Proyecto Final de Sistemas Digitales Avanzados

Pérez, Hugo; Campos, Julio
{A01337226, A1339588} @itesm.mx
Instituto Tecnológico y de Estudios Superiores de Monterrey, Campus Ciudad de México

***Abstract*—**
En este escrito se reportará el proceso de diseño e implementación del popular juego "Pong" en la tarjeta de desarrollo, basada en FPGA, Altera DE2-115, a través de la programación de la misma utilizando el lenguaje VHDL. Se utilizaron componentes de la tarjeta como botones, switches y el reloj interno (50 MHz) para poder crear las condiciones adecuadas de juego. En general, se hizo uso de tres módulos: módulo de debounce para los botones, divisor de frecuencia a 300 Hz y módulo de control.

***Índice de términos*— FPGA, Pong, VHDL.**

MARCO TEÓRICO

El desarrollo en VHDL implica entre otras cosas el conocimiento sobre sistemas digitales. Saber los distintos tipos de operadores lógicos, que dispositivos se pueden formar a partir de compuertas, la lógica combinatoria y secuencial. Para el presente trabajo todos los conocimientos adquiridos a lo largo de los dos cursos de electrónica digital serán demostrados.

VHDL es en realidad una forma corta para referirse a VHSICHDL que significa Very High Speed Integrated Circuit Hardware Description Language. Normalmente es confundido con un lenguaje de programación, pero formalmente hablando es un lenguaje de descripción de hardware, esto quiere decir que VHDL describe a través de código el funcionamiento de un circuito.

Un código en VHDL consta principalmente de dos partes: entidad y arquitectura. Esta primera parte sirve para indicar las entradas y salidas globales del componente a elaborar. Por otra parte la arquitectura describe como tal la estructura o el comportamiento del componente, por ello es que la arquitectura puede ser estructural o de comportamiento. En la primera se conectan componentes entre sí con la ayuda de señales internas, mientras que en el segundo se escriben expresiones que puedan definir el actuar del componente conforme estímulos.

DESARROLLO DE ACTIVIDADES

Se seleccionó el clásico juego de Pong para su diseño e implementación por lo que se partió de una serie de parámetros de diseño:

1. Despliegue del juego en VGA
2. Uso de botones para subir y bajar la barra de los jugadores.
3. Despliegue de marcadores en VGA
4. El primero en ganar 5 juegos gana la partida

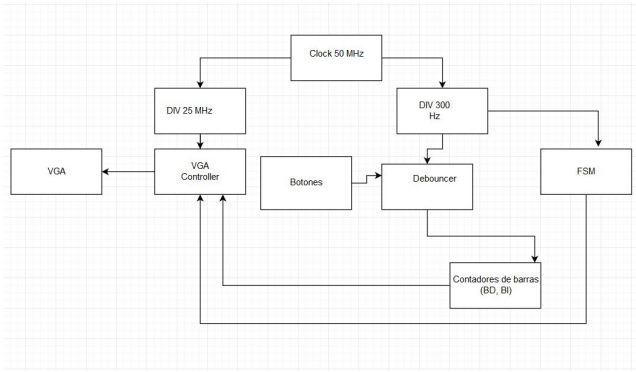

Figura 1: Diagrama de bloques del componente

En la figura 1 se puede apreciar un diagrama de bloques representando la interconexión de los componentes dentro del sistema final. Este es de gran utilidad, pues permite visualizar fácilmente la relación que hay entre distintos bloques funcionales.

