
      PORT MAP ( address => format_address, q => format_data);
-- Display 8 by 8 font with 16 by 16 pixel array
```

```
-- Step through rows for each character pattern

output: process(clock)
begin
if clock'EVENT and clock= '1' then
-- local R,G,B data
red_data   <= (rom_mux_output = '1') or piece1_vga_red or piece2_vga_red;
Green_data <= (rom_mux_output = '1') or piece1_vga_green or piece2_vga_green;
Blue_data  <= (rom_mux_output = '1') or piece1_vga_blue or piece2_vga_blue;

rom_address(2 Downto 0) <= pixel_row_count(3 Downto 1);

-- Mux to pick off correct rom data bit from 8-bit word
-- for on screen character generation
rom_mux_output <= rom_data ( (CONV_INTEGER(NOT pixel_col_count(3 downto 1))));

-- decide the display area
in_screen  <= in_screen_horizontal and in_screen_vertical;

-- The RGB signal pins to the VGA monitor
vga_Red    <= red_data and in_screen;
vga_Green  <= green_data and in_screen;
vga_Blue   <= blue_data and in_screen;

end if;
end process output;



video_display: process(clock)
begin
if clock'event and clock ='1' then

col_address <= countX(9 downto 4);
pixel_col_count <= countX(3 downto 0);
row_address <= countY(9 downto 4);
pixel_row_count <= countY(3 downto 0);
in_screen_horizontal <= col_address >= "000000" and col_address <= "100111";
in_screen_vertical <= row_address >= "000000" and row_address <= "011101";
end if;
end process video_display;



-- Address for Constant Character Data ROM
format_address(2 Downto 0) <= con_col(2 Downto 0);
format_address(5 Downto 3) <= con_row(2 Downto 0);

-- This Process Provides Character Data for Video Display
-- by generating addresses for the Character Generator ROM
-- using row address and col address provided by the Video
-- Display process

VIDEO_DISPLAY_DATA: process(clock)
begin

IF clock'EVENT and clock = '1' THEN
                if piece1 then
                        piece1_vga_red <= col_address = "000001" and
                        row_address = "001010";
                piece1_vga_green <= false;
                        piece1_vga_blue  <= col_address = "000001" and
                        row_address = "001010";
                end if;
```

```
                if piece2 or piece2_wins then
                        piece2_vga_red <= false;
                        piece2_vga_green <= col_address = "000001" and
                        row_address = "001100";
                        piece2_vga_blue  <= col_address = "000001" and
                        row_address = "001100";
                end if;

-- displaying "PLAYER 1"
        if col_address >= "000011" and col_address <= "001010" and
                row_address = "001010" THEN
                        con_col <= col_address - "000011";
                        con_row <= row_address - "001010";
                        rom_address(8 downto 3) <= format_data;

-- displaying "PLAYER 2"
        elsif col_address >= "000011" and col_address <= "001010" and
                        row_address = "001100" THEN
                        con_col <= col_address - "000011";
                        con_row <= row_address - "001011";
                        rom_address(8 downto 3) <= format_data;

-- displaying "STEP: "
        elsif col_address >= "000011" and col_address <= "001000" and
                        row_address = "001110" THEN
                        con_col <= col_address - "000011";
                        con_row <= row_address - "001100";
                        rom_address(8 downto 3) <= format_data;

-- displaying "STOP"
        elsif col_address >= "100010" and col_address <= "100101" and
                        row_address = "001010" then
                        con_col <= col_address - "010010";
                        con_row <= row_address - "000111";
                        rom_address(8 downto 3) <= format_data;

-- displaying the message "BONUS"
        elsif col_address >= "100010" and col_address <= "100110" and
                        row_address = "001100" THEN
                        con_col <= col_address - "010010";
                        con_row <= row_address - "001000";
                        rom_address(8 downto 3) <= format_data;

-- displaying the message "BACK" - meaning moving backwards
        elsif col_address >= "100010" and col_address <= "100101" and
                        row_address = "001110" THEN
                        con_col <= col_address - "010010";
                        con_row <= row_address - "001001";
                        rom_address(8 downto 3) <= format_data;


-- displaying the step size
        elsif col_address = "001001" and row_address = "001110" then
                rom_address(8 downto 3) <= step_size + "110000";

        ELSE
                rom_address(8 downto 3) <= "100000";
        END IF;
END IF;
end process VIDEO_DISPLAY_DATA;
end dice_message;
```