```
LIBRARY IEEE;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;
USE work.dice_states.all;

ENTITY dice IS
        PORT(
        clock                           : in STD_LOGIC;
        reset                           : in STD_LOGIC;
        PB2                             : in std_logic;
        HSync, VSync                    : in BOOLEAN;
        countX, countY                  : in STD_LOGIC_VECTOR(9 downto 0);
        step_size                       : in STD_LOGIC_VECTOR(2 downto 0);
        piece1, piece2                  : buffer boolean;
        vga_red, vga_green, vga_blue: out BOOLEAN;
        piece1_wins, piece2_wins        : out BOOLEAN;
        bonus, stop, penalty            : out BOOLEAN
        );
END dice;

ARCHITECTURE background_dice OF dice IS

SIGNAL  piece1_vga_red,  piece1_vga_green,  piece1_vga_blue   : BOOLEAN;
SIGNAL  piece2_vga_red,  piece2_vga_green,  piece2_vga_blue   : BOOLEAN;
SIGNAL step             : std_logic_vector(2 downto 0);
-- step size necessary to take

--------------------------------------------------------------------------
SIGNAL  throw_state, next_throw_state : throw_dice_state;
SIGNAL  piece1_done, piece2_done : BOOLEAN;
--------------------------------------------------------------------------
TYPE    pb2_state_type is (done, wait_for_step_size, move);
SIGNAL  pb2_next_state,pb2_state        :  pb2_state_type;
--------------------------------------------------------------------------
SIGNAL  equal_x2, equal_y2              :  BOOLEAN;
SIGNAL  state2, next_state2             : board_row_state;
SIGNAL  posX2, posY2                    : STD_LOGIC_VECTOR(9 downto 0);
--------------------------------------------------------------------------
SIGNAL  equal_x1, equal_y1              :  BOOLEAN;
SIGNAL  posX1, posY1                    : STD_LOGIC_VECTOR(9 downto 0);
signal  state1, next_state1             : board_row_state;
-------------------------------------------------------------------------
SIGNAL  piece1_stop, piece2_stop        : BOOLEAN;
SIGNAL  piece1_bonus, piece2_bonus      : BOOLEAN;
SIGNAL  piece1_penalty, piece2_penalty  : BOOLEAN;
-------------------------------------------------------------------------
SIGNAL  stop_position_p1, stop_position_p2      : BOOLEAN;
SIGNAL  bonus_position_p1, bonus_position_p2    : BOOLEAN;
SIGNAL  penalty_position_p1, penalty_position_p2: BOOLEAN;

BEGIN
--------------------------------------------------------------------------
state_update : process (clock)
begin
        if reset = '0' then
                state1 <= initializing;
                state2 <= initializing;
                throw_state <= reset_both;
                pb2_state <= done;
        ELSIF clock'EVENT and clock = '1' THEN
                state1 <= next_state1;
                state2 <= next_state2;
                pb2_state <= pb2_next_state;
```

```
                throw_state <= next_throw_state;
                bonus <= piece1_bonus or piece2_bonus;
                penalty <= piece1_penalty or piece2_penalty;
                stop <= piece1_stop or piece2_stop;
        end if;
end process state_update;

---------------------------------------------------------------------------
throw_dice_control : process(clock, throw_state)
begin
IF clock'EVENT and clock ='1' THEN

        case throw_state is

                when reset_both =>
                        piece1_done <= false;
                        piece2_done <= false;

                        piece1_bonus <=false;
                        piece1_stop <= false;
                        piece1_penalty <=false;


                        piece1 <= true;
                        piece2 <= true;
                        next_throw_state <=end_piece2;

                when move_piece1 =>
                        if piece1_stop then
                        piece1 <= false;
                        piece2 <= true;
                        else
                        piece1 <= true;
                        piece2 <= false;
                        end if;

                        if piece1_stop then
                                next_throw_state <= end_piece1;
                        elsif PB2_state = done then
                                next_throw_state <= status_piece1;
                        end if;

                when status_piece1 =>
                        piece1 <= false;
                        piece2 <= true;

                        if (posX1(9 downto 4) = second_column_x1 and
state1 = board_row_7) or
                         (posX1(9 downto 4) = seventh_column_x1 and
                         state1 = board_row_3)   then

                                next_throw_state <= stop_piece1;
                        -- piece1 loses one draw stop block

                        elsif (posX1(9 downto 4) = third_column_x1 and
                        state1 = board_row_2) or
                        (posX1(9 downto 4) = eighth_column_x1
                        and state1 = board_row_6)    then

                                next_throw_state <= bonus_piece1;
                        -- get a random bonus step

                        elsif (posX1(9 downto 4) = sixth_column_x1
                        and state1 = board_row_8) or
```

```
        (posX1(9 downto 4) = fifth_column_x1
        and state1 = board_row_4)   then

                next_throw_state <= penalty_piece1;
        -- get a random penalty step
        else
                next_throw_state <= end_piece1;
        -- decide the end of piece1
        end if;
when stop_piece1 =>
        piece1 <= false;
        piece2 <= true;

