PROYECTO PONG

Sebastian De La Cruz
Ingenieria Electronica
Pontificia Universidad Javeriana
Bogotá, Colombia
delacruzsebastian@javeriana.edu.co

Natalia Guevara
Ingenieria Electronica
Pontificia Universidad Javeriana
Bogotá, Colombia
guevara.pnatalia@javeriana.edu.co

Harrinson Sanguino
Ingenieria Electronica
Pontificia Universidad Javeriana
Bogotá, Colombia
juansanguino@javeriana.edu.co

***Abstract*— This report shows the implementation of the game "pong" which consists of two rackets with a ball that will bounce on the screen until one of the rackets does not hit the ball, for the visualization of this a VGA screen was used, to achieve a correct visualization was carried out a sweep of the 60 Hz screen generating stable images for the human eye, finally it was possible to show a stable game with a counter of up to 7 points, after that the game will be reset to start its operation again.**

I. INTRODUCCIÓN

En este proyecto se realizó un sistema digital para el manejo de una pantalla VGA y hacer representaciones de figuras para así conformar el videojuego pong.

Analizamos la estructura que tendría el pong y se realizó el diseño de los respectivos bloques principales que este tendría, obteniendo así un orden de esta estructura iniciando con la visualización en la pantalla VGA hasta la estructuración del puntaje basándonos en el choque de la pelota con las raquetas.

Asi lograr el pong se llevó a cabo el siguiente desarrollo: Primero se buscó una forma de visualizar elementos en la pantalla, para esto se realizó un barrido de forma horizontal y otro de forma vertical teniendo esto en base a los ciclos de reloj que tardaran estos, luego se tomaban rangos de bits a los cuales se realizaron modificación en el color para diferenciarse del resto de la pantalla, de esta forma se logró mostrar algo en la pantalla, luego se realizó una máquina de estados para mostrar una raqueta con movimiento con once casos diferentes donde se modificó el rango de bits con distintos colores, para la pelota se realizó una máquina de estados con cuatro casos (arriba_adelante, abajo_adelante, arriba_atras, abajo_atras) se inicializo en uno de los casos y de esta forma seguía su trayectoria, en los bordes se configuro los estados para cambiar la dirección de forma correcta, después se configuro el choque de la pelota con la raqueta con dos contadores para determinar el puntaje de los jugadores y poder saber cuándo resetear nuestro juego.

A continuación del documento se explicarán detalladamente los bloques y las máquinas de estados que componen el pong, viendo, así como se realizó la configuración de este para el correcto funcionamiento del proyecto, mostrando detalladamente el control de la pantalla VGA lo cual se podría decir que era el principal reto de este.

II. MARCO TEÓRICO

PONG

Este consiste videojuego para dos personas que tenga tres elementos móviles y varios fijos de forma que interactúen entre ellos a partir de las órdenes que introduzcan ambos usuarios. Se debe realizar con joystick y una interfaz VGA. El diseño se describe en VHDL.

Ambos jugadores disponen de una raqueta con la que golpear una pelota que puede rebotar o no contra las paredes superior e inferior. La pelota no debe sobrepasar la línea de fondo de cada jugador. Si lo hace, el jugador contrario gana un punto.

FPGA

Una matriz de puertas programables o FPGA (del inglés field-programmable gate array), es un dispositivo programable que contiene bloques de lógica cuya interconexión y funcionalidad puede ser configurada en el momento, mediante un lenguaje de descripción especializado. La lógica programable puede reproducir desde funciones tan sencillas como las llevadas a cabo por una puerta lógica o un sistema combinacional hasta complejos sistemas en un chip.

MÁQUINAS DE ESTADO FINITO EN VHDL

Las máquinas de estado finito, más conocidas por su acrónimo en inglés FSM (Finite State Machine), se utilizan ampliamente en el diseño de circuitos digitales (además de en otros ámbitos de la ingeniería, como la programación), para describir el comportamiento de un sistema según el valor de sus entradas y de cómo van cambiando en el tiempo. Ésta es una definición parcial pero que nos permite hacernos una primera idea intuitiva. Desde el punto de vista de las FSM, un sistema está compuesto

de *estados*por los que va pasando el sistema, de *señales de entrada* que modifican esos estados y de *señales de salida* que pueden utilizarse para conocer el estado del sistema y actuar en consecuencia. Un ejemplo muy visual podría ser un semáforo, el cuál dispone de tres estados diferentes, uno para cada color. Las entradas del sistema las podría generar un temporizador que activa una señal cada cierto tiempo, indicando que hay que pasar al siguiente estado. Por último, las salidas del sistema podrían ser tres señales que indiquen qué lámpara, de las tres disponibles, tiene que encenderse.

PANTALLA VGA

Los Monitores VGA (siglas en inglés de *Video Graphics Array*) de aplicación universal en todos los ordenadores e incluyéndose inclusive en el mundo de los Dispositivos Portátiles.
Originalmente fue utilizado por la compañía IBM, presentando monitores que contaban con una resolución nativa de 640 x 480 píxeles, siendo un estándar de gráficos que se empleaba en su línea de ordenadores IBM PC, y pasando a ser la funcionalidad mínima que debe tener el Hardware Gráfico del ordenador, inclusive previo al inicio del sistema operativo.
Esto es visible en el caso de los sistemas operativos de Microsoft por ejemplo cuando estamos dando inicio al Microsoft Windows, siendo la pantalla de carga presentada en una resolución de 640 x 480 píxeles, sin demasiada profundidad de color y bastante simple, para luego adaptarse a la configuración de pantalla que ha elegido el usuario mediante el sistema operativo.
Este estándar de gráficos pretendió ser reemplazado en su momento con la instauración del Estándar XGA (siglas en inglés de *Extended Gráphics Array*, reemplazándose la E por la X para no confundir con EGA, *Enhanced Graphics Adapter*) aunqu en la actualidad contamos con distintas extensiones que presentan variaciones, que son conocidas bajo el nombre de Súper VGA, que cubre un amplio espectro de estándares.
El conector que utiliza el Estándar VGA cuenta con una conexión de tres hileras de 15 pines, teniendo las variantes de DDC2, DE-9 y una tecnología de última generación conocida como Mini-VGA que es utilizada en Dispositivos Portátiles, siendo en todo caso adoptado universalmente por las Tarjetas Gráficas tanto como las conexiones de los Monitores disponibles en el mercado.
Los cableados de los Conectores VGA cuentan con el transporte de datos correspondiente a la modalidad Analógica RGBHV (siglas de Rojo, Verde, Azul, Horizontal y Vertical, siendo estos últimos destinados a la sincronización de la imagen) además de señales de video de tipo DDC2 junto a los canales de Datos y Reloj Digital.

CONVERSOR ADC

Un conversor analógico-digital , (ADC del inglés "Analog-to-Digital Converter") es un dispositivo electrónico capaz de convertir una entrada analógica de voltaje en un valor binario, Se utiliza en equipos electrónicos como ordenadores, grabadores de sonido y de vídeo, y equipos de telecomunicaciones.

La señal analógica, que varía de forma continua en el tiempo, se conecta a la entrada del dispositivo y se somete a un muestreo a una velocidad fija. La digitalización consiste básicamente en realizar de forma periódica medidas de la amplitud (tensión) de una señal, redondear sus valores a un conjunto finito de niveles preestablecidos de tensión (conocidos como niveles de cuantificación) y registrarlos como números enteros en cualquier tipo de memoria o soporte. Los procesos que dan lugar a esta conversión son el muestreo, la retención, la cuantificación y la codificación:

Muestreo: el muestreo (en inglés, sampling) consiste en tomar muestras periódicas de la amplitud de onda. La velocidad con que se toma esta muestra, es decir, el número de muestras por segundo, es lo que se conoce como frecuencia de muestreo. El ingeniero sueco Harry Nyquist formuló el siguiente teorema para obtener una grabación digital de calidad. “La frecuencia de muestreo mínima requerida para realizar una grabación digital de calidad, debe ser igual al doble de la frecuencia de audio de la señal analógica que se pretenda digitalizar y grabar”. (“Condición de Nyquist”). Sino cumplimos este requisito aparecerá el fenómeno de aliasing el cual propiciará la aparición de frecuencias "alias", y la señal original no puede ser reconstruida de forma unívoca a partir de la señal digital.

Retención (en inglés, hold): las muestras tomadas han de ser retenidas (retención) por un circuito de retención (hold), el tiempo suficiente para permitir evaluar su nivel (cuantificación). Desde el punto de vista matemático este proceso no se contempla, ya que se trata de un recurso técnico debido a limitaciones prácticas, y carece, por tanto, de modelo matemático.

Cuantificación: en el proceso de cuantificación se mide el nivel de voltaje de cada una de las muestras. Consiste en asignar un margen de valor de una señal analizada a un único nivel de salida. Error de cuantificación: incluso en su versión ideal, añade, como resultado, una señal indeseada a la señal de entrada: el ruido de cuantificación: señal en tiempo discreto y amplitud continua que resulta de igualar los niveles de las muestras de amplitud continua a los niveles de cuantificación más próximos.

Codificación: la codificación consiste en traducir los valores obtenidos durante la cuantificación al código binario. Hay que tener presente que el código binario es el más utilizado, pero también existen otros tipos de códigos que también son utilizados.

III. Diseño del Sistema

El diseño de nuestro sistema tiene como entradas dos joysticks que con ayuda de un adc conectamos a la FPGA además tenemos un botón de reset, otro de start y un reloj que en nuestro caso es el propio de la tarjeta. Como salidas tenemos dos displays 7 segmentos, y las entradas de nuestra pantalla VGA que fuero un hsync, un vsync, un R, un G y un B estos últimos (RGB) que nos dan el color a visualizar. Con estas entradas y salidas realizamos un diseño funcional de un juego pong que fue segmentado en bloques de la siguiente forma un control paleta1 , un control paleto dos, un bloque de visualización de la bola que me da la posición de la bola en cada momento, un bloque para los límites de la bola, un bloque de puntaje del jugador 1,otro del jugador 2, y un pequeño boque que me genera un reloj de menos frecuencia. Los siguientes bloques que se observan en la siguiente figura son explicados claramente más adelante.

