Universidad Nacional Autónoma de México

Facultad de Ingeniería

Diseño Digital Moderno VLSI

Nombre del Profesor:

M.I. Rafael Prieto Meléndez

Semestre 2023-1

Grupo 3

Proyecto Final - "Ping Pong"

Nombre del alumno:

- Castelan Ramos Carlos
- Torres Martínez Marco Antonio

Fecha de entrega: 10/01/2023

ÍNDICE

ÍNDICE **1**

Descripción del proyecto. **2**

Análisis básico. **2**

Aclaraciones previas: 2
Reloj CLK: 3
Controles de las barras: 4
Control Pelota: 4
Control Marcador: 5
Control Codificador 7 Segmentos (Marcador): 6
Control Reloj: 6
Control Codificador 7 Segmentos (Reloj-Cronómetro): 7
Sincronizador VGA: 8
Generado Imagen: 8
VGA: 9

Diseño conceptual. **10**

Implementación del código. **13**

Módulo Reloj CLK: 13
Módulo Principal (Pong): 14
Control Reloj - Cronómetro: 19
Control Barras: 20
Control Pelota: 22
Control Máquina de Estados: 24
Control de Marcadores: 28

Conclusión general del proyecto. **30**

Proyecto Final “Ping Pong”

Descripción del proyecto.

Se desarrolló el juego de Ping Pong en 2D haciendo uso del lenguaje VHDL, el cual contiene distintos elementos para su jugabilidad, los cuales son:

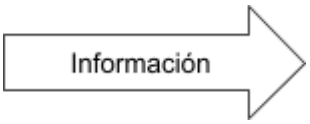
- Salida de imagen en monitor VGA:
    - Cancha del juego.
    - Paletas de los jugadores (2).
    - Pelota.
- Cronómetro: Se muestra en los displays de 7 segmentos que la tarjeta DE10-Lite tiene, este cronómetro tiene segundos y minutos, además se reinicia cada que un jugador hace una anotación.
- Marcador: Se muestra en los displays de 7 segmentos que la tarjeta DE10-Lite tiene. Marca el puntaje de cada jugador por cada anotación hecha.
- Controles del jugador: Para controlar las paletas del jugador, se hace uso de pushbuttons.

Análisis básico.

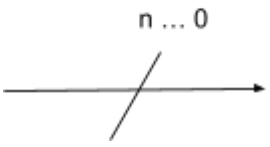
Aclaraciones previas:

- Representa entradas o salidas en bits.

- Representa entradas o salida de información de un bloque o el mismo bloque.



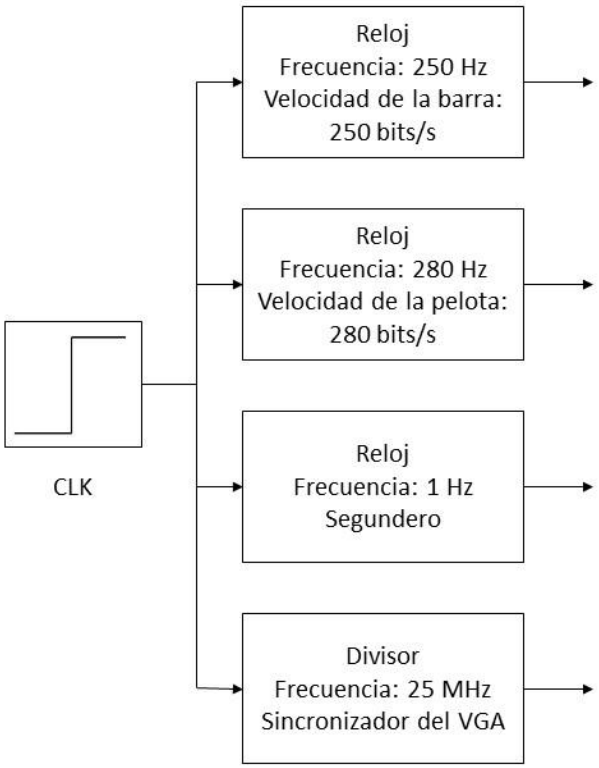
- Representa la entrada o salida en forma de un vector.



Reloj CLK:

Tenemos el bloque reloj CLK, el cual funciona como un multiplicador de periodo y a su vez un divisor de frecuencias, el cual a partir de la manipulación de distintas frecuencias este funcionará como entrada para distintos bloques o módulos del proyecto.

- Reloj barras de jugadores: Indicará la velocidad en la cual se podrán desplazar las barras(paletas) de los jugadores.
- Reloj pelota: Indicará la velocidad en la cual se desplazará la pelota a lo largo de la cancha de juego.
- Reloj segundero: Indicará la velocidad en la cual el segundero del cronómetro irá incrementando.
- Divisor VGA: Obtendrá la frecuencia para la generación de imagen VGA.

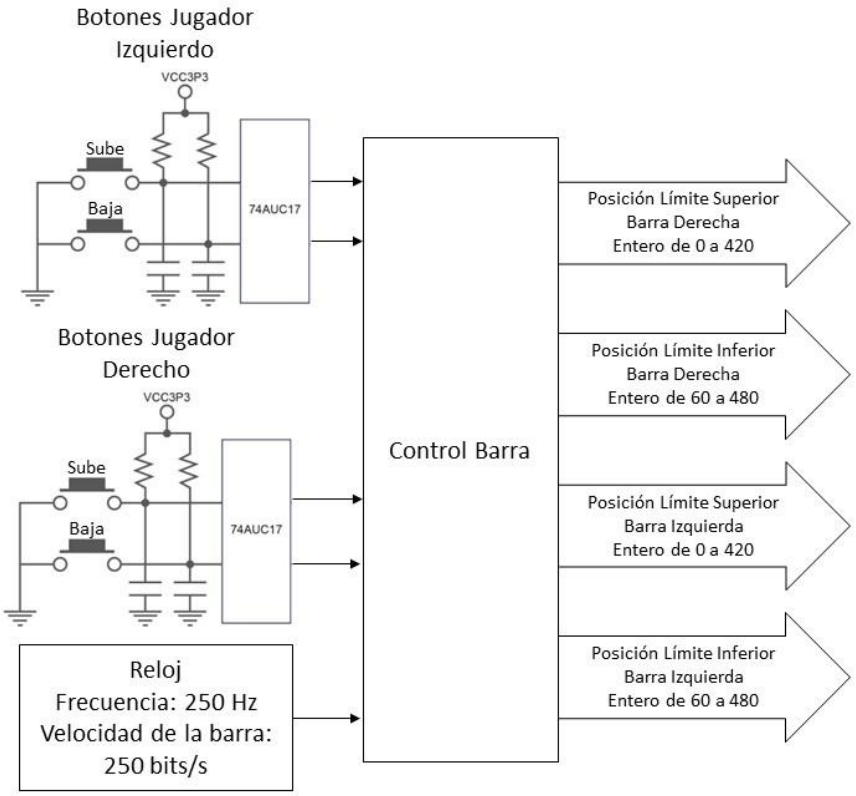


Controles de las barras:

Tenemos el bloque Control Barra, el cuál, recibirá 5 entradas: la pulsación de pushbuttons (sube y baja) de ambos jugadores (izquierdo y derecho) y el reloj de la velocidad de la barra.