        piece1_bonus <=false;
        piece1_stop <=true;
        piece1_penalty <=false;

        if PB2_state = wait_for_step_size then
                next_throw_state <= move_piece2;
        end if;

when bonus_piece1 =>
        piece1 <= true;
        piece2 <= false;

        piece1_bonus <= true;
        piece1_stop <= false;
        piece1_penalty <=false;

        if PB2_state = wait_for_step_size then
                next_throw_state <= move_piece1;
        end if;

when penalty_piece1 =>
        piece1 <= true;
        piece2 <= false;

        piece1_bonus <=false;
        piece1_stop <=false;
        piece1_penalty <=true;

        if PB2_state = wait_for_step_size then
                next_throw_state <= move_piece1;
        end if;

when end_piece1 =>
        if piece2_stop THEN

        piece1 <= true;
        piece2 <= false;
        else
        piece1 <= false;
        piece2 <= true;
        end if;

        piece1_bonus <=false;
        piece1_stop <=false;
        piece1_penalty <=false;

        if (posX1 = left_edge_x1) and
                (state1 = board_row_8) then
                piece1_done <= true;
        end if;
```

```
                if PB2_state = wait_for_step_size then
                        next_throw_state <= move_piece2;
                end if;
--------------------------------------------------------------------------------

        when move_piece2 =>
                if piece2_stop then
                piece1 <= true;
                piece2 <= false;
                else
                piece1 <= false;
                piece2 <= true;
                end if;


                if piece2_stop then
                        next_throw_state <= end_piece2;

                elsif PB2_state = done then
                        next_throw_state <= status_piece2;
                end if;

        when status_piece2 =>
                piece1 <= true;
                piece2 <= false;

                if (posX2(9 downto 4) = second_column_x2
                and state2 = board_row_7) or
                 (posX2(9 downto 4) = seventh_column_x2
                 and state2 = board_row_3)  then

                        next_throw_state <= stop_piece2;
                -- piece2 loses one draw
                elsif (posX2(9 downto 4) = third_column_x2
                and state2 = board_row_2) or
                (posX2(9 downto 4) = eighth_column_x2
                and state2 = board_row_6)  then

                        next_throw_state <= bonus_piece2;

                -- get a random bonus step
                elsif (posX2(9 downto 4) = sixth_column_x2 and
                state2 = board_row_8) or
                (posX2(9 downto 4) = fifth_column_x2 and
                state2 = board_row_4) then

                        next_throw_state <= penalty_piece2;
                -- get a random penalty step
                else
                        next_throw_state <= end_piece2;
                -- decide the end of piece1
                end if;


        when stop_piece2 =>
                piece1 <= true;
                piece2 <= false;

                piece2_bonus <=false;
                piece2_stop <=true;
                piece2_penalty <=false;

                next_throw_state <= who_wins;
```

```
        when bonus_piece2 =>
                piece1 <= false;
                piece2 <= true;

                piece2_bonus <=true;
                piece2_stop <= false;
                piece2_penalty <=false;

                if PB2_state = wait_for_step_size then
                        next_throw_state <= move_piece2;
                end if;

        when penalty_piece2 =>
                piece1 <= false;
                piece2 <= true;

                piece2_bonus <=false;
                piece2_stop <= false;
                piece2_penalty <=true;

                if PB2_state = wait_for_step_size then
                        next_throw_state <= move_piece2;
                end if;

        when end_piece2 =>
                if piece1_stop THEN
                piece1 <=false;
                piece2 <= true;
                else
                piece1 <= true;
                piece2 <=false;
                end if;


                piece2_bonus <=false;
                piece2_stop <= false;
                piece2_penalty <=false;

                next_throw_state <= who_wins;

                if (posX2 = left_edge_x2) and (state2 = board_row_8) then
                        piece2_done <= true;
                end if;

        when who_wins =>
                if piece1_done then
                        if piece2_done then
                                piece1_wins <= true;
                                piece2_wins <= true;
                                piece1 <= true;
                                piece2 <= true;
                        else
                                piece1_wins <= true;
                                piece2_wins <= false;
                                piece1 <= true;
                                piece2 <= false;
                        end if;
                        next_throw_state <= who_wins;
                elsif piece2_done then
                        piece1_wins <= false;
                        piece2_wins <= true;
                        next_throw_state <= who_wins;
                        piece1 <= false;
                        piece2 <= true;
```

```
                        elsif PB2_state = wait_for_step_size then
                                piece1_wins <= false;
                                piece2_wins <= false;