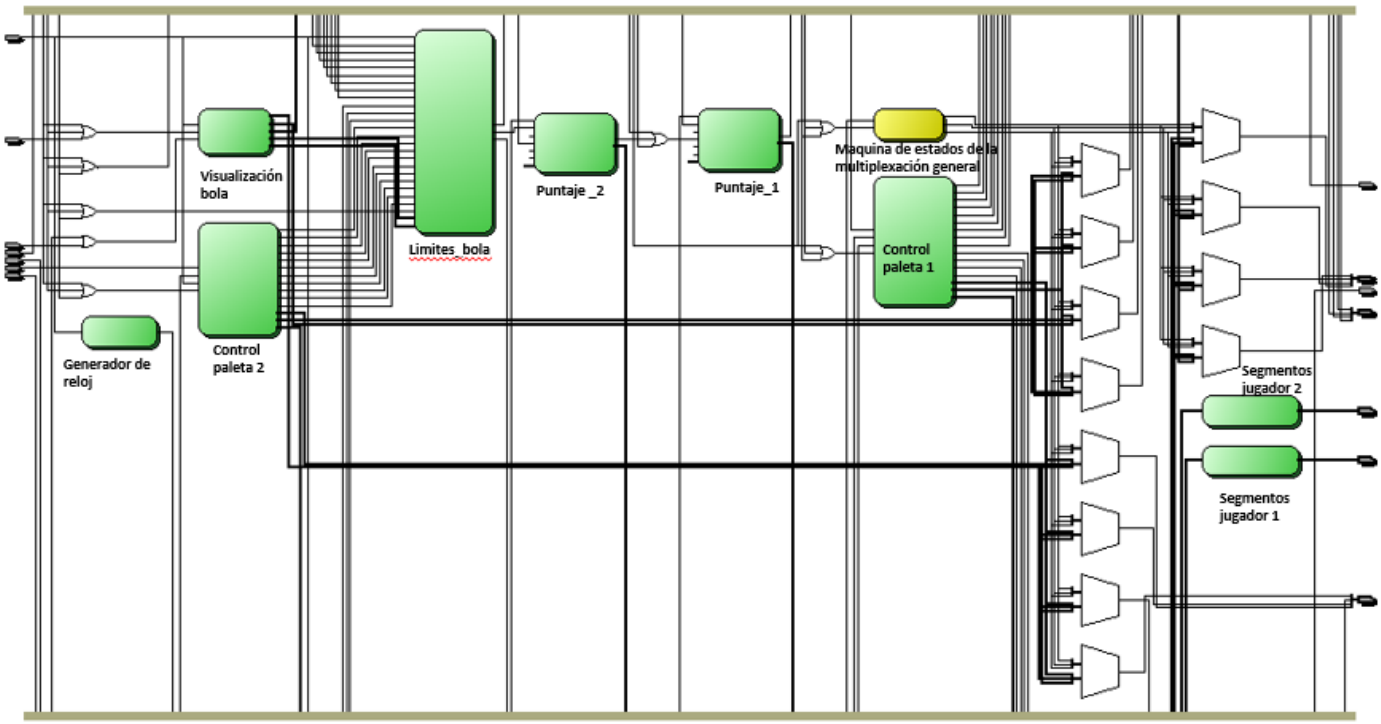

Fig. 1. Diagrama de bloques general generada por Quartus.

A. *Descripción de Bloques*

1. Digital_clock_top

Este bloque toma el reloj de la tarjeta y lo multiplica por cierto periodo para obtener nuevos relojes con tiempos más prolongados, los cuales se establecerán dependiendo de la necesidad en el proyecto.

Fig. 2. Digital_clock_top. Entradas/salidas. generado por Quartus.

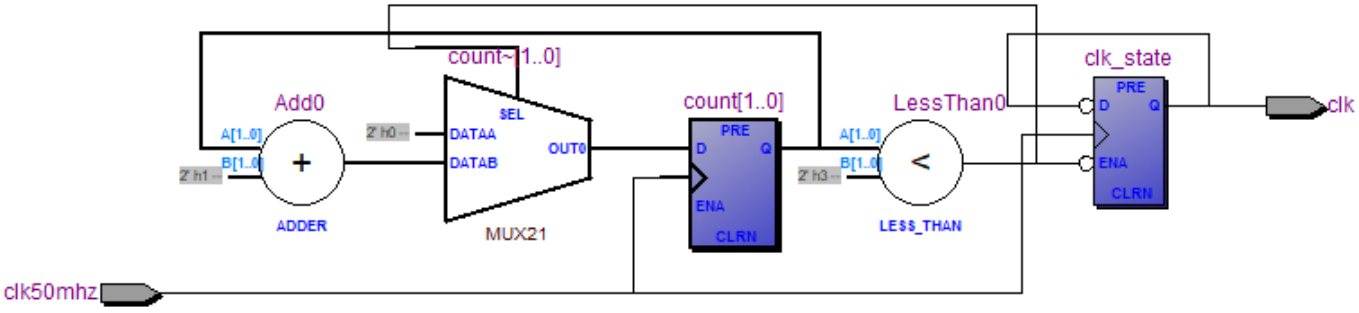

Fig. 3. Digital_clock_top generado por Quartus.

2. Control paleta

Para el siguiente bloque contamos con cuatro entradas, un reset, un clk, y dos bits que serán valores dados por los joysticks, estos dos bits están conectados a una máquina de estados llamada máquina de estados de la posición de la raqueta ubicada y explicada en la figura 40 y en la figura 41 que me brindaran la posición actual de la paleta en nuestro diseño creamos once posiciones predeterminadas como salidas de esta máquina de estados, estas salidas están conectadas como selectores a un multiplexor que decide cuál de las salidas de mis bloques de visualización mostrar.

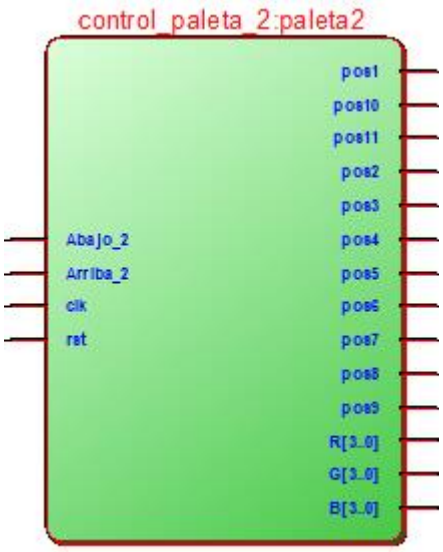

Fig. 4.control_paleta. Entradas/salidas generado por Quartus.

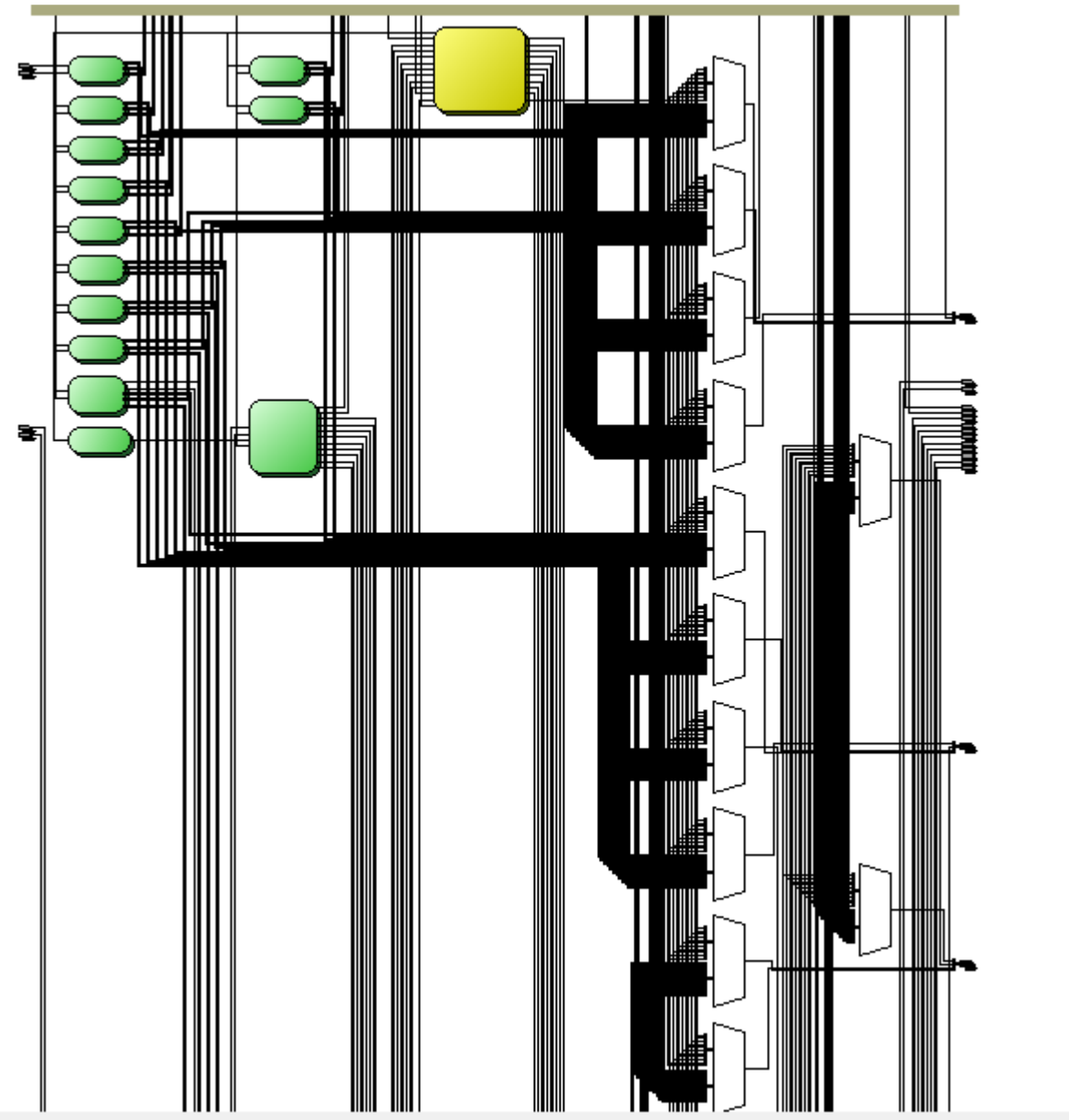

Fig. 5. control_paleta generado por Quartus.