Obtendremos 4 salidas, que son los límites inferior y superior de las barras de ambos jugadores, estos están delimitados mediante el manejo de píxeles del monitor VGA.

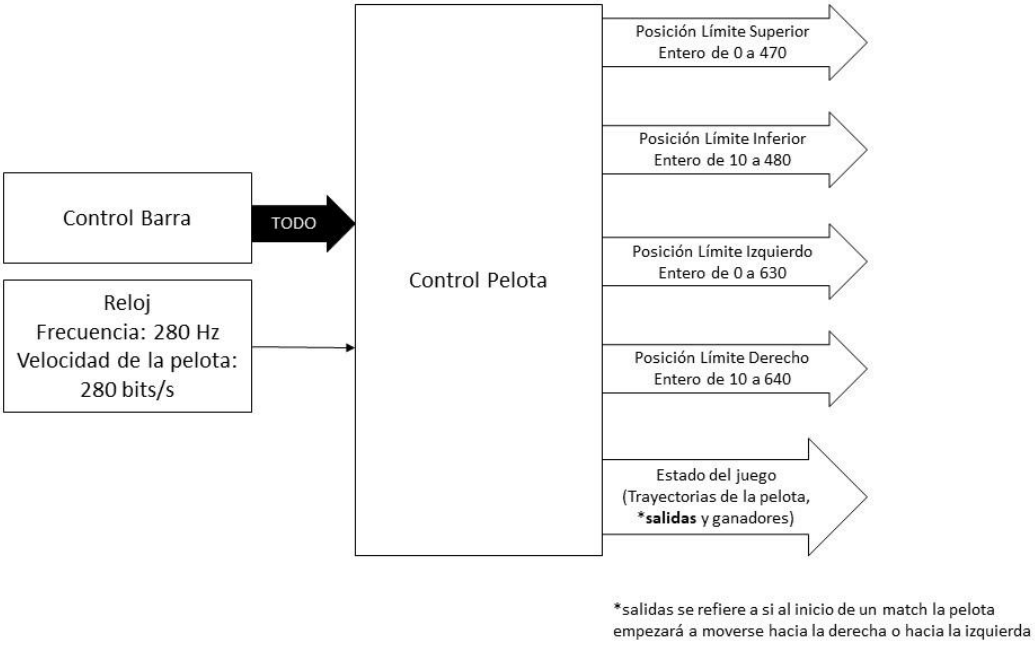
Estas posiciones fueron definidas al único movimiento que tendrán las barras (vertical).



Control Pelota:

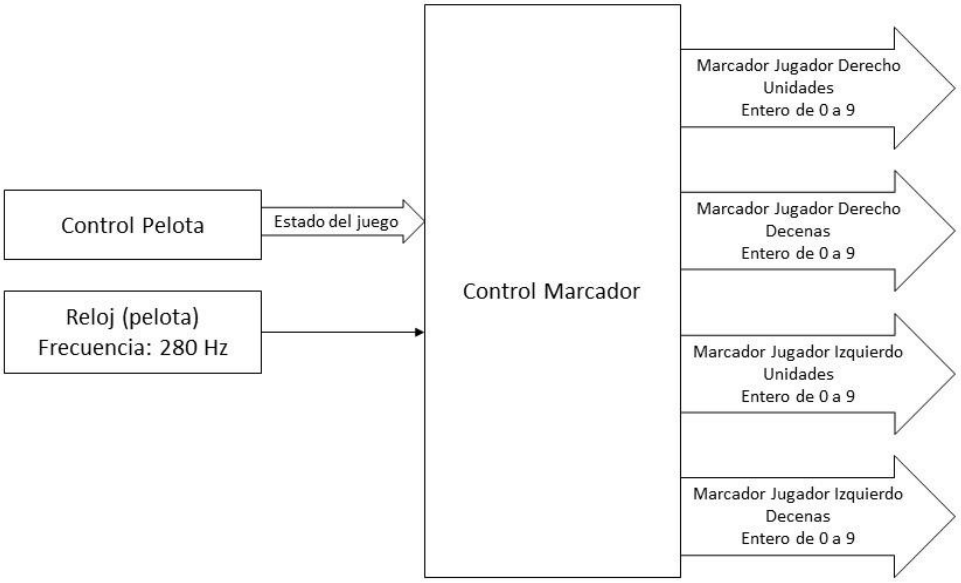
Tenemos el bloque control pelota, en donde dadas las 4 salidas de posicionamiento de las barras de ambos jugadores y además la velocidad de la pelota tenemos las entradas. Obtendremos como salida, todas las posibles posiciones en la que la pelota puede estar, a diferencia de las barras que se movían únicamente horizontalmente(1 eje), la pelota puede moverse en todas

direcciones de la cancha (2 ejes). Además existe una salida de estado de la pelota en el juego, que indica la aparición inicial de la pelota en el juego ó la trayectoria de la pelota ó el estado del jugador que hizo un punto por anotar la pelota.



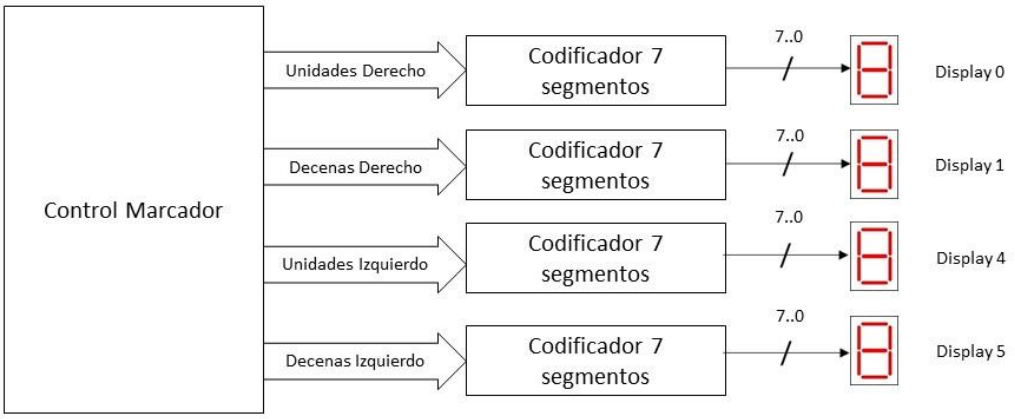
Control Marcador:

Tenemos el bloque Control Marcador, el cual como entrada tiene a las salidas del bloque de Control Pelota, especialmente nos interesa el estado de la pelota en la cancha de juego, para indicar si esta está iniciando un nuevo juego, el juego está transcurriendo o se acaba de hacer una anotación, en conjunto con el reloj de la pelota, estaremos actualizando la información de estos estados con la intención de conservar o mantener el marcador de los jugadores, por lo que a partir de estos, tendremos el marcador en unidades y decenas para ambos jugadores, siendo las 4 salidas del bloque.