                                next_throw_state <= move_piece1;
                        end if;

                when others =>
                        next_throw_state <= end_piece2;
                end case;
end if;
end process throw_dice_control;
--------------------------------------------------------------------------------
PB2_states: process (clock,pb2_state)
begin
IF clock'EVENT and clock ='1' THEN
        IF VSync THEN
                CASE pb2_state IS

                        WHEN done =>
                                IF PB2 = '1' THEN
                                        pb2_next_state <= done;
                                ELSE
                                        pb2_next_state <= wait_for_step_size ;
                                END IF;
                        WHEN wait_for_step_size =>
                                step <= step_size;
                                IF PB2 = '1' THEN
                                        pb2_next_state <= move;
                                ELSE
                                        pb2_next_state <= wait_for_step_size;
                                END IF;
                        WHEN move =>
                            step <= step - '1';
                                IF step   <= "001" THEN
                                        pb2_next_state <=done;
                                END IF;
                        WHEN OTHERS=>
                                pb2_next_state <= done;
                END CASE;
     END IF;
END IF;
end process PB2_states;
----------------------------------------------------------------------------

Generate_Newxy1: process(state1,clock)
BEGIN
IF clock'EVENT and clock = '1' THEN
        IF   VSync  AND ( (reset = '0') OR (pb2_state = move)) and
                 piece1 THEN
                CASE state1 IS
---------------------------------------------------------------------------
                        WHEN initializing =>
                                posX1 <= startx1;
                                posY1 <= starty1;
                                next_state1 <= board_row_1;
---------------------------------------------------------------------------
                        WHEN board_row_1 =>
                                posY1 <= starty1;
                                IF posX1  = right_edge_x1  THEN
                                                next_state1 <= board_row_2;
                                                posY1 <= posY1 + Dy;
                                ELSE
                                        posX1 <= posX1 + Dx;
                                END IF;
```

```
--------------------------------------------------------------------------
                        WHEN board_row_2 =>

                                IF posX1  = left_edge_x1  THEN
                                                next_state1 <= board_row_3;
                                                posY1 <= posY1 + Dy;

                                ELSE

                                                posX1 <= posX1 - Dx;
                                END IF;
-----------------------------------------------------------------------------
                        WHEN board_row_3 =>

                                IF piece1_penalty THEN
                                        IF posX1  = left_edge_x1  THEN
                                                next_state1 <= board_row_2;
                                                posY1 <= posY1 - Dy;

                                        ELSE
                                                posX1 <= posX1 - Dx;
                                        END IF;
                                ELSE
                                        IF posX1  = right_edge_x1  THEN
                                                next_state1 <= board_row_4;
                                                posY1 <= posY1 + Dy;
                                        ELSE
                                                posX1 <= posX1 + Dx;
                                        END IF;
                                END IF;


---------------------------------------------------------------------------
                        WHEN board_row_4 =>
                                IF piece1_penalty THEN
                                        IF posX1  = right_edge_x1  THEN
                                                next_state1 <= board_row_3;
                                                posY1 <= posY1 - Dy;
                                        ELSE
                                                posX1 <= posX1 + Dx;
                                        END IF;
                                ELSE
                                        IF posX1  = left_edge_x1  THEN
                                                next_state1 <= board_row_5;
                                                posY1 <= posY1 + Dy;
                                        ELSE
                                                posX1 <= posX1 - Dx;
                                        END IF;
                                END IF;

------------------------------------------------------------------------------
                        WHEN board_row_5 =>

                                IF posX1  = right_edge_x1  THEN
                                                next_state1 <= board_row_6;
                                                posY1 <= posY1 + Dy;

                                ELSE
                                        posX1 <= posX1 + Dx;
                                END IF;
