Proyecto final de la asignatura Tecnología de computadores (TC)

“Juego de memoria” en VHDL

Autor: Sergio Domínguez Alcalá

Implementación usando VHDL de un juego de memoria de matriz, donde se le muestran al usuario celdas de diferente color durante un tiempo determinado de tiempo y este debe memorizarlas, luego se pintan todas las celdas iguales y el usuario tiene que seleccionar las celdas que han sido iluminadas por el sistema para ganar la ronda/partida.

Ejemplo de juego de memoria de una matriz:

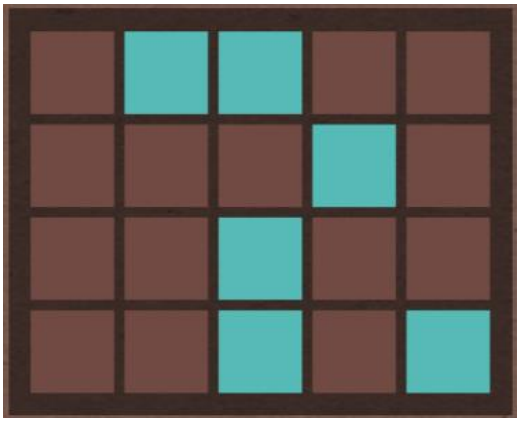
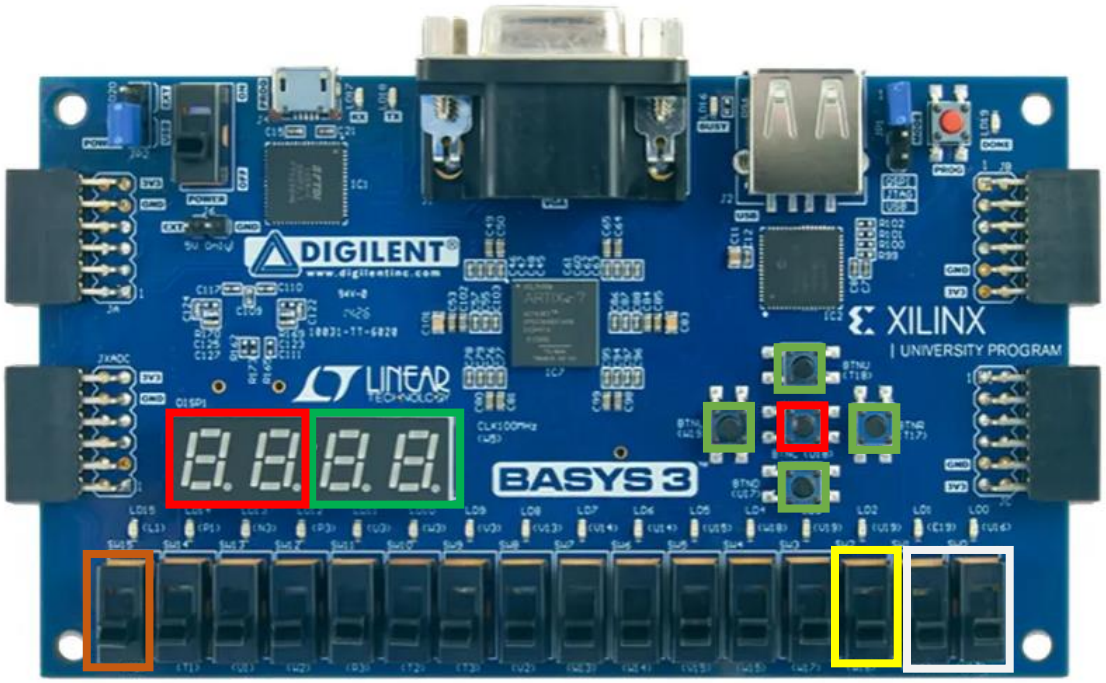


Imagen del juego implementado en basys3:

Implementación en VHDL:

Primero vamos a explicar cómo interactúa el código VHDL con la placa



Entidad top del sistema:

```
entity controladorVGA is
Port (clk,sw_rst, btn_aceptar, sw_seleccionar ,btn_arriba, btn_izq, btn_der, btn_abajo: in std_logic;
        controlDificultad: in std_logic_vector(1 downto 0);
        hSync, vSync: out std_logic;
        vgaRed, vgaGreen, vgaBlue: out std_logic_vector(3 downto 0);
        display     : OUT std_logic_vector (6 DOWNTO 0);
        s_display   : OUT std_logic_vector (3 DOWNTO 0));
end controladorVGA;
```