Control Codificador 7 Segmentos (Marcador):

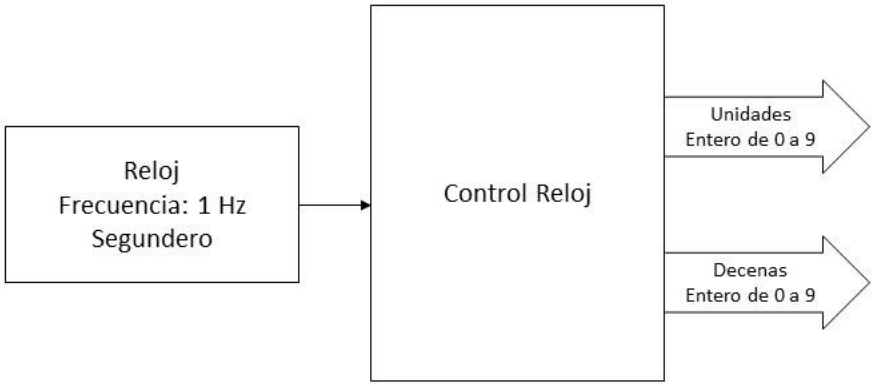
Existe el bloque de control de un codificador de 7 segmentos, el cuál recibe como entrada las salidas del bloque del Control Marcador, ya que la información de estas salidas será la información decodificada de enteros a binarios para unidades y decenas del marcador de cada jugador, pasando esta información como vectores de magnitud 7 a una salida física en displays de 7 segmentos.



Control Reloj:

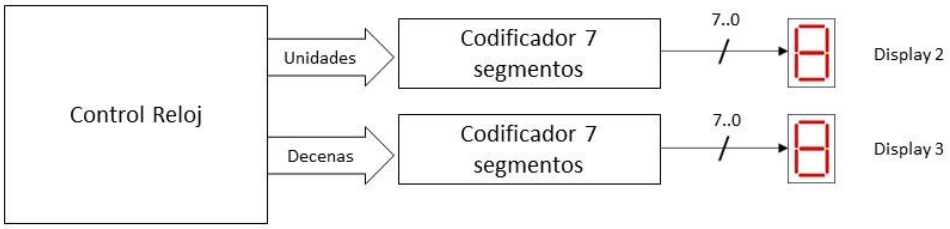
Haciendo uso del bloque Reloj CLK como entrada, tenemos al bloque Control Reloj, el cual funcionará como cronómetro del juego, en donde este iniciará cada que la pelota esté en un estado de inicio y se reiniciará cada que un

jugador realice una anotación. Las salidas serán las unidades y decenas del cronómetro en información entera decimal.



Control Codificador 7 Segmentos (Reloj-Cronómetro):

El principio de funcionamiento de este bloque es el mismo que el presentado en el codificador de 7 segmentos presentado para el marcador, a excepción de que varía en el número de entradas y el números de salidas. Como entradas tendremos las salidas de información enteras decimales del bloque Control Reloj, la cual pasarán por el control codificador para obtener la salida binaria de estos y así tengan su representación en los displays de 7 segmentos.



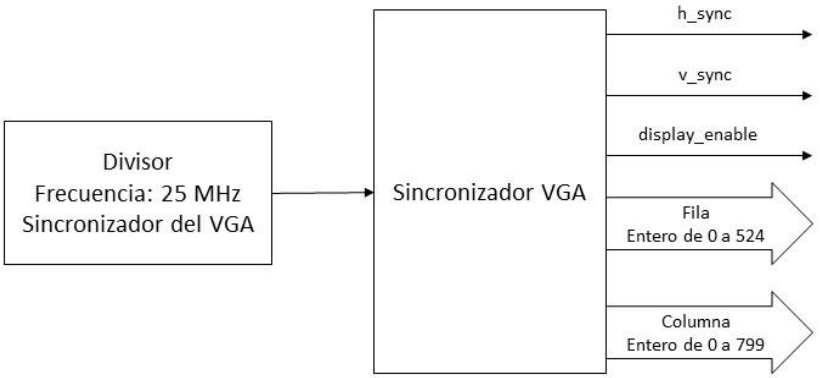
A partir de los dos controles codificadores mostrados, observamos que la distribución de displays es la siguiente:

- Display 0 - Marcador Unidades Jugador Derecho
- Display 1 - Marcador Decenas Jugador Derecho
- Display 2 - Unidades Reloj
- Display 3 - Decenas Reloj

- Display 4 - Marcador Unidades Jugador Izquierdo
- Display 5 - Marcador Decenas Jugador Izquierdo

Sincronizador VGA:

Este bloque es el mismo utilizado en las prácticas de la materia. Como entrada al bloque de Sincronizador VGA, tenemos al divisor de frecuencia de 25 MHz generado en el bloque Reloj CLK, como salida obtendremos el sincronizador horizontal, sincronizador vertical, display_enable(Para comprobar que estamos dentro del área visible del monitor), información de filas y columnas para el manejo de píxeles.



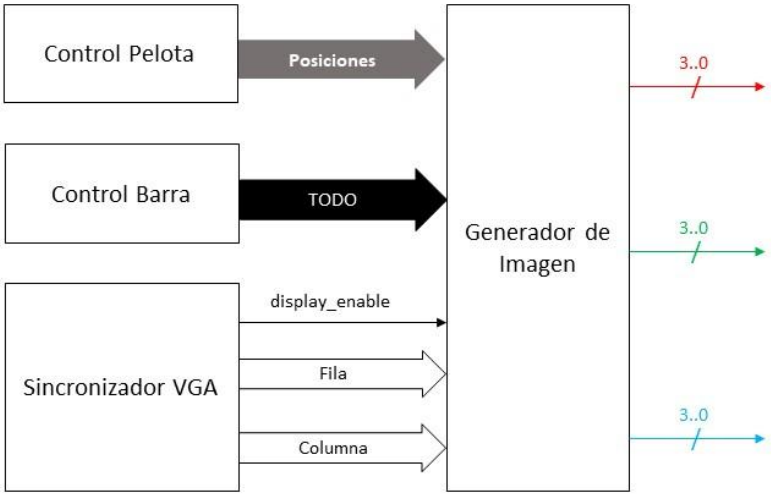
Generado Imagen:

Este bloque es muy importante, porque presenta toda la recopilación de datos que el usuario podrá observar, por ende todas las funciones presentadas irán de la mano con las entradas de este bloque.