3. Visualización VGA

Este bloque es el encargado de sacarme las señales necesarias para visualizar en una pantalla VGA dentro de este encontramos varios sub-bloques como se evidencia en la figura 7.

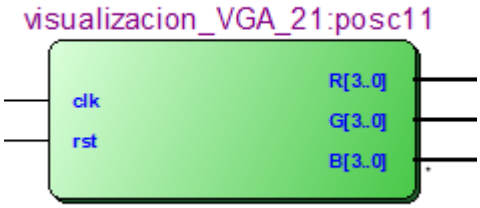

Fig. 6. Visualizacion_VGA. Entradas/salidas generado por Quartus.

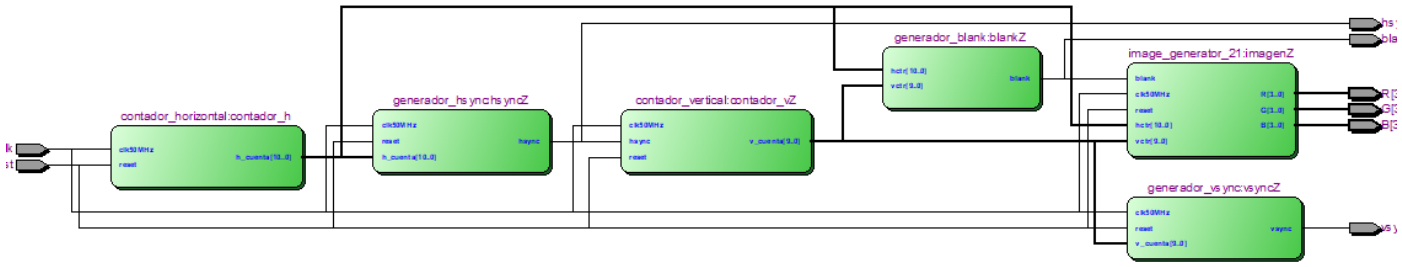

Fig. 7. Visualizacion_VGA generado por Quartus.

3.1 contador_horizontal

Este bloque genera un contador de ciclos de reloj que va desde 0 a 1587 ya que 1587 ciclos de reloj son 31,743us que sería el tiempo que tardo en recorrer una línea horizontal en nuestra pantalla ,para este contador se necesitan 11bits, y este bloque me da este contador como salida.

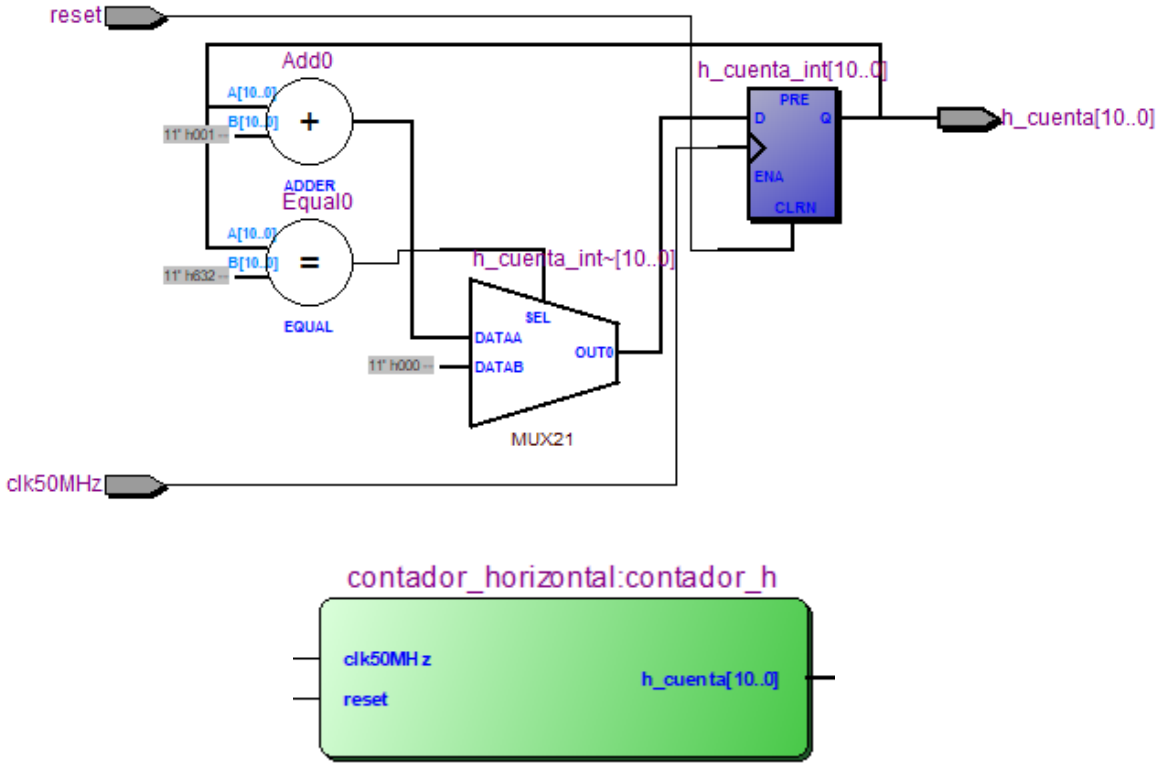

Fig. 8. Contador_horizontal. Entradas/salidas generado por Quartus.

Fig. 9. Contador_horizontal.generado por Quartus.

3.2 contador_vertical

Este bloque genera un contador de ciclos de reloj que va desde 0 a 529 ya que 529 ciclos de reloj son 16,784 que sería el tiempo que tardo en recorrer una línea vertical en nuestra pantalla, en binario para lo que se necesitan 10bits y este bloque me da este contador como salida.

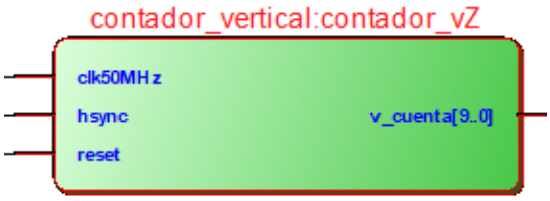

Fig. 10. Contador_vertical. Entradas/salidas generado por Quartus.

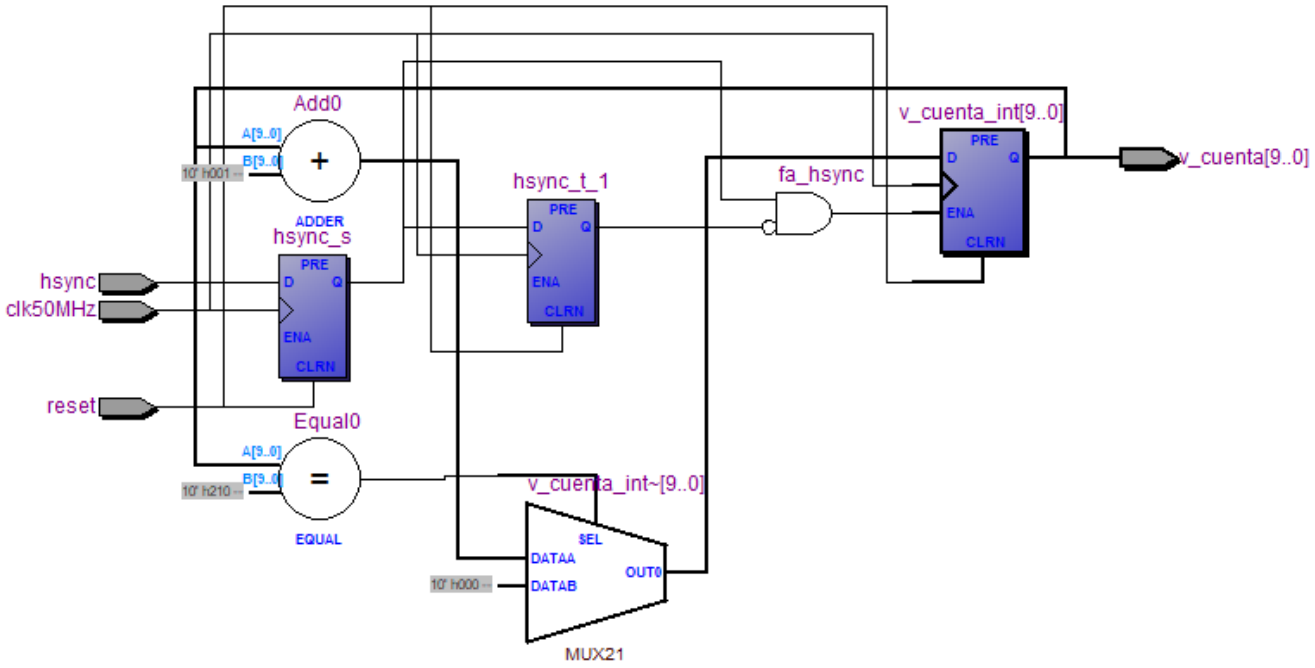

Fig. 11. Contador_vertical generado por Quartus.