En cuanto a la implementación del trabajo, se comenzó por utilizar un código genérico de despliegue en VGA, mostrado en la figura 3. Solamente se tiene que tomar el momento donde HDisp = '1' y VDisp = '1' para poder realizar modificaciones en la visualización del dispositivo. Este corresponder al bloque "VGA Controller" del diagrama de bloques de la figura 1. Es importante mencionar que este funciona con un divisor de frecuencia de 25 MHz, mostrado en la figura 2.

```
--Se necesita divisor de 25MHz --
  process(clk_div)
      begin
          if rising_edge(CLOCK_50) then
              clk_div <= not(clk_div);
          else
              clk_div <= clk_div;
          end if;
  end process;
--Se necesita divisor de 25MHz --
```

Figura 2. Divisor de frecuencia de 25 MHz

```
if rising_edge(clk_div) then
--Análisis horizontal --
    if (contHS <= 95) then -- TPW 0-95
        HS_signal <= '0'; --No es necesario el HSync
        HDisp <= '0';   --No es zona para poner color
        contHS := contHS + 1; --Sigue contando
    elsif (contHS <= 143) then --Back porch 96-143
        HS_signal <= '1'; --HSync
        HDisp <= '0'; --No es zona para poner color
        contHS := contHS + 1; --Sigue contando
    elsif (contHS <= 783) then --Horizontal display time 144-783
        HS_signal <= '1'; --HSync
        HDisp <= '1'; -- Ya es zona para poner color
        contHS := contHS + 1; --Sigue contando
    elsif (contHS <= 799) then --Front porch 784-799
        HS_signal <= '1'; --HSync
        HDisp <= '0'; --No es zona para poner color
        contHS := contHS + 1;
    else--Ya terminó la línea, contHS contó 800 veces ya
        contHS := 0;
        contVS := contVS + 1;
    end if;
    --Análisis vertical--
    if (contVS <= 1) then -- TPW 0-1
        VS_signal <= '0';
        VDisp <= '0'; --No es zona para poner color
    elsif (contVS <= 30) then -- Back Porch 2-30
        VS_signal <= '1';
        VDisp <= '0'; --No es zona para poner color
    elsif (contVS <= 510) then -- Vertical Display time 31-510
        VS_signal <= '1';
        VDisp <= '1'; -- Ya es zona para poner color
    elsif (contVS <= 520) then -- Front Porch 511-520
        VS_signal <= '1';
        VDisp <= '0';   --No es zona para poner color
    else --Se terminó pantalla, contVS contó 521 veces ya
        contVs := 0;
    end if;
    else
    contHs := contHs;
end if;
```

Figura 3: Código genérico para la comunicación con VGA

Posterior a este, se incluirían los objetos como barras, y líneas en los bordes para limitar la zona de juego. En primera instancia, se logró el movimiento de las barras (arriba y abajo), así como la implementación de una pelota de movimiento manual y el despliegue del marcador en VGA.

Viéndolo desde el punto de vista del hardware, las barras se manipulaban con los botones, mientras que la pelota se manipulaba mediante el uso de switches (cada switch controlaba el movimiento en un sentido).

En la figura 4 se muestra la implementación de un divisor de frecuencia de 300 Hz, el cual permite el control de los módulos de conteo de las barras y el movimiento de la pelota.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY DIV_Pong is
    PORT(
    CLOCK_50: IN STD_LOGIC;
    clk_out: OUT STD_LOGIC
    );
END DIV_Pong;
ARCHITECTURE beh of DIV_Pong is
    signal clk_div: STD_LOGIC;
    begin
    process (CLOCK_50)
    variable cont: integer := 0;
    begin
        if rising_edge (CLOCK_50) then
                cont := cont + 1;
            if cont = 100000 then --Número de flancos de subida
                cont := 0;
                clk_div <= NOT (clk_div);
            else
                clk_div <= clk_div;
            end if;
        else
        cont := cont;
        end if;
    end process;
        clk_out <= clk_div;
end beh;
```

Figura 4: Divisor de frecuencia de 300 Hz

En la Figura 5 se muestra el código que se utilizó para lograr que las barras, através de distintos contadores, se movieran sin salirse de la pantalla con la activación de los botones, siendo BD_Arriba, BD_Abajo generadas a través port map con el Debouncer, como se muestra en la Figura 6.

Es importante mencionar que lo que en realidad se estaba manipulando tanto en las barras, como en la bola, fue el pixel central. El efecto de la forma bidimensional se obtuvo dándole un rango al contador que causaba el efecto de "pintar" en la VGA a partir de este pixel central. Así se obtuvieron las medidas esperadas (bola 12 x 12 y barras 14 x 80). Esto se muestra en la Figura 7.