Entradas:

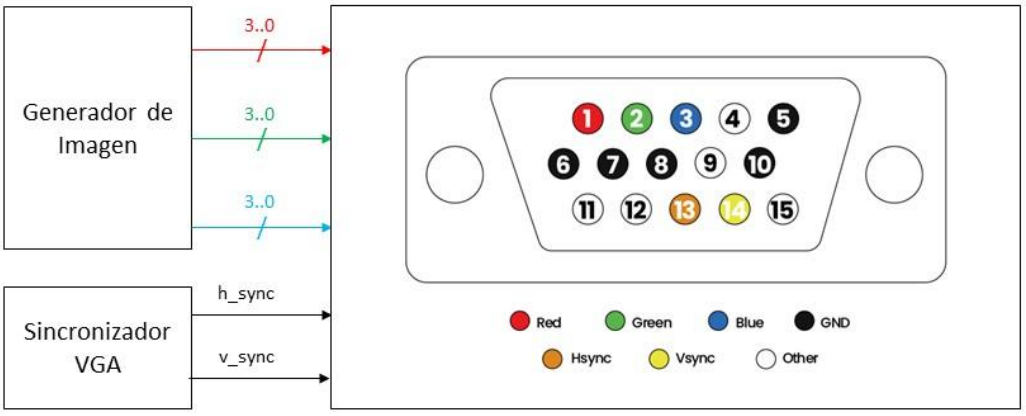
- Control pelota: De este bloque obtendremos sus salidas, las cuales son todas las posibles posiciones en las cuales la pelota puede aparecer en la cancha
- Control barra: De este bloque obtendremos la posición vertical del movimiento de las barras de ambos jugadores.
- Sincronizador VGA: Obtendremos los datos del aŕea disponible del monitor, además del uso de píxeles para la generación de imágen.

Como salidas tenemos el manejo de colores de la imagen de salida en RGB del monitor VGA.



VGA:

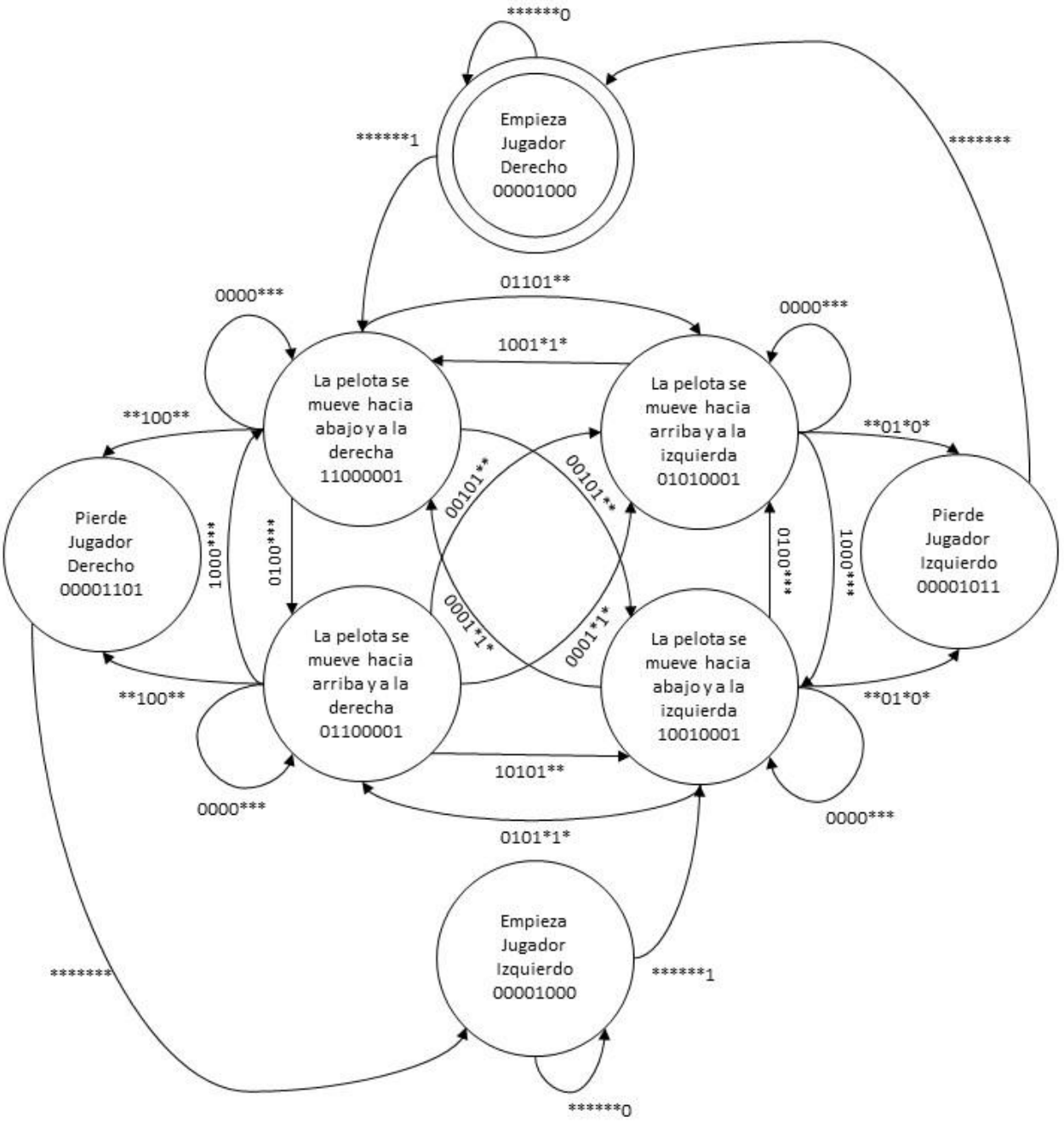
Este bloque es el bloque final para la muestra del funcionamiento de la imagen generada, como entrada tendremos la salida del generador de imagen, que recordemos incorpora la información del control de la pelota y la barra, así como del sincronizador VGA que nos genera la imagen de la cancha, barras y pelota en vectores para la emisión de colores RGB mediante la sincronización horizontal y vertical.



Diseño conceptual.

Las entradas son condiciones que codificaremos de la siguiente manera						
La pelota llega al límite superior de la pantalla	La pelota llega al límite inferior de la pantalla	La pelota llega a la posición donde empieza la barra derecha	La pelota llega a la posición donde empieza la barra izquierda	La barra derecha se encuentra en posición para generar un rebote con la pelota	La barra izquierda se encuentra en posición para generar un rebote con la pelota	El tiempo de la partida se ha reiniciado
•	•	•	•	•	•	*

Las salidas son más bien acciones que se deben realizar y son							
Sumar a los límites superior en inferior de la pelota	Sumar a los límites laterales de la pelota	Restar a los límites superior en inferior de la pelota	Restar a los límites laterales de la pelota	Regresar la pelota a su posición de salida (arriba y al centro horizontal)	Sumar al marcador del jugador derecho	Sumar al marcador del jugador izquierdo	Reiniciar (0) o dejar correr (1) el tiempo de la partida
•	•	•	•	•	•	•	*