3.3 generador_hsync

Este bloque me genera una señal (hsync)que se comporta de la siguiente manera:
El pulso negativo de la señal sincronismo horizontal (*hsync*), marca el inicio y elfinal de una línea y asegura que el monitor muestra los pixeles entre los bordes izquierdoy derecho de la parte visible del monitor. Los pixeles reales se envían al monitor en una ventana de 25,17µs.La señal de sincronismo horizontal (*hsync*) se pone a cero como mínimo 0,94µs después del último pixel y se mantiene a cero 3,77µs. Una nueva línea de pixeles se puede iniciar1,89µs después de que la señal(*hsync*) ha terminado, es decir, se ha puesto a uno. Como una línea ocupa 25,17µs de los 31,77µs que dura la línea de video, esto significa que en los restantes 6,6µs la pantalla está oscura, siendo este tiempo el que se conoce como horizontal *blanking interval*.

Fig. 12. Generador_hsync. Entradas/salidas generado por Quartus.

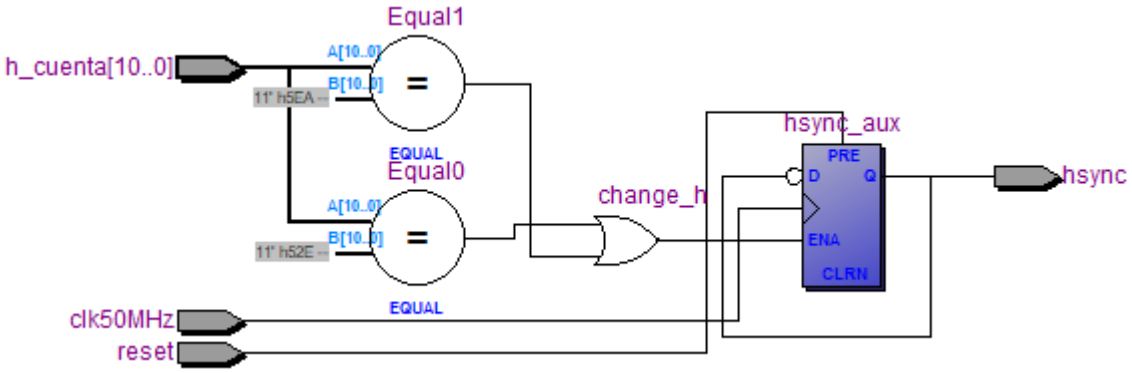

Fig. 13. Generador_hsync generado por Quartus.

3.4 generador_vsync

Este bloque me genera una señal(vsync) que se comporta de la siguiente manera:
La señal de sincronismo vertical (vsync)se pone a cero como mínimo 0,45ms después de la última línea y se mantiene a cero 64µs. La primera línea del siguiente frame se puede iniciar 1,02ms después de que la señal vsync ha terminado, es decir, se ha puesto a uno. Como un frame ocupa 15,25ms de los 16,784ms, esto hace que, durante 1,534ms la pantalla esté oscura, siendo este tiempo el que se conoce como vertical blanking interval.

Fig. 14. Generador_vsync. Entradas/salidas generado por Quartus.

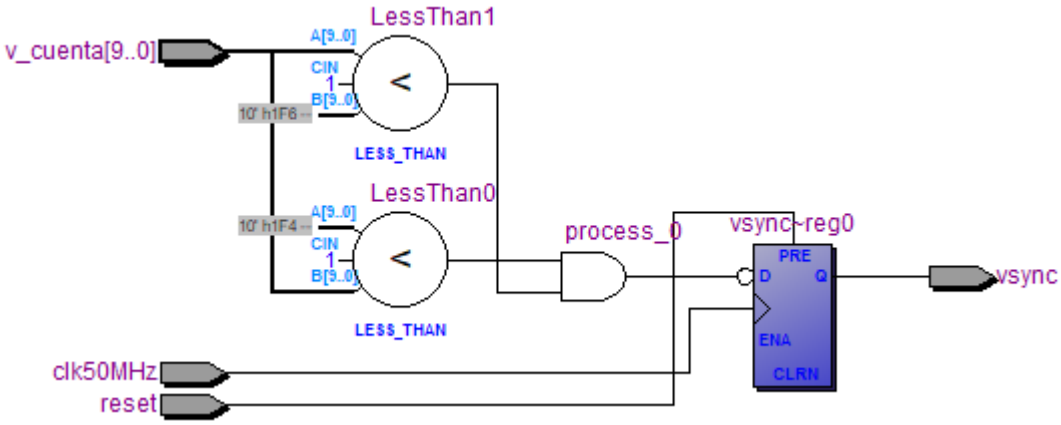

Fig. 15. Generador_vsync. Entradas/salidas generado por Quartus.

3.5 generador_blank

En este bloque se plantean los límites de mi pantalla con lo que se genera la parte negra de la pantalla que nunca va a cambiar.

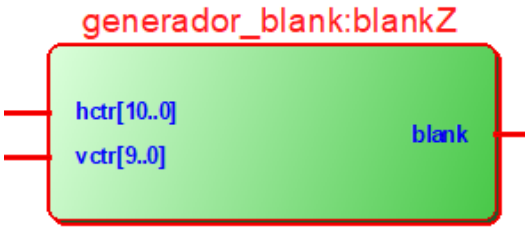

Fig. 16. Generador_blank. Entradas/salidas generado por Quartus.

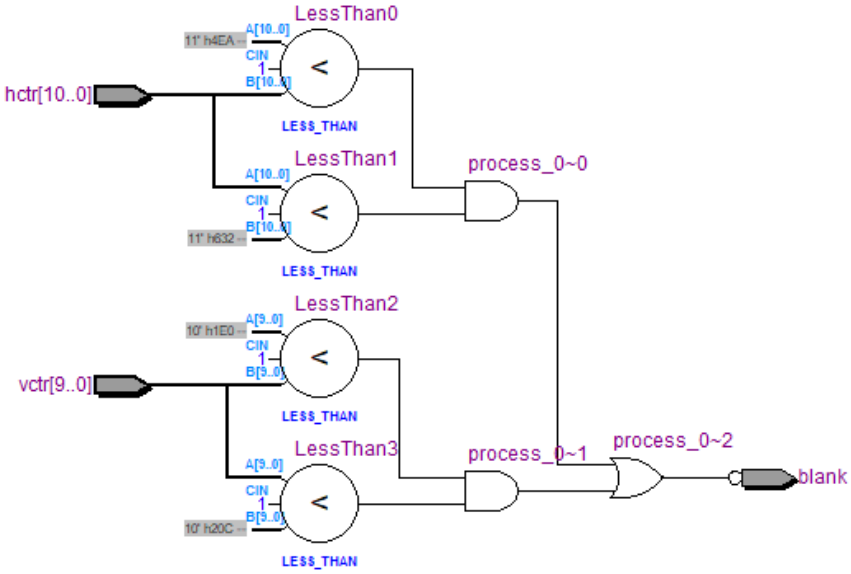

Fig. 17. Generador_blank generado por Quartus.

3.6 image_generator

Gracias a los contadores horizontal y vertical hechos con anterioridad en este bloque manejamos como salida tres bits de RGB utilizando los contadores como espacios en la pantalla y de esta manera poder mostrar varias cosas en la pantalla como se observa en la figura 20.

Fig. 18. Image_generator. Entradas/salidas generado por Quartus.