Puertos Sistema:

- Los botones con el recuadro verde corresponden con btn_arriba, btn_izq, btn_der, btn_abajo que se encargan del movimiento del jugador
- El botón con el recuadro rojo corresponde con btn_aceptar
- Los switches con el recuadro blanco corresponden con el puerto controlDificultad
- El switch con el recuadro naranja corresponde con el puerto sw_seleccionar
- hSync, vSync, vgaRed, vgaGreen, vgaBlue son los puertos del vga
- display y s_display corresponden con los display de 7 segmentos de la fpga

Como funciona:

Se usan botones del cuadro verde para que el jugador pueda desplazarse hasta el cuadrado/celda que desea seleccionar, una vez el jugador se encuentra encima de la celda que desea seleccionar usa el switch de seleccionar para marcar esa celda como seleccionada, cuando no desee marcar seleccionar celdas debe tener el switch bajado (a 0).

Una vez el usuario quiera terminar la ronda pulsara el botón de aceptar, y el sistema calculara si ha fallado o a acertado, si ha fallado el sistema pone a 0 el contador de rondas ganadas seguidas (recuadro verde en displays), si ha ganado aumenta en 1 el contador de rondas ganadas seguidas y si las rondas ganadas actuales son mayores que el récord de rondas ganadas seguidas (recuadro rojo en displays) actualiza el contador del récord de rondas ganadas.

Cada vez que finaliza una ronda el sistema selecciona de forma pseudoaleatoria nuevas celdas y se las muestra al jugador.

Los switches de dificultad ahora mismo están sin uso, pero se han dejado implementados por si en un futuro se desea usarlos.

Componentes del TOP:

```
component syncJuegoVga is
Port (clk, rst,seleccionar,aceptar, arriba, izquierda, derecha, abajo: in std_logic;
        dificultad: in std_logic_vector(1 downto 0);
        hSync, vSync: out std_logic;
        rondGanadasTotal_der,rondGanadasTotal_izq, rondGanadasSeg_der,rondGanadasSeg_izq: out std_logic_vector (3 DOWNTO 0);
        r,g,b: out std_logic_vector(3 downto 0));
end component;

component debouncer IS
  PORT (
    rst: IN std_logic;
    clk: IN std_logic;
    x: IN std_logic;
    xDeb: OUT std_logic;
    xDebFallingEdge: OUT std_logic;
    xDebRisingEdge: OUT std_logic
  );
END component;

component displays is
    Port (
        rst : in STD_LOGIC;
        clk : in STD_LOGIC;
        digito_0 : in  STD_LOGIC_VECTOR (3 downto 0);
        digito_1 : in  STD_LOGIC_VECTOR (3 downto 0);
        digito_2 : in  STD_LOGIC_VECTOR (3 downto 0);
        digito_3 : in  STD_LOGIC_VECTOR (3 downto 0);
        display : out  STD_LOGIC_VECTOR (6 downto 0);
        display_enable : out  STD_LOGIC_VECTOR (3 downto 0)
    );
end component;
```

El componente syncJuegoVga es el que se encarga de toda la lógica del juego, usamos debouncer para los botones y el componente de display para usar los displays de 7 segmentos.

Componente syncJuegoVga:

Se encarga de toda la lógica del juego y de controlar como se muestra el estado actual del juego a través de la vga

```
entity syncJuegoVga is
    Port (
      clk, rst, seleccionar, aceptar, arriba, izquierda, derecha, abajo: in std_logic;
      dificultad: in std_logic_vector(1 downto 0);
      rondGanadasTotal_der,rondGanadasTotal_izq, rondGanadasSeg_der,rondGanadasSeg_izq: out std_logic_vector (3 DOWNTO 0);
      hSync, vSync: out std_logic;
      r, g, b: out std_logic_vector(3 downto 0)
    );
end syncJuegoVga;
```

Contiene las entradas del sistema que hemos visto, usamos puertos de salida rondGanadasTotal_der, rondGanadasTotal_izq que corresponden con el numero que se muestra en el display en la parte del recuadro rojo, el contador del record de rondas ganadas seguidas y rondGanadasSeg_der, rondGanadasSeg_izq que se muestran en el display en la parte del recuadro verde, el contador de rondas ganadas seguidas.