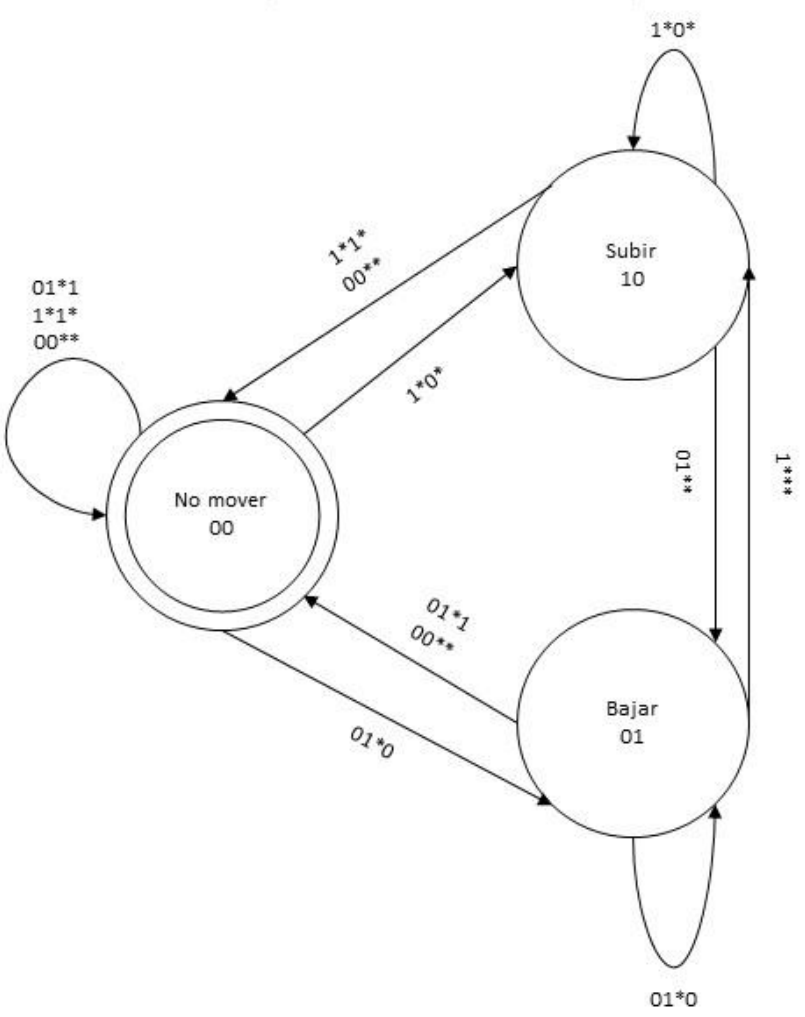


Todo esto a reserva de las siguientes condiciones imposibles	
Alcanzar al mismo tiempo el límite superior e inferior	11****
Alcanzar al mismo tiempo los límites derecho a izquierdo	**11**
Para las barras si se pueden encontrar ambas a una misma altura (posición vertical) que genera un rebote, pero solo importa la que se encuentra en la trayectoria de la pelota, pero esto no es como tal una condición imposible, simplemente no importa la posición de la otra barra.	

La máquina de estados controla directamente la trayectoria de la pelota, sin embargo, indirectamente también controla el marcador y hasta el reloj, ya que si un jugador pierde o no, depende de la posición de la pelota, lo que afecta al marcador y cómo el reloj se reinicia cada que un jugador pierde, también termina dependiendo de la pelota, aún cuando este tenga su propio reloj a una distinta frecuencia.

Entradas y condiciones del control de la barra (para cada barra)			
Boton subir	Botón bajar	Alcanzó el límite superior	Alcanzó el límite inferior
*	*	*	*

Salidas para el control de la barra (Acciones)	
Restar a las posiciones de los limites superior e inferior	Sumar a las posiciones de los limites superior e inferior
*	*



Se consideran condiciones imposibles de que la barra alcance los límites superior e inferior al mismo tiempo: **11.

E igualmente, proviniendo del estado “Bajar” es imposible que la barra haya alcanzado el “Límite Superior”, e igualmente es imposible que proviniendo del estado “Subir” se haya alcanzado el límite superior.

Implementación del código.

Módulo Reloj CLK:

De acuerdo a lo visto en el Análisis Básico, este módulo nos ayudará a generar distintas frecuencias para el manejo de las velocidades de otros módulos o bloques funcionales del proyecto, los cuales son la velocidad de la barra (para ambos jugadores), velocidad de la pelota, reloj o cronómetro del juego y el divisor de frecuencia para el Sincronizador VGA.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

Entity rel is
   Generic ( lim : integer := 25000000);
   Port (
       clk     : in std_logic;
       clk_esc : out std_logic
   );
End rel;

Architecture Behavioral of rel is

   signal count : integer range 0 to (lim * 2) := 0;

begin

   process(clk, count)
   begin
       if rising_edge(clk) then
--Multiplica el limite por 2 y cuenta hasta dicho limite
--por lo que funciona como un multiplicador del periodo, y por
ende, un divisor de frecuencia
--pero con opciones de frecuencias más variadas que 50MHz/2^n
           if count < (lim * 2) then
```

```
                count <= count + 1;
                if count < lim then --La mitad del tiempo en alto
                    clk_esc <= '1';
                else --La otra mitad en bajo
                    clk_esc <= '0';
                end if;
            else
                count <= 0;
            end if;
        end if;
    end process;
end Behavioral;
```

Módulo Principal (Pong):

Este módulo principal pong incorpora las partes fundamentales y bloques principales del código siendo:

- Control de barra.
- Control de pelota.
- Reloj - Cronómetro.
- Marcador.

Por lo que se irá explicando a detalle el funcionamiento de cada línea de código a continuación.

Declaramos puertos y variables.

Para el control de barra tenemos:

- bds - Boton derecho (sube)
- bdb - Boton derecho (baja)
- bis - Boton izquierdo (sube)
- bib - Boton izquierdo (baja)

Para el Sincronizador VGA y Generador de imágen tenemos:

- hs - Horizontal Syncronic
- vs - Vertical Syncronic
- red - rojo
- green - verde
- blue - azul

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

Entity pong is
```

```
    Port (
        --Reloj
        clk : in std_logic; --Reloj
        --Botones
        bds : in std_logic; --Boton derecho (sube)
        bdb : in std_logic; --Boton derecho (baja)
        bis : in std_logic; --Boton izquierdo (sube)
        bib : in std_logic; --Boton izquierdo (baja)
        --Switch de inicio
        start   : in std_logic; --Inicio del juego
        --VGA
        hs      : out std_logic; --Horizontal Syncronic
        vs      : out std_logic; --Vertical Syncronic
        red : out std_logic_vector(3 downto 0);
        green   : out std_logic_vector(3 downto 0);
        blue    : out std_logic_vector(3 downto 0);
        --Display
        disp1   : out std_logic_vector(7 downto 0);
        disp2   : out std_logic_vector(7 downto 0);
        disp3   : out std_logic_vector(7 downto 0);
        disp4   : out std_logic_vector(7 downto 0);
        disp5   : out std_logic_vector(7 downto 0);
        disp6   : out std_logic_vector(7 downto 0)
    );
End pong;
```

