```
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 dicerace IS
        PORT(
        clock                           : in STD_LOGIC;
        reset                           : in STD_LOGIC;
        PB2                             : in std_logic;
        -- player dice button
        vga_h_sync,vga_v_sync           : out BOOLEAN;
        vga_red, vga_green, vga_blue    : out BOOLEAN
        );
END dicerace;

ARCHITECTURE structure OF dicerace IS

SIGNAL  back_vga_red,  back_vga_green,  back_vga_blue           : BOOLEAN;
SIGNAL  dice_vga_red,  dice_vga_green,  dice_vga_blue           : BOOLEAN;
SIGNAL  message_vga_red,  message_vga_green,  message_vga_blue  : BOOLEAN;
SIGNAL  play_vga_red,  play_vga_green,  play_vga_blue           : BOOLEAN;


SIGNAL  HSync, VSync                    :  BOOLEAN;
SIGNAL  countX, countY                  :  STD_LOGIC_VECTOR(9 downto 0);
SIGNAL  piece1_wins, piece2_wins        :  BOOLEAN;
SIGNAL  piece1, piece2                  :  BOOLEAN;
SIGNAL  step_size                       :  STD_LOGIC_VECTOR( 2 downto 0);
SIGNAL  bonus, stop, penalty            :  BOOLEAN;
SIGNAL  vclock                          :  STD_LOGIC;

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

COMPONENT message
        PORT(
        clock                           : in STD_LOGIC;
        reset                           : in STD_LOGIC;
        piece1_wins, piece2_wins        : in BOOLEAN;
        piece1, piece2                  : in boolean;
        vga_red, vga_green, vga_blue    : out BOOLEAN;
        countX, countY                  : in STD_LOGIC_VECTOR(9 downto 0);
        step_size                       : in STD_LOGIC_VECTOR(2 downto 0);
        vclock                          : in STD_LOGIC
        );

END COMPONENT;

COMPONENT 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 COMPONENT;



COMPONENT backgrnd
        PORT(
        clock                           : in STD_LOGIC;
        countX, countY                  : in STD_LOGIC_VECTOR(9 downto 0);
        vga_red, vga_green, vga_blue    : out BOOLEAN
        );
END COMPONENT;


COMPONENT SYNCGEN
        PORT
        (
                Clock                   : IN    std_logic;
                vclock                  : OUT   STD_LOGIC;
                HSync, VSync            : OUT   BOOLEAN;
                vga_h_sync, vga_v_sync  : OUT   BOOLEAN
        );
END COMPONENT;

component random

        PORT (
        clock           : IN STD_LOGIC;
        PB2             : IN STD_LOGIC;
        VSync           : IN BOOLEAN;
        step_size       : OUT STD_LOGIC_VECTOR(2 downto 0)
        );
END component;

COMPONENT count_xy
        PORT(
        clock                       : in STD_LOGIC;
        countX,countY               : out STD_LOGIC_VECTOR(9 downto 0);
        HSync,VSync                 : in BOOLEAN
        );
END COMPONENT;

BEGIN

use_syncgen: SYNCGEN port map(
                Clock  => clock,
                vclock => vclock,
                HSync  => HSync,
                VSync  => VSync,
            vga_h_sync =>  vga_h_sync,
            vga_v_sync =>  vga_v_sync
           );

use_background: backgrnd PORT MAP(
                        clock      => clock,
```

```
                        countX     => countX,
                        countY     => countY,
                        vga_red    =>  back_vga_red,
                        vga_green  =>  back_vga_green,
                        vga_blue   =>  back_vga_blue
                        );

use_counter: count_xy PORT MAP(
        clock  => clock,
                HSync  => HSync,
                VSync  => VSync,
                countX => countX,
                countY => countY
);

use_ramdom : random port map(
                        clock     => clock,
                        PB2       => PB2,
                        VSync     => VSync,
                        step_size => step_size );

use_dice : dice port map(
                        clock       => clock,
                        HSync       => HSync,
                        VSync       => VSync,
                        countX      => countX,
                        countY      => countY,
                        step_size   => step_size,
                        reset       => reset,
                        PB2         => PB2,
                        piece1_wins => piece1_wins,
                        piece2_wins => piece2_wins,
                        stop        => stop,
                        bonus       => bonus,
                        penalty     => penalty,
                        piece1      => piece1,
                        piece2      => piece2,
                        vga_red     => dice_vga_red,
                        vga_green   => dice_vga_green,
                        vga_blue    => dice_vga_blue );

use_message: message port map (
                        clock       => clock,
                        vclock      => vclock,
                        countX      => countX,
                        countY      => countY,
                        step_size   => step_size,
                        reset       => reset,
                        piece1_wins => piece1_wins,
                        piece2_wins => piece2_wins,
                        piece1      => piece1,
                        piece2      => piece2,
                        vga_red     => message_vga_red,
                        vga_green   => message_vga_green,
                        vga_blue    => message_vga_blue  );
use_play_ms: play_ms port map (
                        clock       => clock,
                        vclock      => vclock,
                        countX      => countX,
                        countY      => countY,
                        step_size   => step_size,
                        reset       => reset,
                        piece1_wins => piece1_wins,
                        piece2_wins => piece2_wins,
```

```
                        piece1      => piece1,
                        piece2      => piece2,
                        vga_red     => play_vga_red,
                        vga_green   => play_vga_green,
                        vga_blue    => play_vga_blue  );


vga_red    <=  ((not dice_vga_green) and (dice_vga_red or back_vga_red)) or
                message_vga_red or play_vga_red;
vga_green  <=  ((not dice_vga_red) and (dice_vga_green or back_vga_green)) or
                message_vga_green or play_vga_green;
vga_blue   <=  (dice_vga_red or dice_vga_green or back_vga_blue) or
                message_vga_blue or play_vga_blue;


END structure;
```