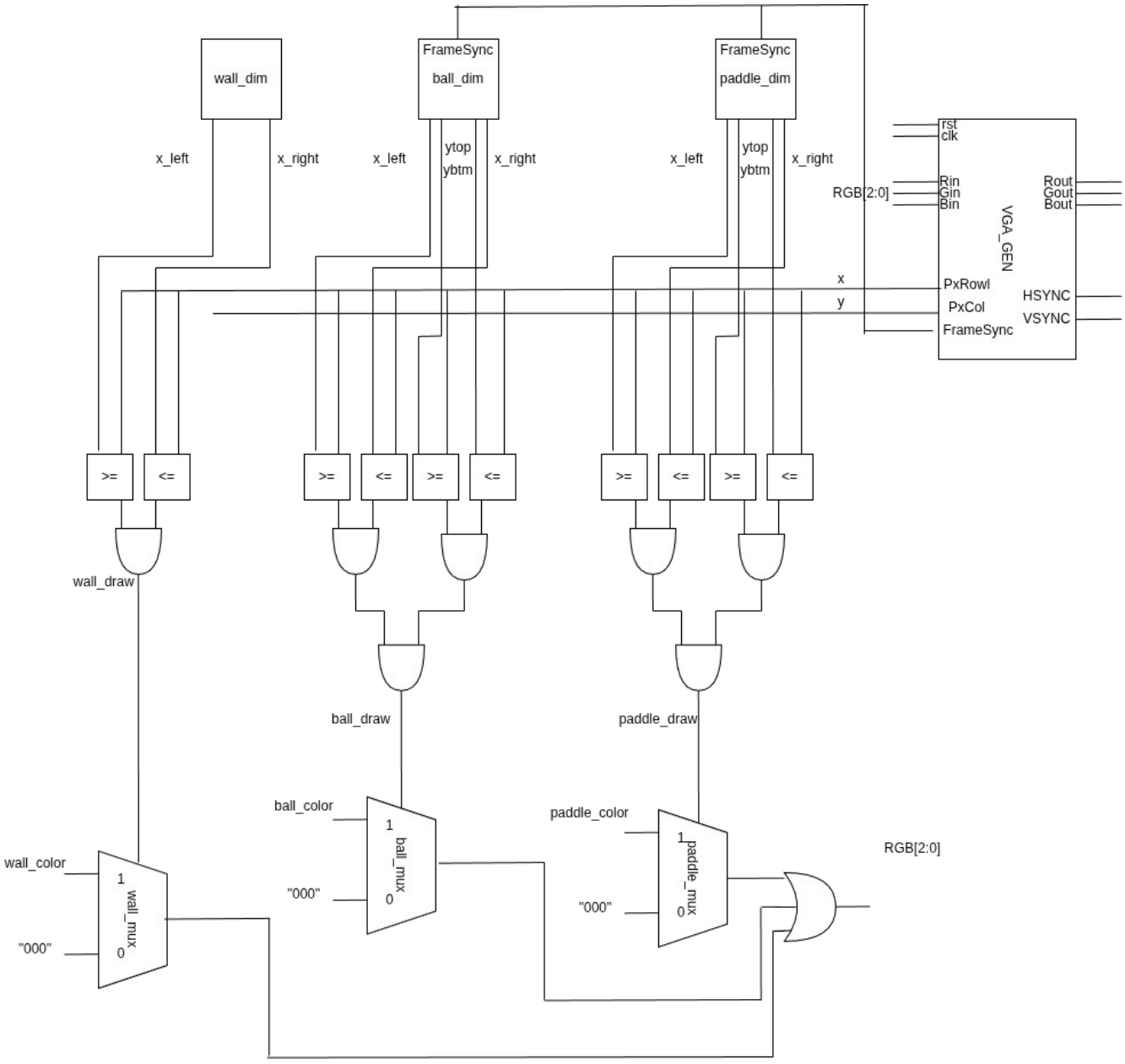
wall_dim
FrameSync
ball_dim
FrameSync
paddle_dim
x_left
x_right
x_left
ytop
ybtm
x_right
x_left
ytop
ybtm
x_right
rst
clk
RGB[2:0]
Rin
Gin
Bin
Rout
Gout
Bout
VGA_GEN
x
y
PxRowl
PxCol
FrameSync
HSYNC
VSYNC
>=
<=
>=
<=
>=
<=
>=
<=
wall_draw
ball_draw
paddle_draw
ball_color
paddle_color
wall_color
1
wall_mux
0
"000"
1
ball_mux
0
"000"
1
paddle_mux
0
"000"
RGB[2:0]

Mogla je da se izvrsi optimizacija podelom operatora, ali u svrhu modulranosti projekta, optimizacija nije bila izvrsena. Detektciija kolizije uradjena je na osnovu koordinata strana loptice. Dok je njeno odbijanje vrseno pomocu promene koeficijenta K u zavisnosti od toga da li je naisla na zid, ivicu ekrana ili reket.