Componentes:

```
Architecture Behavioral of pong is
--Componentes
```

- Tenemos el divisor de frecuencia para 50 y 25 Mhz:

```
    component div is --Genera pulsos de reloj de 25Mhz
    port(
        clk50MHz: in std_logic;
        clk25MHz: out std_logic
    );
    end component;

    component rel is --Genera pulsos de reloj a razón de 50M / (2
* lim)Hz
        Generic ( lim : integer := 25000000);
        Port (
```

```
            clk     : in std_logic;
            clk_esc : out std_logic
        );
    end component;
```

- Manda a llamar al módulo de codificación de 7 segmentos el cual es utilizado para la codificación de los bloques de Control Reloj y Control Marcador.

```
    component cod7seg is --Codificador de Decimal a 7 Segmentos
        Port (
            num : in integer range 0 to 9;
            cod : out std_logic_vector (7 downto 0)
        );
    end component;
```

- Manda a llamar al módulo VGA utilizado en las prácticas de laboratorio.

```
    component vga is --Sincronizador de VGA
        generic(
            constant h_pulse    : integer := 96;
            constant h_bp       : integer := 48;
            constant h_pixels   : integer := 640;
            constant h_fp       : integer := 16;
            constant v_pulse    : integer := 2;
            constant v_bp       : integer := 33;
            constant v_pixels   : integer := 480;
            constant v_fp       : integer := 10
        );

        port(
            reloj_pixel : in std_logic;
            display_ena : out std_logic;
            column      : out integer range 0 to 96 + 48 + 640 +
16 - 1;
            row         : out integer range 0 to 2 + 33 + 480 + 10
- 1;
            h_sync      : out std_logic;
            v_sync      : out std_logic
        );
    end component;
```

Señales:

```
--Señales
```

Para el Sincronizador VGA y Generador de imágen tenemos:

- clka_vga - Reloj VGA
- de - Display Enable: Interfiere hasta que la imagen es generada.
- col - Columnas de píxeles
- row - Renglones de píxeles
- mid - Marcador jugador Izquierdo Decenas
- miu - Marcador jugador Izquierdo Unidades
- mdd - Marcador jugador Derecho Decenas
- mdu -Marcador jugador Derecho Unidades

```
signal clk_vga  : std_logic := '0';
signal de       : std_logic := '0';
signal col      : integer range 0 to (96 + 48 + 640 + 16 - 1);
signal row      : integer range 0 to (2 + 33 + 480 + 10 - 1);

signal mid, miu, mdd, mdu       : integer range 0 to 9;
```

Para el Control de Barra tenemos:

- bils - Barra Izquierda Límite Superior
- bili - Barra Izquierda Límite Inferior
- bdls - Barra Derecha Límite Superior
- bdli - Barra Derecha Límite Inferior

Estas representan la posición en bits refiriéndose a los píxeles vga de las barras, así como el tamaño que estas representan.

```
        --Limites de las barras (centrados verticalmente), cada barra
    mide 60 bits
        signal bils: integer range 0 to 420 := 210;
        signal bili: integer range 60 to 480 := 270;
        signal bdls: integer range 0 to 420 := 210;
signal bdli: integer range 60 to 480 := 270;
```

Para el Control de Pelota tenemos:

- pls - Pelota Límite Superior
- plb - Pelota Límite Bajo
- pli - Pelota Límite Izquierdo
- pld - Pelota Límite Derecho

Estos nos indican las posibles posiciones y tamaño de la pelota, siendo de 10 x 10 píxeles.
Ejemplo pli: Ancho pelota + Ancho barras = 10px +10px = 20px.

```
        --Limites de la pelota (mide 10x10)
```

```
--Pegada hasta arriba
signal pls: integer range 0 to 470 := 0;
signal plb: integer range 10 to 480 := 10;
--Centrada horizontalmente
signal pli: integer range 20 to 610 := 315;
signal pld: integer range 30 to 620 := 325;
```

Salidas de reloj a partir del módulo Reloj CLK (rel):

- tim_bot - Salida del reloj para botón
- tim_bal - Salida del reloj para la pelota
- tim_relo - Salida del reloj para el cronómetro del juego
- duclk - Decenas Unidades
- ddclk - Decenas Decenas

```
signal tim_bot  : std_logic := '0';
signal tim_bal  : std_logic := '0';
signal tim_rel  : std_logic := '0';

signal duclk, ddclk : integer range 0 to 9;
```

Para el Control de Pelota tenemos:
Los posibles estados de movimiento de la pelota, desde un estado de inicio hasta un estado de ya movimiento incluyendo cuando esta no toca una barra.

- bd - Baja Derecha
- bi - Baja Izquierda
- sd - Sube Derecha
- si - Sube Izquierda
- ei - Empieza Izquierda
- ed - Empieza Derecha
- pi - Pierde izquierda
- pd - Pierde Derecha

```
--Tipo para estados
   type state is (bd, bi, sd, si, ei, ed, pi, pd);
   signal ballst : state := ED;
```

Declaración de variables:

```
begin
```

Declaración de relojes para diversas frecuencias:

```
R1: div port map(clk, clk_vga);
R2: rel generic map(100000) port map(clk, tim_bot);
   --Frecuancia de salida: 50M / (2 * 100k) = 250Hz
R3: rel generic map(90000) port map(clk, tim_bal);
   -- Frecuancia de salida: 50M / (2 * 90k) = 277Hz -> 280Hz
```

```
R4: rel generic map(25000000) port map(clk, tim_rel);
   -- Frecuancia de salida: 50M / (2 * 25M) = 1 Hz
```

Marcador Jugador Izquierdo:

```
--Marcador Izquierdo
C1: cod7seg port map(mid, disp6);--Decenas
C2: cod7seg port map(miu, disp5);--Unidades
```

Marcador Jugador Derecho:

```
--Marcador Derecho
C3: cod7seg port map(mdd, disp2);--Decenas
C4: cod7seg port map(mdu, disp1);--Unidades
```

Reloj - Cronómetro del juego.

```
--Digitos del reloj segundero
C5: cod7seg port map(ddclk, disp4);
C6: cod7seg port map(duclk, disp3);
```

Sincronizador VGA: Todos los elementos del Bloque Generador de Imágen

```
--Sincronizador VGA
V1: vga port map(clk_vga, de, col, row, hs, vs);
```

Control Reloj - Cronómetro:

Trabaja en base a condiciones, donde la primera condición verifica el inicio del juego mediante la variable Start de inicio/reinicio, la cual establecerá en ceros las unidades y decenas de su propio reloj.
Enseguida el caso contrario evalúa que la pelota se encuentre en Empieza Derecho o Empieza Izquierdo para colocar las unidades y decenas en cero y así empiece a correr el tiempo, así el caso en el que cuando este vaya creciendo y las unidades lleguen a 9, las unidades se reinicien y las decenas empiezan a incrementarse.

```
process(tim_rel, ballst) --Control del reloj
begin
    if rising_edge(tim_rel) then --Cada segundo se incrementa
        if start = '0' then --Condición de reinicio
            duclk <= 0;
            ddclk <= 0;
        else --Corre tiempo
            case ballst is
                when ed =>
                    duclk <= 0;
```

```
                    ddclk <= 0;
                when ei =>
                    duclk <= 0;
                    ddclk <= 0;
                when others =>
                    if duclk < 9 then
                        duclk <= duclk + 1;
                    else
                        duclk <= 0;
                        if ddclk < 9 then
                            ddclk <= ddclk + 1;
                        else
                            ddclk <= 0;
                        end if;
                    end if;
            end case;
        end if;
    end if;
end process;
```