La entidad syncJuegoVga tiene los siguientes componentes:

```
component pseudorng is
Port ( clock : in STD_LOGIC;
       reset : in STD_LOGIC;
       en : in STD_LOGIC;
       Q : out STD_LOGIC_VECTOR (7 downto 0);
       Q_int : out INTEGER;
       check: out STD_LOGIC);
end component;

component divisor is
port (
    rst: in STD_LOGIC;
    clk_entrada: in STD_LOGIC; -- reloj de entrada de la entity superior
    clk_salida: out STD_LOGIC -- reloj que se utiliza en los process del programa principal
);
end component;

 component contadorMod10 is
 port(rst: in std_logic;
      clk: in std_logic;
      enable: in std_logic;
      rst_contador: in std_logic;
      contador_segs: out INTEGER) ;
 end component;
```

El componente divisor es un divisor de frecuencia cuya salida se usa en el componente contadorMod10, el cual se encarga de contar segundos, su salida la usamos para que el sistema muestre al usuario las celdas seleccionadas durante un tiempo especifico

El componente pseudorng (basado LFSR) es el componente encargado de generar el numero pesudoaleatorio que usa el sistema para seleccionar las celdas que se le muestran al usuario usando puertas XOR:

```
architecture Behavioral of pseudorng is
  signal Qt : STD_LOGIC_VECTOR(7 downto 0) := x"01";
  signal Q_int_signal : INTEGER;  -- Señal interna para representar Q como entero
begin
  PROCESS(clock)
    variable tmp : STD_LOGIC := '0';
  BEGIN
    IF rising_edge(clock) THEN
      IF (reset = '1') THEN
        Qt <= x"01";
      ELSIF en = '1' THEN
        tmp := Qt(4) XOR Qt(3) XOR Qt(2) XOR Qt(0);
        Qt <= tmp & Qt(7 downto 1);
      END IF;
    END IF;
  END PROCESS;
  -- Convertir la salida Q a un número entero
  Q_int_signal <= TO_INTEGER(unsigned(Qt));
  -- Asignar las salidas
  check <= Qt(7);
  Q <= Qt;
  Q_int <= Q_int_signal;

end Behavioral;
```

Para facilitar la modificación de como se muestra al usuario la salida de la VGA se usan las siguientes constantes:

```
  -- ***800x600@60Hz*** --
  constant FRAME_WIDTH : natural := 800;
  constant FRAME_HEIGHT : natural := 600;

  constant H_FP : natural := 40;   -- H front porch width (pixels)
  constant H_PW : natural := 128;  -- H sync pulse width (pixels)
  constant H_MAX : natural := 1040; -- H total period (pixels)

  constant V_FP : natural := 1;    -- V front porch width (lines)
  constant V_PW : natural := 4;    -- V sync pulse width (lines)
  constant V_MAX : natural := 666;  -- V total period (lines)

  constant H_CELL : natural := 28;   -- Ancho de cada celda en píxeles
  constant V_CELL : natural := 28;   -- Alto de cada celda en píxeles

  constant H_CELLS : natural := 8;  -- Número de celdas en el eje horizontal
  constant V_CELLS : natural := 8;  -- Número de celdas en el eje vertical

  constant H_MARGIN : natural := 2;  -- Margen alrededor de cada celda
  constant V_MARGIN : natural := 2;  -- Margen alrededor de cada celda

  constant H_TOTAL : natural := FRAME_WIDTH; -- Ancho total de la pantalla
  constant V_TOTAL : natural := FRAME_HEIGHT; -- Alto total de la pantalla
```

Además, vamos a usar las siguientes señales:

Señales del vga:

```
-- Synchronization Signals
signal hPosCurrent, hPosNext: integer range 1 to H_MAX;
signal vPosCurrent, vPosNext: integer range 1 to V_MAX;

-- RGB Signals
signal rgbCurrent, rgbNext: std_logic_vector(11 downto 0);
```

Para hSync vamos a usar dos señales, hPosCurrent que indica la posición actual de hSync y hPosNext que indica la posición que va a pasar a tener hSync, hacemos lo mismo para vSync.

La señas rgbCurrent indica el color del pixel actual y rbgNext indica el color que va a pasar a tener el pixel actual, hemos juntado vgaRed,vgaBlue,vgaGreen en una sola señal de 12 bits, los bits (3-0) corresponden con el rojo, los bits (7-4) al verde y los bits (11-8) al azul.