------------------------------------------------------------------------------
                        WHEN board_row_6 =>

                                IF posX1  = left_edge_x1  THEN
```

```
                                        next_state1 <= board_row_7;
                                        posY1 <= posY1 + Dy;
                        ELSE
                                posX1 <= posX1 - Dx;
                        END IF;
------------------------------------------------------------------------------
                WHEN board_row_7 =>
                        IF piece1_penalty THEN
                                IF posX1  = left_edge_x1  THEN
                                        next_state1 <= board_row_6;
                                        posY1 <= posY1 - Dy;

                                ELSE
                                        posX1 <= posX1 - Dx;
                                END IF;
                        ELSE

                                IF posX1  = right_edge_x1  THEN
                                        next_state1 <= board_row_8;
                                        posY1 <= posY1 + Dy;

                                ELSE
                                        posX1 <= posX1 + Dx;
                                END IF;
                        END IF;
------------------------------------------------------------------------------
                WHEN board_row_8 =>
                        IF piece1_penalty THEN
                                IF posX1  = right_edge_x1  THEN
                                        next_state1 <= board_row_7;
                                        posY1 <= posY1 - Dy;
                                ELSE
                                        posX1 <= posX1 + Dx;
                                END IF;
                        ELSE
                                IF posX1  = left_edge_x1  THEN
                                        next_state1 <= board_row_8;
                                ELSE
                                        posX1 <= posX1 - Dx;
                                END IF;

                        END IF;
-------------------------------------------------------------------------------
                WHEN OTHERS =>
                                null;
        END CASE;

    END IF;
END IF;

END process Generate_Newxy1;


-----------------------------------------------------------------------------
Generate_Newxy2: process(state2,clock)
BEGIN
IF clock'EVENT and clock = '1' THEN
        IF   VSync  AND ( (reset = '0') OR (pb2_state = move)) and piece2 THEN
                CASE state2 IS
                        WHEN initializing =>
                                posX2 <= startx2;
                                posY2 <= starty2;
                                next_state2 <= board_row_1;
                        WHEN board_row_1 =>
```

```
                posY2 <= starty2;
        IF posX2  = right_edge_x2  THEN
                                next_state2 <= board_row_2;
                                posY2 <= posY2 + Dy;
                ELSE
                        posX2 <= posX2 + Dx;
                END IF;

        WHEN board_row_2 =>
        IF posX2  = left_edge_x2  THEN

                                next_state2 <= board_row_3;
                                posY2 <= posY2 + Dy;

                ELSE
                        posX2 <= posX2 - Dx;
                END IF;

        WHEN board_row_3 =>

                IF piece2_penalty THEN
                        IF posX2  = left_edge_x2  THEN
                                next_state2 <= board_row_2;
                                posY2 <= posY2 - Dy;

                        ELSE
                                posX2 <= posX2 - Dx;
                        END IF;
                ELSE
                        IF posX2  = right_edge_x2  THEN
                                next_state2 <= board_row_4;
                                posY2 <= posY2 + Dy;
                        ELSE
                                posX2 <= posX2 + Dx;
                        END IF;
                END IF;


--------------------------------------------------------------------------
        WHEN board_row_4 =>
                IF piece2_penalty THEN
                        IF posX2  = right_edge_x2  THEN
                                next_state2 <= board_row_3;
                                posY2 <= posY2 - Dy;
                        ELSE
                                posX2 <= posX2 + Dx;
                        END IF;
                ELSE
                        IF posX2  = left_edge_x2  THEN
                                next_state2 <= board_row_5;
                                posY2 <= posY2 + Dy;
                        ELSE
                                posX2 <= posX2 - Dx;
                        END IF;
                END IF;

--------------------------------------------------------------------------

        WHEN board_row_5 =>
        IF posX2  = right_edge_x2  THEN

                                next_state2 <= board_row_6;
                                posY2 <= posY2 + Dy;
```

```
                                ELSE
                                        posX2 <= posX2 + Dx;
                                END IF;

                        WHEN board_row_6 =>
                        IF posX2  = left_edge_x2  THEN
                                                next_state2 <= board_row_7;
                                                posY2 <= posY2 + Dy;

                                ELSE