Control Barras:

Como en el diagrama de bloque de este control, trabajaremos con el tiempo de pulsación de los botones de lo jugadores ya que de esto dependerá la velocidad de desplazamiento de las barras de los jugadores, así enseguida se evaluará la variable de inicio/reinicio del juego donde se asignará la posición inicial en límites superior en inferior de ambas barras.
Los desplazamientos de las barras se hicieron independientes de cada una.

```
process(tim_bot, start, bds, bdb, bis, bib) --Control de las barras
   begin
       if rising_edge(tim_bot) then --Velocidad de desplazamiento de
las barras (250 bits por segundo)
           if start = '0' then --Condición de reinicio
               bdls <= 210;
               bdli <= 270;
               bdls <= 210;
               bdli <= 270;
           else
```

Posiciones iniciales

- Para el desplazamiento de la barra derecha:

```
--Desplazamiento de la barra derecha
```

Sube, si el “botón derecho sube” está activado y además la “barra derecha límite superior” es mayor que cero, entonces el límite superior de la barra irá disminuyendo.

```
if bds = '0' and bdls > 0 then --Sube
    bdls <= bdls - 1;
    bdli <= bdli - 1;
```

Baja, si el “botón derecho baja” está activado y además la “barra derecha límite inferior” es menor que 480, entonces el límite inferior de la barra irá aumentando.

```
elsif bdb = '0' and bdli < 480 then --Baja
    bdls <= bdls + 1;
    bdli <= bdli + 1;
```

Para otros casos, mantiene su posición.

```
else
    bdls <= bdls;
    bdli <= bdli;
end if;
```

Sube, si el "botón izquierdo sube” está activado y además la “barra izquierda límite superior” es mayor que cero, entonces el límite superior de la barra irá disminuyendo.

```
--Desplazamiento de la barra izquierda
if bis = '0' and bils > 0 then --Sube
    bils <= bils - 1;
    bili <= bili - 1;
```

Baja, si el “botón izquierdo baja” está activado y además la “barra izquierdo límite inferior” es menor que 480, entonces el límite inferior de la barra irá aumentando.

```
            elsif bib = '0' and bili < 480 then --Baja
                bils <= bils + 1;
                bili <= bili + 1;
```

Para otros casos, mantiene su posición.

```
            else
                bils <= bils;
                bili <= bili;
            end if;
        end if;
    end if;
end process;
```

Control Pelota:

Con ballst trabajaremos el estado de la pelota, para conocer su posición de acuerdo a su movimiento. Generamos la velocidad de la pelota a base del Control Reloj CLK y enseguida evaluamos la variable star para inicio/reinicio del juego.

```
process (tim_bal, start, ballst) --Control de la pelota
begin
    if rising_edge(tim_bal) then --Velocidad de aprox 280 bits/s
```

Si no se ha iniciado el juego se establece la posición inicial de la pelota.

```
            if start = '0' then --Reinicio
                pls <= 0;
                plb <= 10;
                pli <= 315;
                pld <= 325;
```

Posiciones iniciales

Es entonces cuando inicia el juego que se generan los movimientos de la pelota.

```
            else
                case ballst is
```

Para cuando la pelota Baja Derecha:

```
                    when bd => --Baja derecha
                        pls <= pls + 1;
                        plb <= plb + 1;
                        pli <= pli + 1;
                        pld <= pld + 1;
```

Para cuando la pelota Baja Izquierda:

```
                    when bi =>--Baja izquierda
                        pls <= pls + 1;
                        plb <= plb + 1;
                        pli <= pli - 1;
                        pld <= pld - 1;
```

Para cuando la pelota Sube Derecha:

```
                    when sd =>--Sube Derecha
                        pls <= pls - 1;
                        plb <= plb - 1;
                        pli <= pli + 1;
                        pld <= pld + 1;
```

Para cuando la pelota Sube Izquierda:

```
                    when si =>--Sube Izquierda
                        pls <= pls - 1;
                        plb <= plb - 1;
                        pli <= pli - 1;
                        pld <= pld - 1;
```

Para otros casos, la pelota regresará a su posición inicial.

```
                    when others =>--Vuelve a la posición inicial
                        pls <= 0;
                        plb <= 10;
                        pli <= 315;
                        pld <= 325;
                end case;
            end if;
        end if;
    end process;
```

Control Máquina de Estados:

Para el control de movimientos de rebote de la pelota es necesario conocer el estado de la misma.

Primero es necesario evaluar la variable de start de inicio/reinicio.

```
    process (tim_bal, start, ballst, pls, plb, pli, pld, bils,
bili, bdls, bdli)--Maquina de estados de la pelota
    begin
        if rising_edge(tim_bal) then
```

- Si se mantiene el reinicio empieza en derecho.

```
            if start = '0' then --Reinicio
                ballst <= ed;
            else
```

- Inicia el juego en empieza derecho y se evalúa que el reloj de unidades y decenas esté en cero.

```
                case ballst is
                    when ed =>
                        if duclk = 0 and ddclk = 0 then
```

- Es entonces que el estado de la pelota cambia a baja derecha, en caso contrario este mantendrá la misma posición de empieza derecha.

```
                            ballst <= bd;
                        else
                            ballst <= ed;
                        end if;
```

- Inicia el juego en empieza izquierdo y se evalúa que el reloj de unidades y decenas esté en cero.

```
                    when ei =>
                        if duclk = 0 and ddclk = 0 then
```

- Es entonces que el estado de la pelota cambia a baja izquierda, en caso contrario este mantendrá la misma posición de empieza izquierda.

```
                            ballst <= bi;
                        else
                            ballst <= ei;
                        end if;
```

- Cuando pierde derecho, entonces la posición de la pelota es que empiece de izquierda.

```
when pd =>
    ballst <= ei;
```

- Cuando pierde el izquierdo, entonces la posición de la pelota es que empiece derecho.

```
when pi =>
    ballst <= ed;
```

- Empieza caso general cuando sube a la derecha y sus derivaciones.

```
when sd =>
```

  - ❖ Rebota en un ángulo, es decir límite superior y barra derecha.

Rebota con la barra (En ángulo)
(Sube Derecha -> Baja Izquierda)

```
if pld >= 620 and pls <= 0 and bdls <= 10 then
```

Entonces este rebotará y bajará a la izquierda.

```
ballst <= bi;
```

  - ❖ Rebota con la barra derecha

Rebota barra
(Sube Derecha -> Sube Izquierda)

```
elsif pld >= 620 and plb >= bdls and pls <= bdli then
```

La pelota entonces cómo rebota, seguirá subiendo al lado contrario, es decir sube izquierda.

```
ballst <= si;
```

- La pelota llega a la posición de espacio de la barra derecha pero la barra no alcanza a llegar físicamente para generar un rebote.