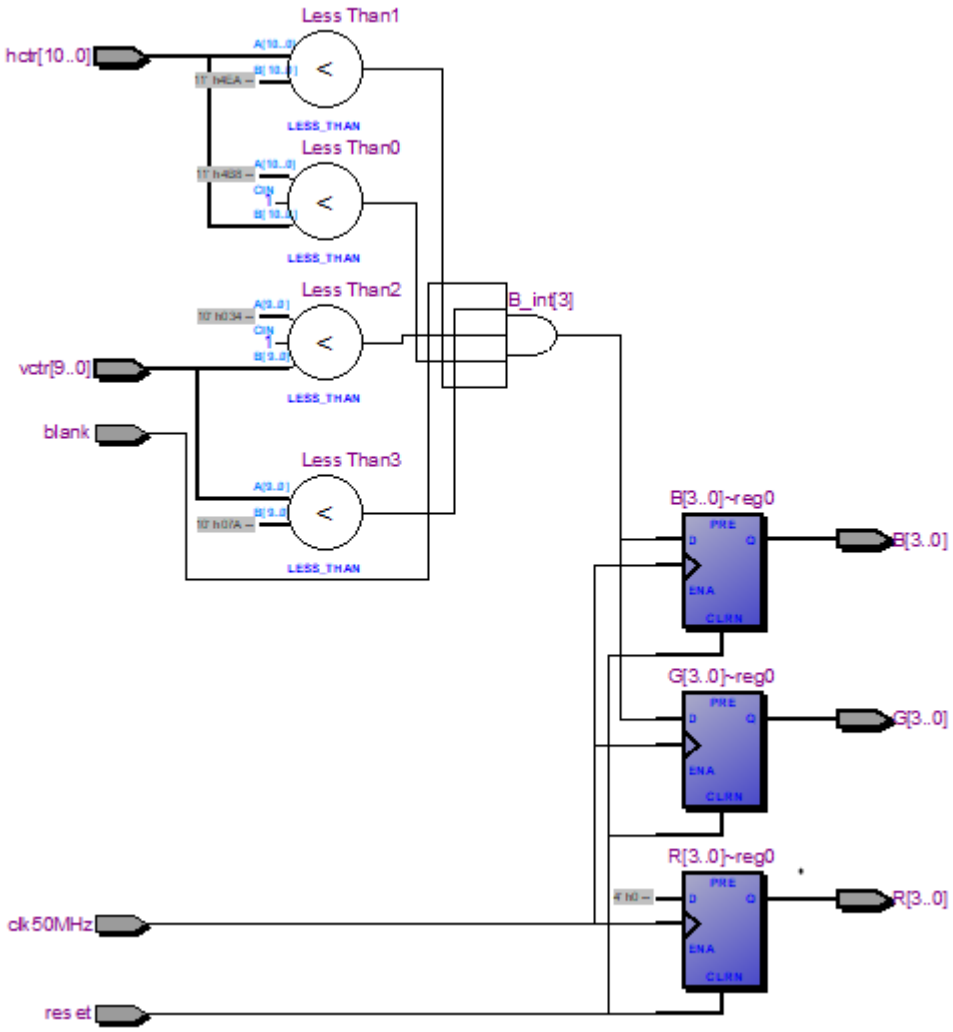

Fig. 19. Image_generator generado por Quartus.

```
color <=
"101" WHEN ((hctr_int >= 1208)   AND (hctr_int < 1258)   AND (vctr_int >= 227)   AND (vctr < 297) AND (blank = '1')) ELSE
"000"; -- Intérvalos blank (blank = 0)

R_int(0) <= color(2);
R_int(1) <= color(2);
R_int(2) <= color(2);
R_int(3) <= color(2);
G_int(0) <= color(1);
G_int(1) <= color(1);
G_int(2) <= color(1);
G_int(3) <= color(1);
B_int(0) <= color(0);
B_int(1) <= color(0);
B_int(2) <= color(0);
B_int(3) <= color(0);
```

Fig. 20. Codigo en VHDL del image_generator

4. visualización_VGA_bola

Con base en el bloque de visualización explicado en el inciso anterior diseñamos un bloque similar para la visualización de la bola, a diferencia de ese, este no tiene posiciones predefinidas pues tiene dos registros(uno para cada eje) los cuales se actualizan con cada ciclo de reloj en este obviamos las salidas hsync y vsync pues para la visualización no se puede modificar estas señales, por lo tanto se nos hizo sencillo solo cambiar el image_generator, agregar los dos registros y conectarlos a una pequeña máquina de estados que me genera la siguiente posición.

Fig. 21. Visualizacion_VGA_b. Entradas/salidas generado por Quartus.

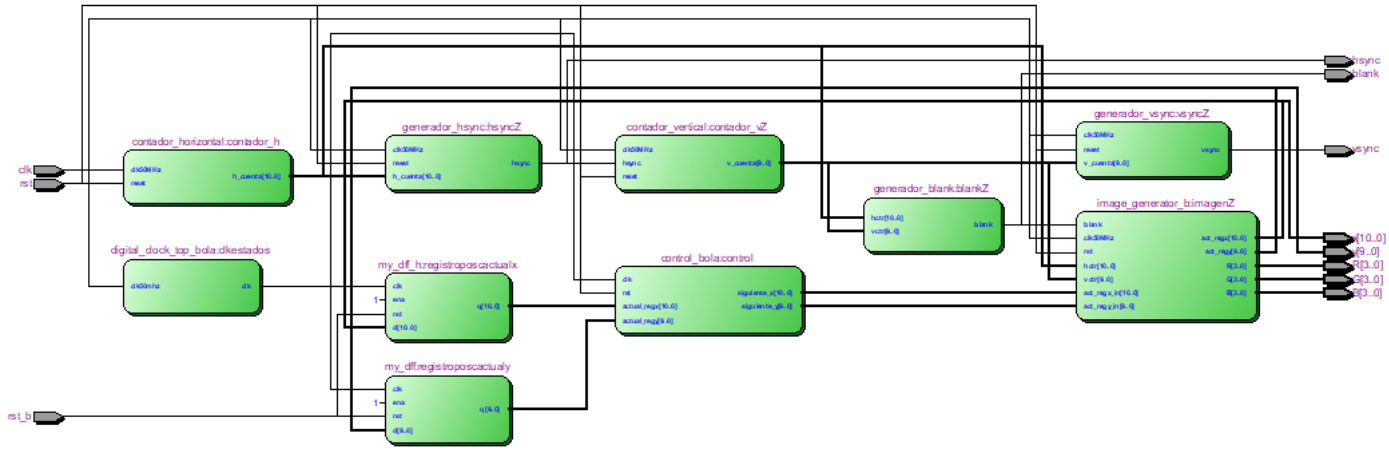

Fig. 22. Visualizacion_VGA_b generado por Quartus.

4.1 my_dff_h

Este bloque es el encargado de guardar una posición en x por un ciclo de reloj lo que nos ayuda a actualizar la posición de la bola.

Fig. 23. My_dff_h. Entradas/salidas generado por Quartus.

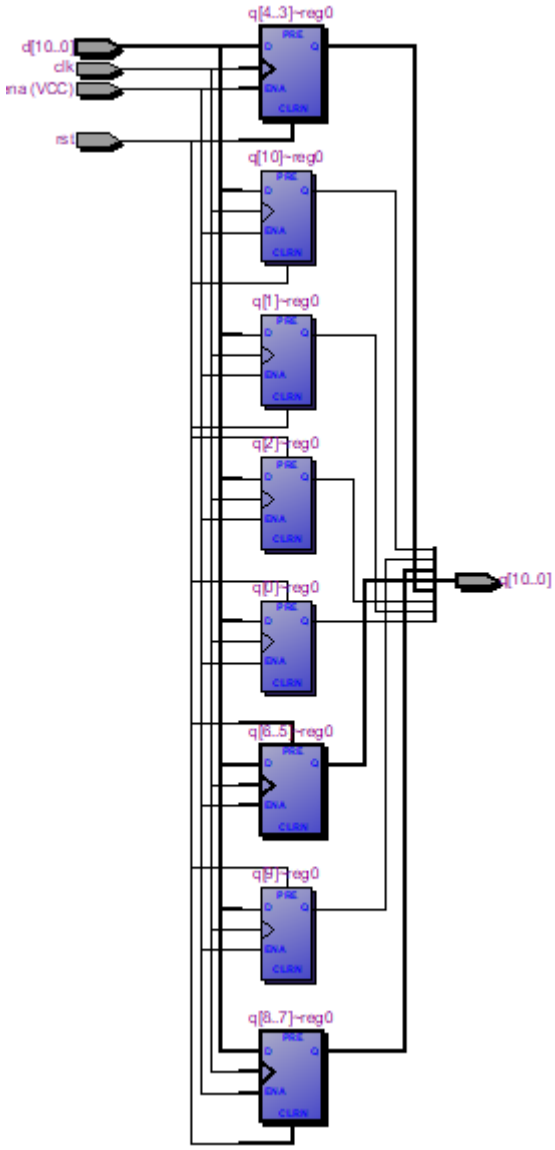

Fig. 24. My_dff_h generado por Quartus.

4.2 my_dff

Este bloque es el encargado de guardar una posición en y por un ciclo de reloj lo que nos ayuda a actualizar la posición de la bola.

Fig. 25. My_dff. Entradas/salidas generado por Quartus.

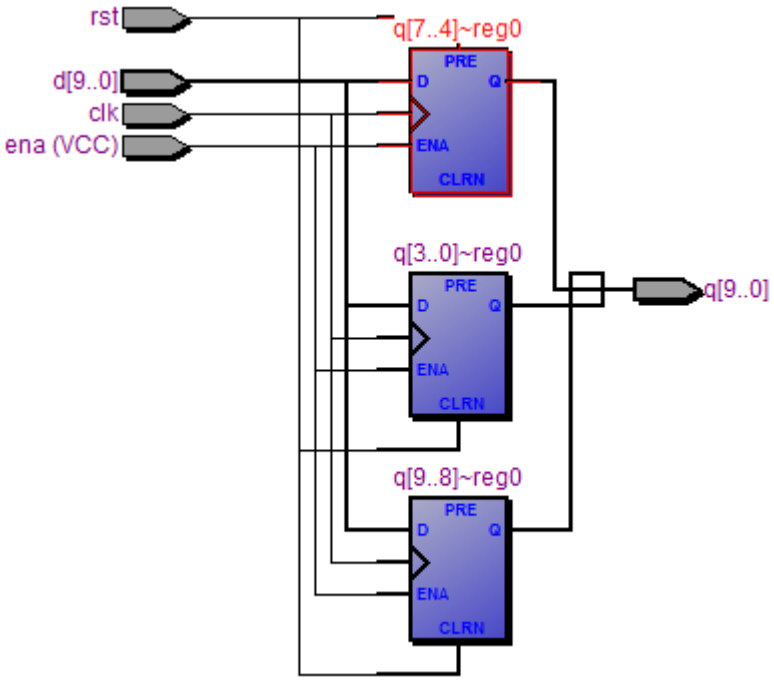

Fig. 26. My_dff generado por Quartus.

4.3 control_bola

En este bloque se genera la siguiente poscisicion mediante una maquina de estadosque se puede observar en la Fig. 42. Maquina de estados finitos del control de las direcciones de la bola. Mas adelante está será explicada en detalle.

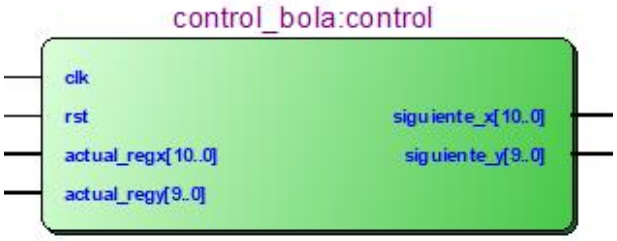

Fig. 27. Control_bola. Entradas/salidas generado por Quartus.