```
entity pong is
    Port ( rst : in STD_LOGIC;
           clk : in STD_LOGIC;
           up : in STD_LOGIC;
           down : in STD_LOGIC;
           VGA_HSYNC : out STD_LOGIC;
           VGA_VSYNC : out STD_LOGIC;
           VGA_RED : out STD_LOGIC;
           VGA_GREEN : out STD_LOGIC;
           VGA_BLUE : out STD_LOGIC
         );
end pong;

architecture Behavioral of pong is
  component VGA_GEN is
    Port ( rst : in STD_LOGIC;
           clk : in STD_LOGIC;
           Rin : in STD_LOGIC;
           Gin : in STD_LOGIC;
           Bin : in STD_LOGIC;

           PxRow : out STD_LOGIC_VECTOR (9 downto 0);
           PxCol : out STD_LOGIC_VECTOR (9 downto 0);
           PxDisp : out STD_LOGIC;
           PxClk : out STD_LOGIC;
           LnClk : out STD_LOGIC;
           FrameSync: out STD_LOGIC;
           VGA_HSYNC : out STD_LOGIC;
           VGA_VSYNC : out STD_LOGIC;
```

```
        VGA_RED : out STD_LOGIC;
        VGA_GREEN : out STD_LOGIC;
        VGA_BLUE : out STD_LOGIC);
  end component;
  --pozicije objekata ----------------------------------
  ----ZIIIIIDDDDD-------------------------------------------
  constant wall_x_l: integer :=40;
  constant wall_x_r: integer :=50;
  constant wall_color: std_logic_vector(2 downto 0):="111";
  signal wall_draw:std_logic;
  --REKETTT ----------------------------------------
  --constant paddle_top_y : integer:=360;
 -- constant paddle_bottom_y : integer :=230;
  constant paddle_left_x:integer:=500;
  constant paddle_right_x:integer:=510;
  constant paddle_color: std_logic_vector(2 downto 0):="111";
  signal paddle_draw:std_logic;
  --BALLLLL----------------------------------------
  --constant ball_top_y : integer :=310;
  --constant ball_bottom_y :integer :=300;
 -- constant ball_left_x :integer :=300;
 -- constant ball_right_x: integer :=310;
  constant ball_color: std_logic_vector(2 downto 0):="010";
  signal ball_draw:std_logic;
  ----------------------------singal------------------------
  signal RGB: std_logic_vector(2 downto 0):="000";
  signal FrameSync: std_logic;
  signal x: integer range 0 to 640;
  signal y :integer range 0 to 480;
  signal PxRow,PxCol : std_logic_vector(9 downto 0);
  ---movement_signals----
 signal paddle_top_y : integer:=300;
 signal paddle_bottom_y : integer :=230;

signal ball_top_y : integer :=240;
signal ball_bottom_y :integer :=230;
signal ball_left_x :integer :=300;
signal ball_right_x: integer :=310;
------------------------
signal dir,paddle_hit,wall_hit,paddle_top,paddle_bottom: std_logic :='0';
signal  K: integer:=1;

begin
  vga_sin: VGA_GEN  port map (rst=>rst,clk=>clk,
                        Rin=>RGB(2),Gin=>RGB(1),Bin=>RGB(0),
                        VGA_HSYNC=>VGA_HSYNC,VGA_VSYNC=>VGA_VSYNC,

VGA_GREEN=>VGA_GREEN,VGA_RED=>VGA_RED,VGA_BLUE=>VGA_BLUE,
                        FrameSync=>FrameSync,PxCol=>PxCol,PxRow=>PxRow,
```

```
                        PxClk=>open,LnClk=>open,PxDisp=>open);
  x<=to_integer(unsigned(pxCol));