```
        --MOVIMIENTO DE LAS BARRAS--
-- Movimiento de la barra BI_Abajo--
    process(clk_div2)
    variable CentroBarraIzquierda : integer := 270;
    begin
        if rising_edge(clk_div2) then --72 a 469
            if(BI_Abajo = '1' AND CentroBarraIzquierda < 473) then
                CentroBarraIzquierda := CentroBarraIzquierda + 1;
            end if;
            if(BI_Arriba = '1' AND CentroBarraIzquierda > 71) then
                CentroBarraIzquierda := CentroBarraIzquierda - 1;
            end if;
        end if;
    ContadorBI <= CentroBarraIzquierda;
    end process;
-- Movimiento de la barra BI_Abajo--
-- Movimiento de la barra BI_Arriba--
    process(clk_div2)
    variable CentroBarraDerecha : integer := 270;
    begin
        if rising_edge(clk_div2) then
            if (BD_Abajo = '1' AND CentroBarraDerecha < 473) then
                CentroBarraDerecha := CentroBarraDerecha + 1;
            end if;
            if(BD_Arriba = '1' AND CentroBarraDerecha > 71) then
                CentroBarraDerecha := CentroBarraDerecha - 1;
            end if;
        end if;
    ContadorBD <= CentroBarraDerecha;
    end process;
 -- Movimiento de la barra BI_Arriba--
        --MOVIMIENTO DE LAS BARRAS--
```

Figura 5: Movimiento de Barras usando contadores

```
U0 : Debouncer port map(clk_div, KEY(0), '1', BD_Arriba);
U1 : Debouncer port map(clk_div, KEY(1), '1', BD_Abajo);
U2 : Debouncer port map(clk_div, KEY(2), '1', BI_Arriba);
U3 : Debouncer port map(clk_div, KEY(3), '1', BI_Abajo);
U4 : DIV_Pong   port map(CLOCK_50, clk_div2);
```

Figura 6: Generación de señales BD y BI

```
if (HDisp = '1' AND VDisp = '1') then


    --Para pelota--
    if( (contHS > LimIzqBall AND contHS < LimDerBall) AND (contVS > LimInfBall AND contVS < LimSupBall)) then
        RED<=  "11111111";
        GR  <= "11111111";
        BL  <= "11111111";
    else
        RED<=  "00000000";
        GR  <= "00000000";
        BL  <= "00000000";
    end if;
    --Para pelota--

    --Para Barra BI_Abajo--
    if( (contHS > LimIzqBarrI AND contHS < LimDerBarrI) AND (contVS > LimInfBarrI AND contVS < LimSupBarrI)) then

        RED<=  "11111111";
        GR  <= "11111111";
        BL  <= "11111111";
    end if;
    --Para Barra BI_Abajo--

    --Para Barra BI_Arriba--
    if( (contHS > LimIzqBarrD AND contHS < LimDerBarrD) AND (contVS > LimInfBarrD AND contVS < LimSupBarrD)) then