4.4 image_generator_b

Teniendo como base nuestro image_generator, los cambios que realizamos para este bloque fueron agregar como entrada la posición que sale de los registros los que nos actualiza la posición y agregar esta misma posición como salida para que luego pasen por mi registro y se actualicen nuevamente.

Fig. 28. Image_generator. Entradas/salidas generado por Quartus.

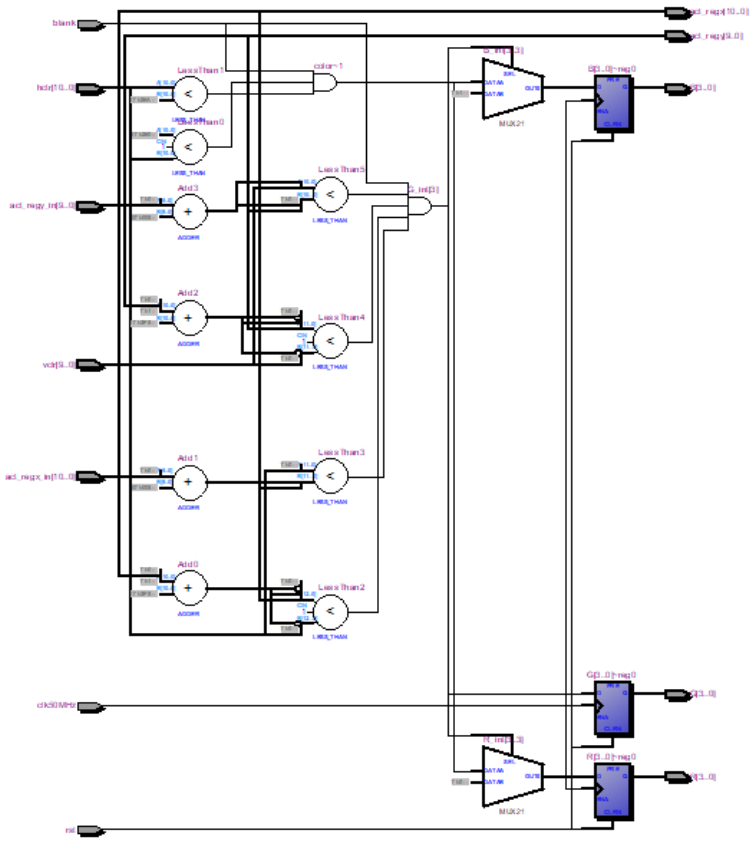

Fig. 29. Image_generator generado por Quartus.

5. limites_bola

Dependiendo de las posiciones de la paleta, de las posiciones de la bola, de dos sub-bloques y una máquina de estados finitos de los límites de la bola, saldrá una señal que nos indica si alguno de los jugadores perdió y está irá al contador de puntaje, la señal de rst_b va a una or con el start, ya que si alguno de los dos jugadores perdió la bola se reinicia y vuelve a su lugar de origen la cual es en el centro de la pantalla.

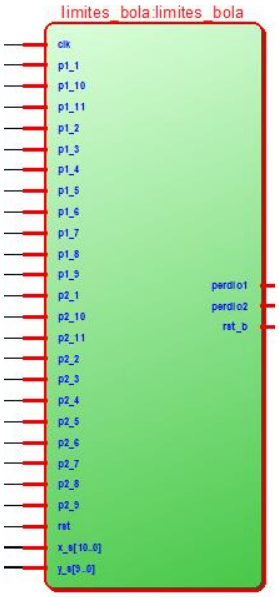

Fig. 30. Limites_bola. Entradas/salidas generado por Quartus.

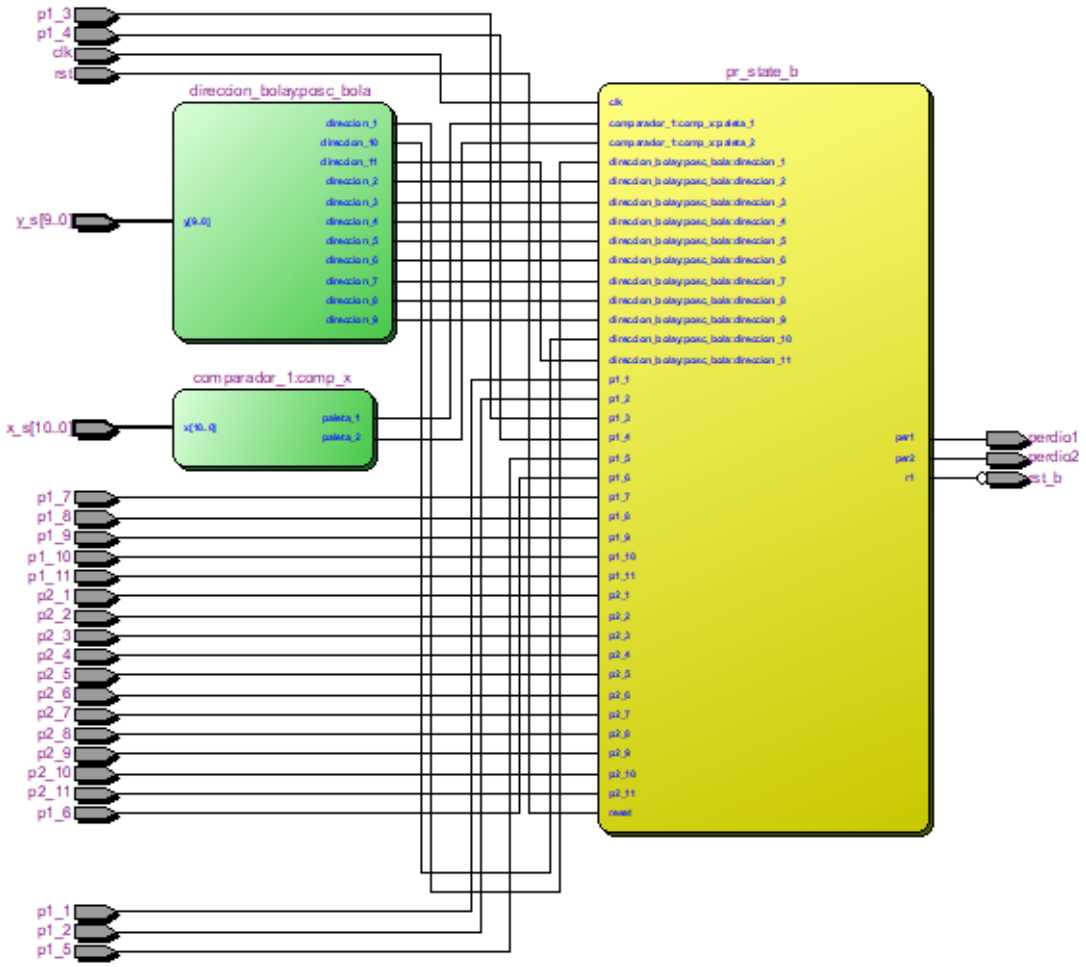

Fig. 31. Limites_bola generado por Quartus.

5.1 comparador _1

En este bloque, dependiendo de la posición en el eje x en el que se encuentre la bola, este sacará dos señales las cuales son una para cada raqueta y estás se pondrán en uno si la bola se encuentra ubicada en alguno de los límites de las raquetas en el eje x.

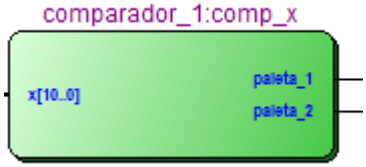

Fig. 32. Comparador_1. Entradas/salidas generado por Quartus.

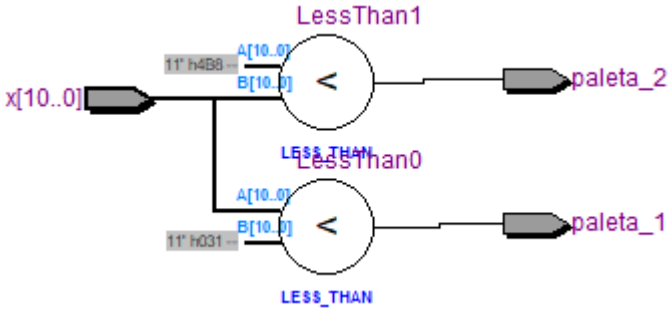

Fig. 33. Comparador_1 generado por Quartus.

5.2 direccion_bolay

En este bloque, dependiendo de la posición en el eje y en la cual se encuentre la bola, este sacará una señal la cual indicará con cuál de las posiciones de la paleta coincide

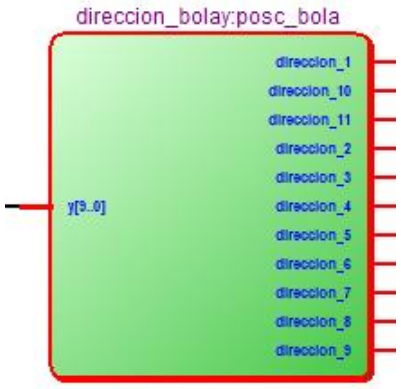

Fig. 34. Direccion_bolay. Entradas/salidas generado por Quartus.