Llega al límite derecho y no hay contacto con la barra

```
elsif pld >= 620 then
```

Entonces el estado pasa a pierde derecho, ya que no hubo rebote.

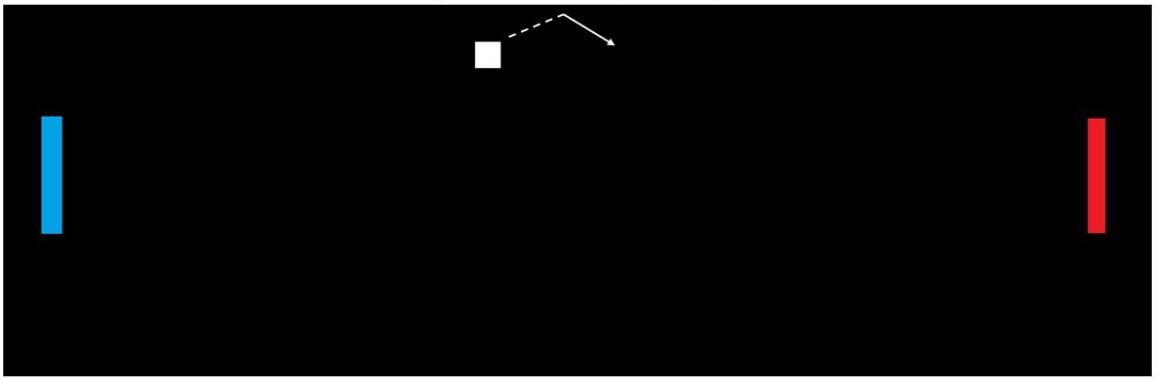
```
ballst <= pd;
```

- La pelota toca el límite superior de la pantalla, entonces rebota.

```
elsif pls <= 0 then
```

Como la pelota va subiendo derecha, empieza a bajar derecha.

Rebota Limite Superior
(Sube Derecha -> Baja Derecha)



```
ballst <= bd;
```

- En caso de que no toque el límite superior, la pelota sime manteniendo su trayectoria de sube derecha.

```
else
    ballst <= sd;
```

```
                end if;
```

- Empieza caso general cuando baja a la derecha y sus derivaciones. El funcionamiento de baja derecha, conserva la misma estructura que sube derecha pero con algunas situaciones contrarias en dirección.

```
when bd =>
    if pld >= 620 and plb >= 480 and bdli >= 470 then
        ballst <= si;
    elsif pld >= 620 and plb >= bdls and pls <= bdli then
        ballst <= bi;
    elsif pld >= 620 then
        ballst <= pd;
    elsif plb >= 480 then
        ballst <= sd;
    else --Mantiene su trayectoria
        ballst <= bd;
    end if;
```

- Empieza caso general cuando sube a la izquierda y sus derivaciones. El funcionamiento de baja derecha, conserva la misma estructura que sube derecha pero con algunas situaciones contrarias en dirección.

```
    when si =>
        if pli <= 20 and pls <= 0 and bils <= 10 then
            ballst <= bd;
        elsif pli <= 20 and plb >= bils and pls <= bili then
               ballst <= sd;
        elsif pli <= 20 then
            ballst <= pi;
        elsif pls <= 0 then
            ballst <= bi;
        else
            ballst <= si;
        end if;
```

- Empieza caso general cuando baja a la izquierda y sus derivaciones. El funcionamiento de baja derecha, conserva la misma estructura que sube derecha pero con algunas situaciones contrarias en dirección.

```
when bi =>
    if pli <= 20 and plb >= 480 and bili >= 470 then
        ballst <= sd;
    elsif pli <= 20 and plb >= bils and pls <= bili then
```

```
        ballst <= bd;
    elsif pli <= 20 then
        ballst <= pi;
    elsif plb >= 480 then
        ballst <= si;
    else
        ballst <= bi;
    end if;
```

Control de Marcadores:

Para el control de los marcadores de los jugadores, es necesario conocer el estado de la pelota, principalmente nos importa el estado pierde derecho o pierde izquierdo.

```
process (tim_bal, start, ballst) --Control de los marcadores
begin
```

- Empezamos evaluando la condición start de inicio/reinicio donde si aún no ha iniciado entonces mantendremos los marcadores en cero para unidades y decenas de ambos jugadores.

```
if rising_edge(tim_bal) then
    if start = '0' then --Reinicio (Marcadores en 0)
        mid <= 0;
        miu <= 0;
        mdd <= 0;
        mdu <= 0;
```

- Cuando el partido inicia se evalúan los casos de pierde derecho y pierde izquierdo.

```
    else
        case ballst is
```

- Si pierde derecho entonces se aumentará un punto al marcador izquierdo en unidades, en caso de que llegue a 9, marcado izquierdo decenas empezará a aumentar también.

```
            when pd => --Pierde derecho

                if miu < 9 then
                    miu <= miu + 1;
                else
                    miu <= 0;
                    if mid < 9 then
                        mid <= mid + 1;
                    else
                        mid <= 0;
                    end if;
```

```
                    end if;
```

- Si pierde izquierdo entonces se aumentará un punto al marcador derecho en unidades, en caso de que llegue a 9, marcado derecho decenas empezará a aumentar también.

```
                when pi => --Pierde Izquierdo
                    if mdu < 9 then
                        mdu <= mdu + 1;
                    else
                        mdu <= 0;
                        if mdd < 9 then
                            mdd <= mdd + 1;
                        else
                            mdd <= 0;
                        end if;
                    end if;
```

- En caso de que no haya perdido nadie aún, se conservan los marcadores.

```
                when others => --No ha perdido nadie aún
                    mid <= mid;
                    miu <= miu;
                    mdd <= mdd;
                    mdu <= mdu;
            end case;
        end if;
    end if;
end process;
```

Conclusión general del proyecto.

Podemos observar que en un principio se implementan varios de los bloques básicos que trabajamos durante las prácticas, esto nos habla de la importancia de esta metodología en la que dividimos un problema en bloques funcionales tan pequeños como sea posible, ya que así podemos reutilizar estos para futuros proyectos. La implementación de esta metodología también nos permite hacer que nuestro programa sea escalable, ya que cada módulo no interfiere con los otros, así pues, por ejemplo, en un futuro podemos adaptar simplemente el módulo del reloj que controla la velocidad de la pelota, para que dicha velocidad aumente cada que transcurre un minuto en la partida (leyendo el tiempo del reloj en segundos), y todo esto sin alterar ninguno de nuestros otros módulos.
En cuanto al sistema VGA, que es la clave angular del proyecto, concluimos que aunque sea un protocolo que ha disminuido significativamente en uso, debido a la aparición de HDMI, aún está vigente y tiene múltiples aplicaciones.