                                        posX2 <= posX2 - Dx;
                                END IF;

------------------------------------------------------------------------
                        WHEN board_row_7 =>
                                IF piece2_penalty THEN
                                        IF posX2  = left_edge_x2  THEN
                                                next_state2 <= board_row_6;
                                                posY2 <= posY2 - Dy;

                                        ELSE
                                                posX2 <= posX2 - Dx;
                                        END IF;
                                ELSE

                                        IF posX2  = right_edge_x2  THEN
                                                next_state2 <= board_row_8;
                                                posY2 <= posY2 + Dy;

                                        ELSE
                                                posX2 <= posX2 + Dx;
                                        END IF;
                                END IF;
----------------------------------------------------------------------------
                        WHEN board_row_8 =>
                                IF piece2_penalty THEN
                                        IF posX2  = right_edge_x2  THEN
                                                next_state2 <= board_row_7;
                                                posY2 <= posY2 - Dy;
                                        ELSE
                                                posX2 <= posX2 + Dx;
                                        END IF;
                                ELSE
                                        IF posX2  = left_edge_x2  THEN
                                                next_state2 <= board_row_8;
                                        ELSE
                                                posX2 <= posX2 - Dx;
                                        END IF;

                                END IF;
-----------------------------------------------------------------------------
                        WHEN OTHERS =>
                                        null;
                END CASE;

        END IF;
END IF;

END process Generate_Newxy2;

----------------------------------------------------------------------------------------
piece1_display : process(clock)
BEGIN
```

```
IF clock'EVENT and clock = '1' THEN
        equal_x1 <= (countX  >= posX1 ) and (countX < (posX1 + "10000"))
                    and ((countX >= left_edge_x1) and (countX <= panel_right_x))
        equal_y1 <= ((countY and "1111110000") = posY1);
-- piece2's position when piece 2 falls in stop block , x= 224, y = 352;
                stop_position_p1 <= ((posX1(9 downto 4) = second_column_x1)
                and (posY1(9 downto 4) = seventh_row_y1)) or
                ((posX1(9 downto 4) = seventh_column_x1) and
                (posY1(9 downto 4) = third_row_y1));
-- piece2's position when piece 2 falls in bonus block , x= 256, y = 192;
                bonus_position_p1 <= ((posX1(9 downto 4) = third_column_x1)
                and (posY1(9 downto 4) = second_row_y1)) or
                ((posX1(9 downto 4) = eighth_column_x1) and
                (posY1(9 downto 4) = sixth_row_y1));
-- piece2's position when piece 2 falls in penalty block , x= 352, y = 384;
                penalty_position_p1 <= ((posX1(9 downto 4) = sixth_column_x1)
                and (posY1(9 downto 4) = eighth_row_y1)) or
                ((posX1(9 downto 4) = fifth_column_x1) and
                (posY1(9 downto 4) = fourth_row_y1));


                piece1_vga_red <= (equal_x1 and equal_y1);
                piece1_vga_green <= false;
                piece1_vga_blue  <= false;
END IF;
END process piece1_display;
-------------------------------------------------------------------------------
piece2_display : process(clock)
BEGIN

IF clock'EVENT and clock = '1' THEN
        equal_x2 <= (countX  >= posX2 ) and (countX < (posX2 + "10000"))
                     and ((countX >= left_edge_x2) and
                     (countX <=panel_right_x));
        equal_y2 <= ((countY and "1111110000") = posY2);
-- piece2's position when piece 2 falls in stop block , x= 224+16,
-- y = 352+16;

        stop_position_p2 <= ((posX2(9 downto 4) = second_column_x2) and
                (posY2(9 downto 4) = seventh_row_y2)) or
                ((posX2(9 downto 4) = seventh_column_x2) and
                (posY2(9 downto 4) = third_row_y2));

-- piece2's position when piece 2 falls in bonus block , x= 256+16, y = 192+16;

        bonus_position_p2 <= ((posX2(9 downto 4) = third_column_x2) and
                (posY2(9 downto 4) = second_row_y2)) or
                ((posX2(9 downto 4) = eighth_column_x2) and
                (posY2(9 downto 4) = sixth_row_y2));

-- piece2's position when piece 2 falls in penalty block , x= 352+16,
-- y = 384+16;

        penalty_position_p2 <= ((posX2(9 downto 4) = sixth_column_x2) and
                (posY2(9 downto 4) = eighth_row_y2)) or
                ((posX2(9 downto 4) = fifth_column_x2) and
                (posY2(9 downto 4) = fourth_row_y2));


                piece2_vga_red <= false;
                piece2_vga_green <= (equal_x2 and equal_y2);
                piece2_vga_blue  <= false;

END IF;
```

```
END process piece2_display;

                vga_red    <=  piece1_vga_red   or piece2_vga_red;
                vga_green  <=  piece1_vga_green or piece2_vga_green;
                vga_blue   <=  piece1_vga_blue  or piece2_vga_blue;

END background_dice;
```