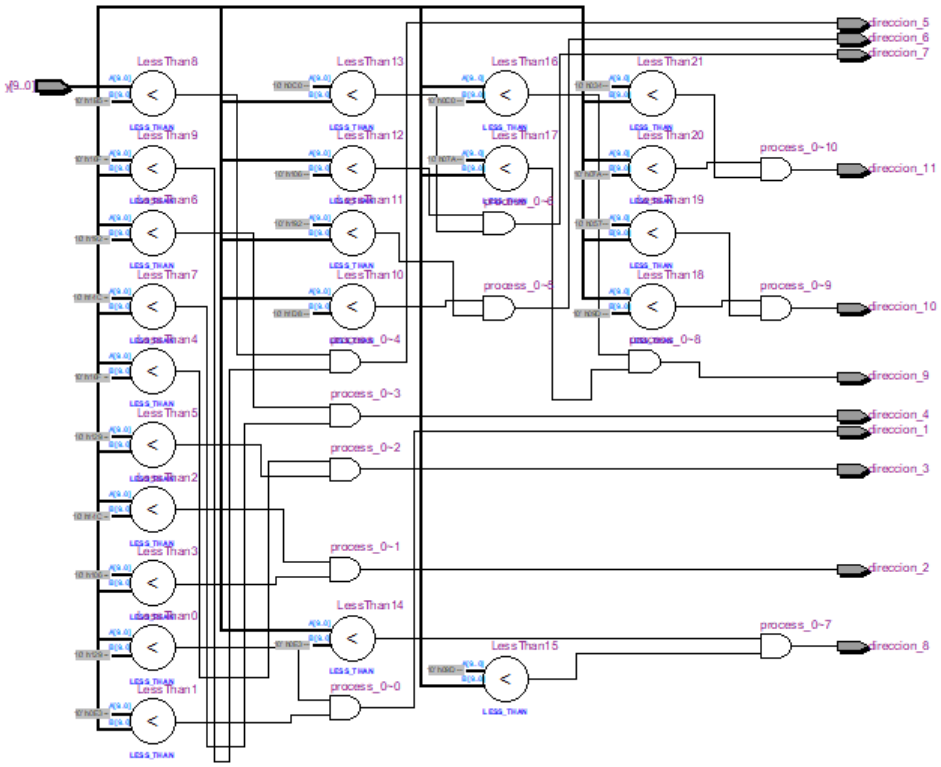

Fig. 35. Direccion_bolay generado por Quartus.

6. puntaje

En este bloque se tendrá en la entrada enable las señales: perdió 1 y perdió 2 las cuales se activarán cuando una de las raquetas no golpee la pelota, ambos contadores van hasta siete, cuando uno de los contadores llegue a siete max_tick se pondrá en uno y activara el reset de la pelota, las dos paletas y ambos puntajes, también se tendrá otra salida que será counter este va a ser el puntaje que se inicializara en 0 y aumentara de 1 en 1 cada vez que el rival pierda.

Fig. 36. puntaje. Entradas/salidas generado por Quartus.

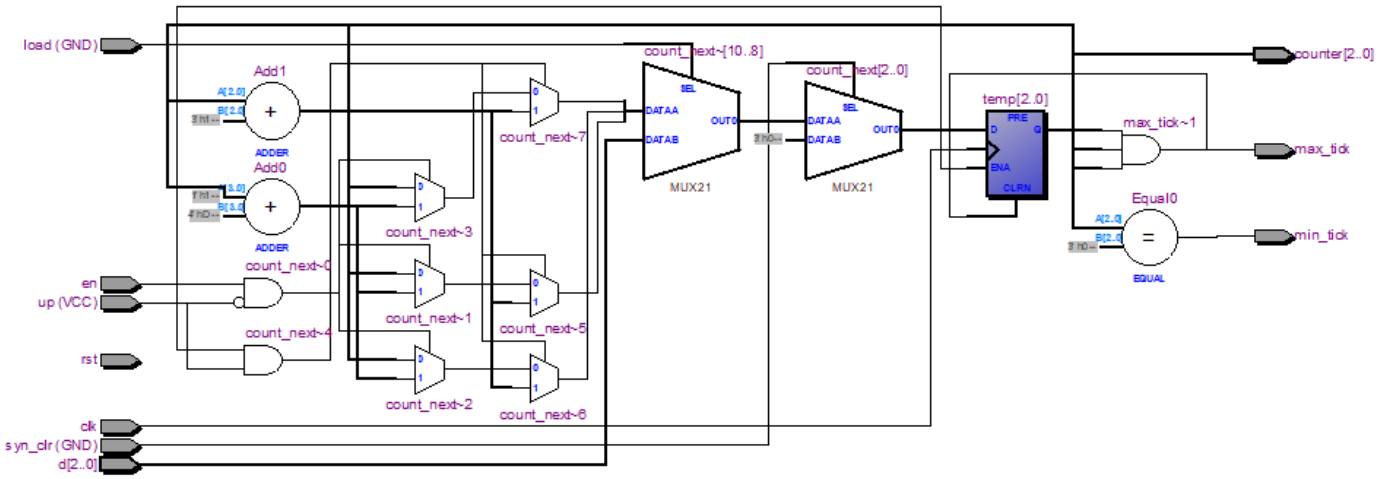

Fig. 37. puntaje generado por Quartus.

7. segmentos

En este bloque tendremos de entrada el counter de los puntajes, donde se visualizará este en el display de la tarjeta.

Fig. 38. segmentos. Entradas/salidas generado por Quartus.

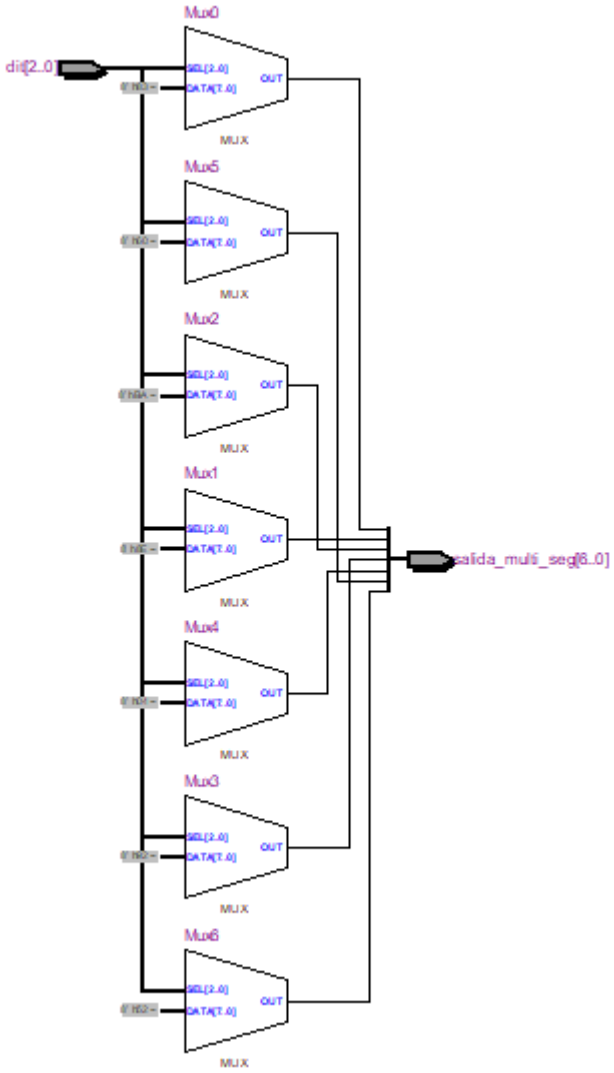

Fig. 39. segmentos generado por Quartus.

B. Máquinas de estados

En este sistema digital se implementaron cinco máquinas de estados diferentes. Dos de estas son usadas para el movimiento de las raquetas, las otras dos para el movimiento de la bola y la última para multiplexar lo que se quiere mostrar en la pantalla.

En la Figura 40 podemos observar la máquina de estados de las posiciones de las raquetas que nos genera Quartus, esta depende si la raqueta se mueve arriba o abajo y de la posición en la que estaba anteriormente.

En la Figura 41 podemos observar la máquina de estados del control de las raquetas que nos genera Quartus, esta máquina de estados simplemente es para verificar en la posición actual en la que se encuentra la raqueta sin importar donde estuvo anteriormente.

En la Figura 42 podemos observar la máquina de estados del control de las direcciones de la bola, dependiendo de la posición actual de la bola y en la dirección de la cual venia, si choca con alguno de los limites este cambia de dirección.

En la Figura 43 podemos observar la máquina de estados de los límites de la bola, la cual depende de la ubicación de la bola y de la raqueta, dependiendo de esto avisa si alguno de los dos jugadores perdió.

En la Figura 44 podemos observar la máquina de estados de la multiplexación que nos genera Quartus, esta máquina de estados sirve para poder visualizar en la pantalla las dos raquetas y la pelota con un reloj lo suficientemente rápido para que siempre se vean estos tres elementos encendidos.

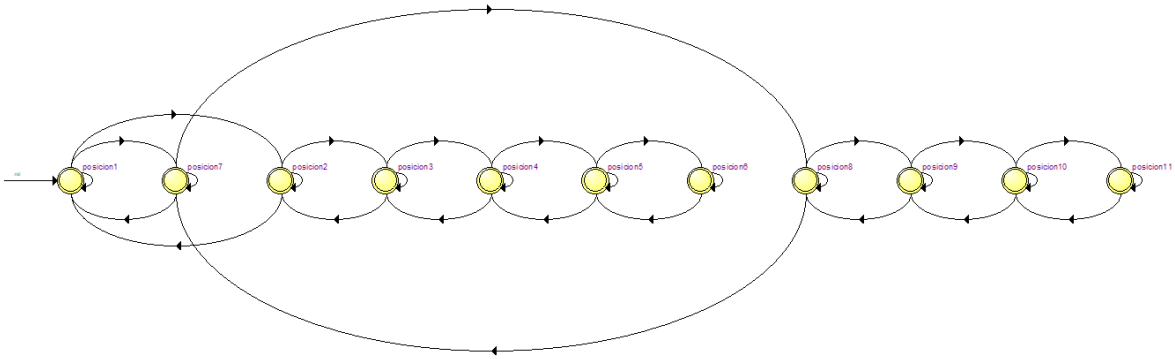

Fig. 40. Maquina de estados finitos de las posiciones de las raquetas generada por Quartus.