        RED<=  "11111111";
        GR  <= "11111111";
        BL  <= "11111111";
    end if;
    --Para Barra BI_Arriba--
```

Figura 7. Forma bidimensional de objetos del proyecto en VGA

Una vez logrado el movimiento manual de la pelota se continuó con la ideación de la máquina de estados que se utilizaría para obtener el movimiento de forma automática, en

esta fase, hubo muchas ideas en cuanto al movimiento que debía tener.

La idea que se aceptó al final fue la de utilizar una máquina de siete estados:

1. Movimiento
2. ReboteArriba
3. ReboteAbajo
4. ReboteIzquierda
5. ReboteDerecha
6. ScoreP1
7. ScoreP2

La idea general que se tuvo fue que el movimiento podía descomponerse en sus componentes en el eje "x" y el eje "y". De aquí podía analizarse su comportamiento al interactuar con los objetos que podían visualizarse (barras, parte superior, parte inferior, zona de anotación).

Se llegó a la conclusión de que el conteo debía cambia su signo (contar ascendentemente o descendentemente) dependiendo del objeto que se golpeara.

Por ejemplo, si se golpeaba la parte superior, debía cambiarse el signo de conteo en el eje "y" y mantener el del eje "x", pues pasaba de un conteo que se visualizaba como un movimiento hacia arriba a un conteo que causara un movimiento hacia abajo; en caso de que se golpeara alguna de las barras, tiene que invertir ambos signos.

En la figura 8 se muestra el código de seis de los siete estados:

```
when ReboteArriba =>
    signoY := '1';
    estado_pelota <= Movimiento;
when ReboteAbajo =>
    signoY := '0';
    estado_pelota <= Movimiento;
when ReboteIzquierda =>
    signoX := '0';
    signoY := NOT(signoY);
    estado_pelota <= Movimiento;
when ReboteDerecha =>
    signoX := '1';
    signoY := NOT(signoY);
    estado_pelota <= Movimiento;
when ScoreP1 =>
    signoX := signoX;
    signoY := signoY;
    CentroX := 470;
    marp1 := marp1 + 1;
    estado_pelota <= Movimiento;
when ScoreP2 =>
    signoX := signoX;
    signoY := signoY;
    CentroX := 458;
    marp2 := marp2 + 1;
    estado_pelota <= Movimiento;
when others =>
estado_pelota <= Movimiento;
```

Figura 8. Seis estados por los que puede pasar

En cuanto a los estados de score, en cuanto llegaba a la zona de score, se enviaba a la bola a un lugar cerca del centro a la misma altura que con la que entró a la misma.

El código del estado de Movimiento se encuentra en la figura 9 y 10. No está optimizado, pues el equipo tenía poco tiempo para que este proyecto pudiera funcionar. Sin embargo, funciona como debe:

```
when Movimiento =>
      if(signoX = '0' AND signoY = '0') then --Se mueve hacia arriba y la derecha
          CentroX := CentroX + 1;
          CentroY := CentroY - 1;
          if(LimSupBall = 44) then
              estado_pelota <= R
              eboteArriba;
          elsif(LimInfBall = 488) then
              estado_pelota <= ReboteAbajo;
          elsif(LimDerBall = LimIzqBarrD - 1 AND LimSupBall < LimSupBarrD AND LimInfBall > LimInfBarrD ) then
              estado_pelota <= ReboteDerecha;
          elsif(LimIzqBall = LimDerBarrI + 1 AND LimSupBall < LimSupBarrI AND LimInfBall > LimInfBarrI ) then
              estado_pelota <= ReboteIzquierda;
          elsif(LimDerBall = 769) then
              estado_pelota <= ScoreP1;
          elsif(LimIzqBall = 157) then
              estado_pelota <= ScoreP2;
          else
              estado_pelota <= estado_pelota;
          end if;
    elsif(signoX = '0' AND signoY = '1') then --Se mueve hacia derecha y hacia abajo
          CentroX := CentroX + 1;
          CentroY := CentroY + 1;
          if(LimSupBall = 44) then
              estado_pelota <= ReboteArriba;
          elsif(LimInfBall = 488) then
              estado_pelota <= ReboteAbajo;
          elsif(LimDerBall = LimIzqBarrD - 1 AND LimSupBall < LimSupBarrD AND LimInfBall > LimInfBarrD ) then
              estado_pelota <= ReboteDerecha;
          elsif(LimIzqBall = LimDerBarrI + 1 AND LimSupBall < LimSupBarrI AND LimInfBall > LimInfBarrI ) then
              estado_pelota <= ReboteIzquierda;
          elsif(LimDerBall = 769) then
              estado_pelota <= ScoreP1;
          elsif(LimIzqBall = 157) then
              estado_pelota <= ScoreP2;
          else
              estado_pelota <= estado_pelota;
          end if;
```

Figura 9. Primera parte de estado de movimiento