Una vez hemos terminado los cálculos correspondientes con las señales, las asignamos a los puertos de salida:

```
-- Salidas de sincronización y color
hSync <= '1' when hPosCurrent <= H_FP or hPosCurrent > H_FP + H_PW + H_TOTAL else '0';
vSync <= '1' when vPosCurrent <= V_FP or vPosCurrent > V_FP + V_PW + V_TOTAL else '0';

r <= rgbCurrent(3 downto 0);
g <= rgbCurrent(7 downto 4);
b <= rgbCurrent(11 downto 8);
```

vSync y hSync sirven para indicar si estamos dentro del periodo de sincronización vertical (vSync = 1) y del periodo de sincronización horizontal (hSync = 1)

Las constantes H_FP, H_PW y H_TOTAL indican el periodo de sincronización horizontal y V_FP, V_PW, V_TOTAL indican el periodo de sincronización vertical, si hPosCurrent se encuentra dentro del periodo de sincronización horizontal, hSync = 1 y si vPosCurrent se encuentra dentro del periodo de sincronización vertical, vSync = 1

Señales donde almacenamos las celdas seleccionadas:

```
signal celdas_elegidas : STD_LOGIC_VECTOR(63 downto 0);
signal celdas_elegidas_jugador : STD_LOGIC_VECTOR(63 downto 0);

signal celda_actual_jugador: integer := 0;
```

Almacenamos las celdas de forma lineal, como tenemos una matriz de 8x8, tiene 64 celdas, por lo que usamos dos STD_LOGIC_VECTOR para almacenar las celdas elegidas por el sistema y por el jugador, si por ejemplo el bit 3 de celdas_elegidas está a 1, significa que el sistema ha seleccionado esa celda.

Usamos la celda_actual_jugador para almacenar la celda actual del jugador

Process de syncJuegoVga:

Tenemos 4 process:

CeldasSeleccionadas: se encarga de seleccionar las celdas del sistema, usa la salida del componente que genera números pseudo aleatorios para seleccionar las celdas del sistema, usamos un contador (cnt_rng_num) para asegurarnos que no seleccione mas de la mitad de las celdas del sistema, si el usuario pulsa el botón de aceptar reseteamos las celdas elegidas y el contador.

```
processCeldasSeleccionadas:process(clk, rst,aceptar)
  begin
    if rst = '1' then
      -- Inicialización cuando se activa el reset
      salida_pseudorng <= (others => '0');
      celdas_elegidas <= (others => '0');
      cnt_rng_num <= 0;
    elsif rising_edge(clk) then
      if en = '1' then
          if cnt_rng_num < (H_CELLS*V_CELLS)/2 then
          cnt_rng_num <= cnt_rng_num + 1;--contador de numeros aleatorios generados
          end if;

          if aceptar = '1' then
          cnt_rng_num <= 0;
          celdas_elegidas <= (others => '0');
          end if;
           --Podemos modificar la comparacion de salida_pseudorng cuando mayor sea el numero con el que comparamos mas dificultad
            if salida_pseudorng_int <20 and cnt_rng_num < (H_CELLS*V_CELLS)/2 then

                celdas_elegidas(cnt_rng_num) <= '1';

          end if;
      end if;
```

MovimientoJug: Se encarga de mover al jugador

```
   processMovimientoJug:process(clk, rst,seleccionar, aceptar, arriba, izquierda, derecha, abajo)
    begin
    if rst = '1' then
      celdas_elegidas_jugador <= (others => '0');
      celda_actual_jugador <= 1;
    elsif rising_edge(clk) then

        if seleccionar = '1' then
            if(celda_actual_jugador<64) then
                celdas_elegidas_jugador(celda_actual_jugador) <= '1';
            end if;
        end if;

        if aceptar ='1' then
            celdas_elegidas_jugador <= (others => '0');
            celda_actual_jugador <= 0;
            --celdas_elegidas <= (others => '0');
            --cnt_rng_num <= 0;
        end if;
        if arriba ='1' then
            if celda_actual_jugador <= H_CELLS then
                celda_actual_jugador <= (H_CELLS * V_CELLS) - (H_CELLS - celda_actual_jugador);
            else
                celda_actual_jugador <= celda_actual_jugador - H_CELLS;
```

```
        if izquierda ='1' then
            if (celda_actual_jugador) mod H_CELLS /= 0 then
                celda_actual_jugador <= celda_actual_jugador - 1;
            else
                celda_actual_jugador <= celda_actual_jugador + V_CELLS -1;
            end if;
        end if;
        if derecha = '1' then
            if (celda_actual_jugador+1) mod H_CELLS /= 0 then
                celda_actual_jugador <= celda_actual_jugador + 1;
            else
                celda_actual_jugador <= celda_actual_jugador - V_CELLS +1;
            end if;
        end if;
        if abajo ='1' then
            if celda_actual_jugador + H_CELLS <= H_CELLS*V_CELLS then
                celda_actual_jugador <= celda_actual_jugador + H_CELLS;
            else
                celda_actual_jugador <= celda_actual_jugador - ((V_CELLS - 1) * H_CELLS);
            end if;
        end if;


    end if;
end process;
```