  y<=to_integer(unsigned(pxRow));
  -----pokusaj generisanja slika------
  wall_draw<='1' when (x>=wall_x_l and x<=wall_x_r) else '0';
  paddle_draw<='1' when (x>=paddle_left_x and x<=paddle_right_x) and (y>=paddle_bottom_y and
y<=paddle_top_y) else '0';
  ball_draw<='1' when (x>=ball_left_x and x<=ball_right_x) and (y>=ball_bottom_y and
y<=ball_top_y) else '0';

  RGB<=wall_color when (wall_draw='1' and wall_hit='0') else
       "100" when (wall_draw='1' and wall_hit='1') else
       ball_color when ball_draw='1' else
       paddle_color when paddle_draw='1' else
       "000";
  -----------PADDLE_MOVEMENT-----------------
  process(FrameSync)
   begin
    if(rising_edge(FrameSync)) then
      if(up='1' and paddle_bottom_y>=10) then
        paddle_bottom_y<=paddle_bottom_y-3;
        paddle_top_y<=paddle_top_y-3;
      elsif(down='1' and paddle_top_y<=470) then
        paddle_bottom_y<=paddle_bottom_y+3;
        paddle_top_y<=paddle_top_y+3;
      end if;
    end if;
  end process;
----BALL_MOVEMENTS-------------------------
  process(FrameSync)
    begin
      if(rising_edge(FrameSync)) then
        if(ball_top_y<=479 and ball_bottom_y>=5) then
          if(K=1) then
            ball_top_y<=ball_top_y-1;
            ball_bottom_y<=ball_bottom_y-1;
         elsif(K=-1) then
            ball_top_y<=ball_top_y+1;
            ball_bottom_y<=ball_bottom_y+1;
          end if;
      end if;
      if(dir='1') then --needs to go left
         ball_left_x<=ball_left_x-1;
         ball_right_x<=ball_right_x-1;
       else --needs to go right
        ball_left_x<=ball_left_x+1;
        ball_right_x<=ball_right_x+1;
       end if;
      if(paddle_hit='1' or paddle_bottom='1' or paddle_top='1') then --paddle hit ide levo
```

```
        dir<='1';
        K<=-K;
       ------------------------------------------------------------
      elsif(wall_hit='1') then --wall hit y=kx+n -- ide desno
         dir<='0';
       ------------------------------------------------------------
      elsif(ball_bottom_y=10) then --bottom screen hit
          K<=-1;
      elsif(ball_top_y=470) then
          K<=+1;
      elsif(ball_right_x>=630) then --respawn
        ball_right_x<=210;
        ball_left_x<=200;
        ball_top_y<=240;
        ball_bottom_y<=230;
        dir<='0';
      end if;
    end if;
  end process;
  --paddle_lenght<=to_unsigned(paddle_top_y - paddle_bottom_y,10);
  paddle_hit<='1' when (ball_right_x=paddle_left_x-1 and (ball_top_y>paddle_bottom_y and
ball_bottom_y<paddle_top_y)) else '0';
  wall_hit<='1' when ball_left_x=wall_x_r+1 else '0';
  paddle_top<='1' when (ball_top_y=paddle_bottom_y and ball_right_x>paddle_left_x and
ball_right_x<paddle_right_x) else '0';
  paddle_bottom<='1' when (ball_bottom_y=paddle_bottom_y and ball_right_x>paddle_left_x and
ball_right_x<paddle_right_x) else '0';


end Behavioral;
```

Resource	Utilization	Available	Utilization %
LUT	795	20800	3.82
FF	247	41600	0.59
IO	9	106	8.49