```
elsif(signoX = '1' AND signoY = '0') then --Se mueve hacia arriba la izquierda
          CentroY := CentroY - 1;
          CentroX := CentroX - 1;
          if(LimSupBall = 44) then
              estado_pelota <= ReboteArriba;
          elsif(LimInfBall = 488) then
              estado_pelota <= ReboteAbajo;
          elsif(LimDerBall = LimIzqBarrD - 1 AND LimSupBall < LimSupBarrD AND LimInfBall > LimInfBarrD ) then
              estado_pelota <= ReboteDerecha;
          elsif(LimIzqBall = LimDerBarrI + 1 AND LimSupBall < LimSupBarrI AND LimInfBall > LimInfBarrI ) then
              estado_pelota <= ReboteIzquierda;
          elsif(LimDerBall = 769) then
              estado_pelota <= ScoreP1;
          elsif(LimIzqBall = 157) then
              estado_pelota <= ScoreP2;
          else
              estado_pelota <= estado_pelota;
          end if;
elsif(signoX = '1' AND signoY = '1') then --Se mueve hacia abajo y hacia la izquierda
          CentroX := CentroX - 1;
          CentroY := CentroY + 1;
          if(LimSupBall = 44) then
              estado_pelota <= ReboteArriba;
          elsif(LimInfBall = 488) then
              estado_pelota <= ReboteAbajo;
          elsif(LimDerBall = LimIzqBarrD - 1 AND LimSupBall < LimSupBarrD AND LimInfBall > LimInfBarrD ) then
              estado_pelota <= ReboteDerecha;
          elsif(LimIzqBall = LimDerBarrI + 1 AND LimSupBall < LimSupBarrI AND LimInfBall > LimInfBarrI ) then
              estado_pelota <= ReboteIzquierda;
          elsif(LimDerBall = 769) then
              estado_pelota <= ScoreP1;
          elsif(LimIzqBall = 157) then
              estado_pelota <= ScoreP2;
          else
              estado_pelota <= estado_pelota;
          end if;
end if;
```

Figura 10. Segunda parte de estado de movimiento

Este fue el estado más importante, ya que en él se llevan a cabo las sumas/restas de los contadores y el chequeo constante de lo que rodeaba a la pelota.

Finalmente, se diseñaron los caracteres para imprimirse en VGA. En la figura 11 se muestra el código de uno de los caracteres que se imprimieron en la pantalla. Se siguió el mismo algoritmo para todos los números que se desplegaban También para el mensaje de victoria se utilizó este algoritmo.