Las celdas están numeradas de la celda 0 a la celda 63, por ejemplo si pulsamos el botón arriba y estamos en una fila que tiene una fila superior solo tenemos que hacer celda_actual_jugador – H_CELLS siendo H_CELLS el numero de celdas por fila, en nuestro caso 8 celdas, sin embargo si estamos en la primera fila, como no tiene fila superior tendríamos que ir a la ultima fila, por ejemplo pasar de la celda 1 a la 57, y eso lo calculamos así : (H_CELLS * V_CELLS) - (H_CELLS - celda_actual_jugador) , V_CELLS son 8 celdas por columna.

Si se pone el switch a 1 las celdas donde este pasando el jugador se van a guardar en el vector de celdas_elegidas_jugador.

Si se pulsa el botón de aceptar se ponen la celda del jugador y el vector de celdas_elegidas_jugador a 0.

Ronda: Se encarga de procesar el resultado de la ronda cuando pulsamos aceptar

```
 processRonda : process(clk, rst, aceptar)
begin
  if rst = '1' then
    -- Inicialización cuando se activa el reset
    rondGanadasTotal_derAux<= (others => '0');
    rondGanadasTotal_izqAux<= (others => '0');
    rondGanadasSeg_derAux<= (others => '0');
    rondGanadasSeg_izqAux<= (others => '0');


  elsif rising_edge(clk) then
    if aceptar = '1' and contador_segs>30 then
      rst_contador <= '1';
      if celdas_elegidas_jugador = celdas_elegidas then

      if unsigned(rondGanadasSeg_derAux) < 9 then
         rondGanadasSeg_derAux <= std_logic_vector(unsigned(rondGanadasSeg_derAux) + 1);
      else
          rondGanadasSeg_derAux<= (others => '0');
          rondGanadasSeg_izqAux <= std_logic_vector(unsigned(rondGanadasSeg_izqAux) + 1);
      end if;

   if rondGanadasTotal_izqAux < rondGanadasSeg_izqAux then
   --Significa que ha cambiado de 9 a 10 por ejemplo
   rondGanadasTotal_izqAux<= std_logic_vector(unsigned(rondGanadasSeg_derAux) + 1);
   rondGanadasTotal_derAux  <= (others => '0');

   else

   --Si ha cambiado de 8 a 9 por ejemplo
          if rondGanadasTotal_derAux  <= rondGanadasSeg_derAux and rondGanadasTotal_izqAux <= rondGanadasSeg_izqAux then
                      rondGanadasTotal_derAux  <=  std_logic_vector(unsigned(rondGanadasSeg_derAux) + 1);
          end if;
   end if;

      else
        rondGanadasSeg_derAux <= "0000";
        rondGanadasSeg_izqAux <= "0000";
      end if;
      else
       rst_contador <= '0';
    end if;
  end if;
end process;
```

Si el jugador pulsa el botón de aceptar mientras se están mostrando aun las celdas seleccionadas del sistema mantiene el rst_contador a 0 (El contador de segundos), si pulsa aceptar una vez ya se han terminado de mostrar las celdas elegidas por el sistema, pone el rst_contador a 1 y se encarga de actualizar los valores que se van a mostrar en el display, primero aumenta el contador de rondas ganadas seguidas, una vez esta actualizado, modifica el contador del record de rondas ganadas seguidas si es necesario.