```
            case (marcadorp1) is
when 0 =>
    if ((contHS > 301 AND contHS < 361) AND (contVS > 70 AND contVS < 82)) then
        RED <=  "11111111";
        GR  <=  "11111111";
        BL  <=  "11111111";
    elsif ((contHS > 301 AND contHS < 321) AND (contVS > 82 AND contVS < 94)) then
        RED <=  "11111111";
        GR  <=  "11111111";
        BL  <=  "11111111";
    elsif ((contHS > 341 AND contHS < 361) AND (contVS > 82 AND contVS < 94)) then
        RED <=  "11111111";
        GR  <=  "11111111";
        BL  <=  "11111111";
    elsif ((contHS > 301 AND contHS < 321) AND (contVS > 94 AND contVS < 106)) then
        RED <=  "11111111";
        GR  <=  "11111111";
        BL  <=  "11111111";
    elsif ((contHS > 341 AND contHS < 361) AND (contVS > 94 AND contVS < 106)) then
        RED <=  "11111111";
        GR  <=  "11111111";
        BL  <=  "11111111";
    elsif ((contHS > 301 AND contHS < 321) AND (contVS > 106 AND contVS < 118)) then
        RED <=  "11111111";
        GR  <=  "11111111";
        BL  <=  "11111111";
    elsif ((contHS > 341 AND contHS < 361) AND (contVS > 106 AND contVS < 118)) then
        RED <=  "11111111";
        GR  <=  "11111111";
        BL  <=  "11111111";
    elsif ((contHS > 301 AND contHS < 361) AND (contVS > 118 AND contVS < 130)) then
        RED <=  "11111111";
        GR  <=  "11111111";
        BL  <=  "11111111";
    end if;
```

Figura 11: Impresión del número 0

Para los marcadores, se mostraban los números del 0 al 5, tanto para el jugador 1 como para el jugador 2.

Para indicar el ganador: El mensaje P1 (o P2) WON, para lo que se diseñaron las letras P,W,O,N.

Estos marcadores funcionaron con contadores y realizaban un conteo ascendente cada vez que la máquina de estados pasaba por el estado ScoreP1 o ScoreP2 y podían reiniciarse utilizando un switch.

RESULTADOS

No se tomaron demasiadas fotos del proyecto, pues es necesario ver la interacción que tiene el jugador con el juego y es difícil capturar esto. Sin embargo, se muestran algunas de estas en las figuras 12 y 13.

Figura 12. Pantalla de victoria P1

Figura 13. Pantalla de victoria P2

A continuación se anexa la liga de un video con los resultados obtenidos:
https://youtu.be/1Svo5O66_Gw

CONCLUSIONES

Julio: El diseño, implementación y mejoras de este proyecto permitieron el refuerzo de todos los temas vistos durante el curso de Sistemas Digitales Avanzados, no cabe duda de la dificultad de utilizar VHDL para un videojuego como el pong. La parte sin duda más complicada es el movimiento de la pelota pues pone a prueba las habilidades de identificar máquinas de estados. El tema más importante en todo este proyecto es el control del VGA, pues como medio de despliegue no es nada sencillo su control.

Hugo: el proyecto en general representó un reto, pues al principio fue difícil de tratar con la visualización utilizando VGA. El uso de contadores para este proyecto fue muy importante, ya que casi todos los sistemas que lo componen se basaban en el uso de los mismos. Sin duda alguna, hacer primero un modelo en el cual la pelota pudiera controlarse manualmente fue una buena idea para el proyecto, ya que fue muy útil para arreglar problemas que tuvieran que ver con la visualización y la relación que tenía la pelota con los otros objetos. Utilizar una máquina de estados para la automatización de esta fue una gran idea; sin embargo, considero que hicieron falta algunos detalles que pudieron hacer el juego más emocionante y divertido (aumentar velocidad de la pelota).

A manera de **coevaluación**, me gustaría destacar que no considero que los integrantes hayamos trabajado de una forma equitativa, pues la repartición de trabajo estuvo lejos de ser la misma para ambos. Considero que no debemos compartir la nota final de este proyecto, ya que yo (Hugo) realicé alrededor del 80% del trabajo. Lo justo para mí sería que Julio tuviera 15-20 puntos menos a los que se obtenga como nota final del proyecto.

FUENTES CONSULTADAS:

Gaming, O. C. (2013). Arcade Game: Pong (1972 Atari) [Re-Uploaded]. Recuperado 29 novembiembre 2018, de https://www.youtube.com/watch?v=e4VRgY3tkh0&fbclid=IwAR0yxu53NiDgVLJOHaGW5ZQ_tVrUYgYaB768mantilCXQY-XlwKFEeJvhyI