VGA: Proceso encargado de la salida por vga.

```
 processVGA:process(clk, rst)
 begin
   if rst = '1' then
     -- Inicialización cuando se activa el reset
     hPosCurrent <= 1;
     vPosCurrent <= 1;
     rgbCurrent <= (others => '0');
     en <= '0';  -- Deshabilitar generador n°aleatorios

   elsif rising_edge(clk) then
        en <=  '1';
    -- Lógica de sincronización horizontal
     if hPosCurrent = H_MAX then
       hPosNext <= 1;
       -- Incrementar la posición vertical al final de cada línea horizontal
       if vPosCurrent = V_MAX then
         vPosNext <= 1;
       else
         vPosNext <= vPosCurrent + 1;
       end if;
     else
       hPosNext <= hPosCurrent + 1;
     end if;
```

Esta parte del process se encarga de actualizar las coordenadas de las posiciones verticales y horizontales

```
 -- Lógica de generación de píxeles para una matriz de celdas 8x8 celdas con tamaño 32 pixeles con marco
if hPosCurrent >= (H_TOTAL/2) - (H_CELLS/2 * (H_CELL+(2*H_MARGIN))) and hPosCurrent < (H_TOTAL/2) + (H_CELLS/2 *(H_CELL)) and
    vPosCurrent >= (V_TOTAL/2) - (V_CELLS/2 *(V_CELL)) and vPosCurrent < (V_TOTAL/2) + (V_CELLS/2 *(V_CELL + 2*V_MARGIN) ) then
   -- Determinar la posición relativa dentro de la celda actual
   if (hPosCurrent mod (H_CELL + H_MARGIN) > H_MARGIN and  --si lo reduzco aumenta el tamaño de los margenes
       vPosCurrent mod (V_CELL + V_MARGIN) > V_MARGIN) and
      (hPosCurrent mod (H_CELL + H_MARGIN) < H_CELL + H_MARGIN and
       vPosCurrent mod (V_CELL + V_MARGIN) <  V_CELL + V_MARGIN) then

        -- Calcular la posición de la celda actual
   currentCellX <= (hPosCurrent - (H_TOTAL/2) + (H_CELLS/2 * (H_CELL+H_MARGIN))) / (H_CELL+H_MARGIN);
   currentCellY <= (vPosCurrent - (V_TOTAL/2) + (V_CELLS/2 * (V_CELL+V_MARGIN))) / (V_CELL + V_MARGIN);

   -- Calcular el índice de celdas_elegidas
   index <= currentCellX + currentCellY * H_CELLS;
```

El primer if se encarga de comprobar si actualmente estamos dentro de la zona de pintado, H_TOTAL y V_TOTAL son 800 y 600 px respectivamente, como queremos mostrar centrada la matriz, primero (H_TOTAL/2) para ponernos en medio de la pantalla, y luego le restamos el espacio que ocuparían 4 celdas, la mitad de la matriz, hacemos lo mismo para hPosCurrent y vPoscurrent, si se cumple la condición del if significa que estamos dentro del área donde se pinta la matriz, si no se cumple significa que estamos fuera y se pinta negro.

El segundo if es para calcular cuando se esta dentro de la celda, la parte donde se pinta, si se cumple la condición significa que estamos dentro de la celda y se pintara del color que haga falta en ese momento y si no se cumple significa que estamos fuera de la parte que se pinta de las celdas y se pinta de negro (esto genera los márgenes)

Si se cumplen las dos condiciones significa que estamos dentro de la celda y calculamos en que celda estamos

```
--Pintado de las celdas
-- Pintado inicial de las celdas seleccionadas aleatoriamente
if  contador_segs <30 then
  if celdas_elegidas(index) = '1' then
   rgbNext <= "111100000000";  -- Píxel azul
  else
    rgbNext <= (others => '1');  -- Píxel blanco
  end if;
else
  if celdas_elegidas_jugador(index) = '1' or celda_actual_jugador = index then
    if celdas_elegidas_jugador(index) = '1' then
        rgbNext <= "111111110000";
    end if;
     if celda_actual_jugador = index then
         rgbNext <= "000011110000";  -- Píxel Verde
    end if;
  else
    rgbNext <= (others => '1');  -- Píxel blanco
  end if;
end if;
```

Aquí realizamos el pintado de las celdas, mientras el contador sea menos que 30 pintamos las celdas seleccionadas por el sistema de azul, si el contador ha pasado de 30 pintamos la celda actual del jugador de verde, y las celdas que este ha seleccionado de azul celeste, si las celdas no han sido seleccionadas se pintan de blanco

```
        else
          -- Pintar el marco en negro
          rgbNext <= (others => '0');  -- Píxel negro
        end if;
      else
        rgbNext <= (others => '0');  -- Píxel negro fuera de las celdas
      end if;

      -- Actualización de las señales actuales
      hPosCurrent <= hPosNext;
      vPosCurrent <= vPosNext;
      rgbCurrent <= rgbNext;
    end if;
  end process;
```

Aquí es donde si no se cumplen las condiciones de los dos ifs se pinta de negro y donde actualizamos las señales.