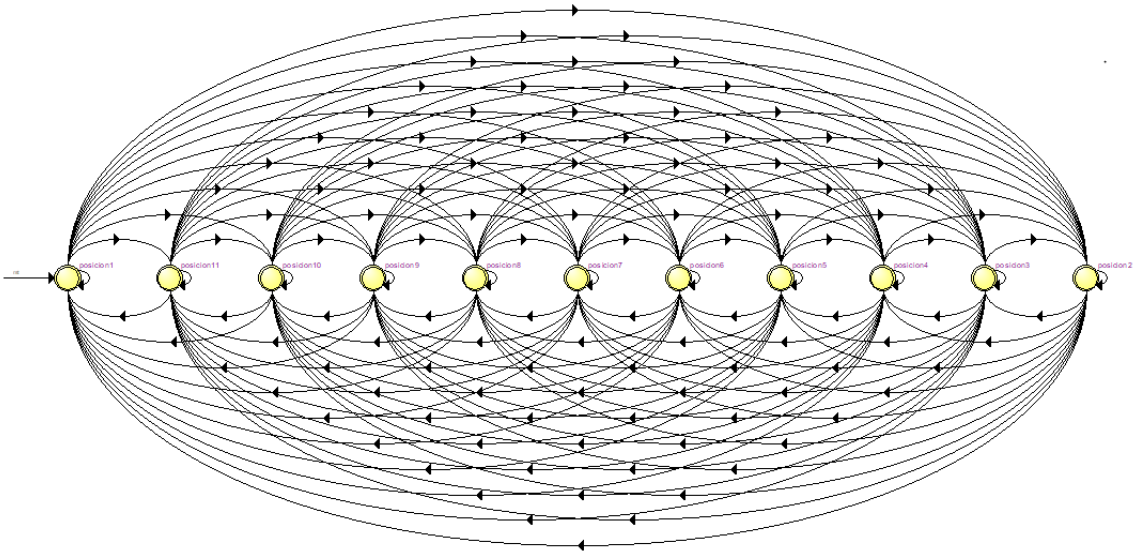

Fig. 41. Maquina de estados finitos del control de las raquetas generada por Quartus.

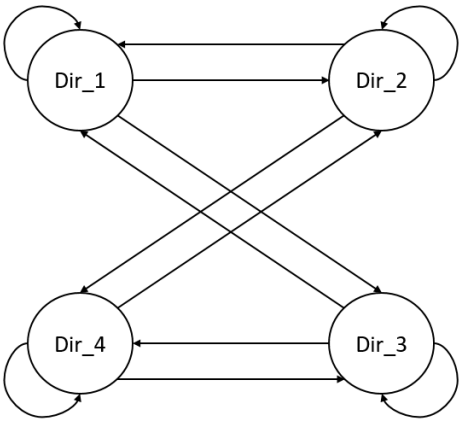

Fig. 42. Maquina de estados finitos del control de las direcciones de la bola.

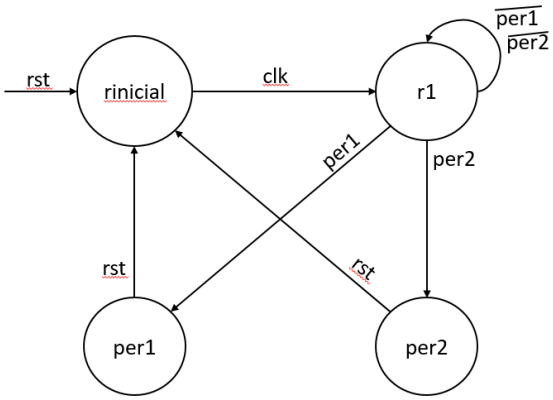

Fig. 43. Maquina de estados finitos de los limites de la bola.

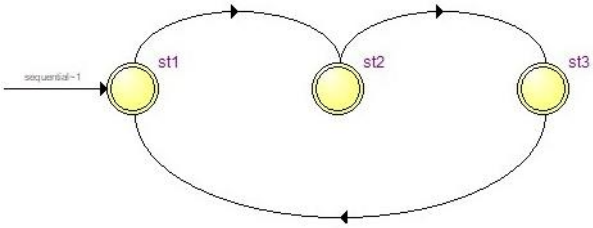

Fig. 44. Maquina de estados finitos de la multiplexación generada por Quartus.

C. Descripción de Estados

- Máquina de estados de las posiciones de las raquetas

  Las entradas de la máquina de estados son el joystick (arriba y abajo), el clk y el reset. Las salidas son las 11 posiciones diferentes que puede obtener la paleta (Figura 47). Los estados son las 11 posiciones diferentes.

  Para poder cambiar de estado se necesita cumplir dos condiciones, que haya un flanco de subida en el reloj y que alguno de los estados del joystick haya cambiado.

  El estado inicial es la posición 1, si el joystick sube el estado siguiente es posición 7, de lo contrario el estado seria posición 2 en la Figura 45 podemos observar donde están ubicadas las diferentes posiciones ya que la posición siguiente no solo dependerá de los joysticks sino también de la posición actual.

Fig. 45 Posiciones de loa raqueta.

- Máquina de estados del control de las raquetas

  En esta máquina de estados las entradas, salidas y estados son las 11 posiciones posibles que pueden tener la paleta, adicionalmente para la entrada tenemos el reset y el clk. (Figura 48)

  Esta máquina es usada para verificar en cual estado se encuentra la raqueta, y con esto podemos saber cuál de los diferentes códigos de visualización VGA ejecutar.

- Máquina de estados del control de las direcciones de la bola

  Las entradas de la máquina de estados son la posición actual de la bola en los dos diferentes ejes (x, y), el clk y el reset. Las salidas son las 4 diferentes direcciones que puede tomar la pelota (Figura 49). Los estados son las 4 direcciones diferentes.

  Para poder cambiar de estado se necesita cumplir dos condiciones, que haya un flanco de subida en el reloj y que la pelota haya llegado a alguno de sus límites, los cuales son las paletas, y el borde superior e inferior de la pantalla (blank).

  El estado siguiente depende de la dirección en la cual se encuentra y en el limite con el cual haya chocado (Figura 46), por esto en el diagrama de estados podemos visualizar que de un estado se puede salir a dos estados diferentes y dos estados diferentes llegan a este. Mientras este en un estado cada ciclo de reloj, se le sumara un pixel o se le restara en los dos ejes dependiendo en la diereccion en la cual se encuentre.

  En la dirección 1 se restan pixeles en el eje y, y se suman en el eje x. En la dirección 2 se suman pixeles en el eje y, y se suman en el eje x. En la dirección 3 se restan pixeles en el eje y, y se restan en el eje x. En la dirección 4 se suman pixeles en el eje y, y se restan en el eje x.

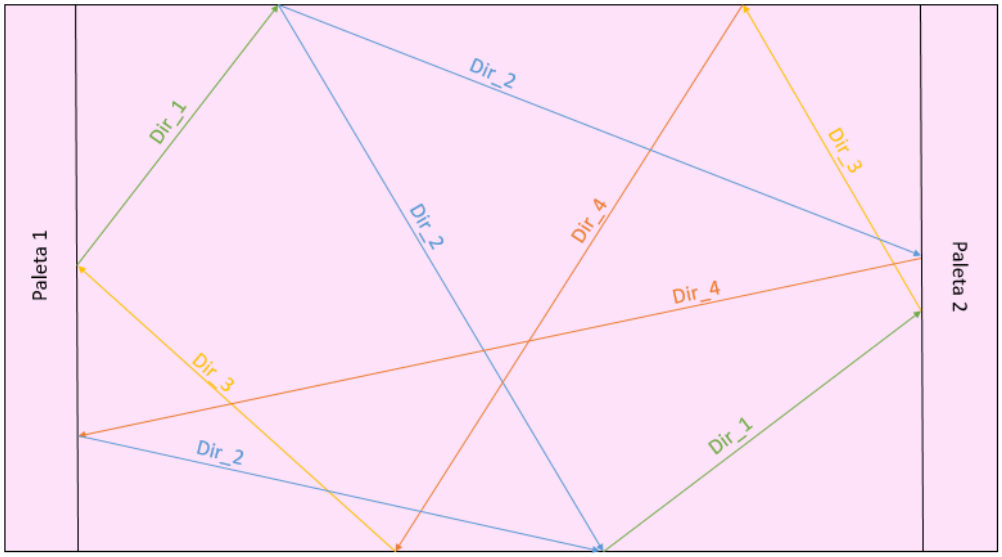

Fig. 46 Direcciones de la bola.

- Máquina de estados de los límites de la bola

  Las entradas de la máquina de estados son la posición de la bola en el eje y, la posición de las dos paletas, y dos comparadores del eje x, el clk y el reset. Las salidas son dos señales de perder (una para cada jugador) y un rst (Figura 50). Los estados son uno de transición(rinicial), otro que espera hasta que uno de los dos jugadores haya perdido(r1), y los otros dos estados son de que perdió alguno de los dos jugadores (per1, per2).

  El comparador en el eje x nos dice si la pelota se encuentra en alguno de los límites de las raquetas si alguno de estos está activo se verifica si la posición de la raqueta, es igual a la posición de la pelota en el eje y, si esto es así se mantiene en el estado r1, de lo contrario ira a alguno de los estados de perder, dependiendo de cuál de los dos comparadores se haya activado se sabrá cuál de los dos jugadores perdió.

  Cuando alguno de los dos jugadores pierde se va a per1 o per2, dependiendo del jugador que haya perdido y sale una señal que nos indica que el jugador perdió y adicionalmente otra para resetear el sistema.

- Máquina de estados de la multiplexación

  Las entradas de la máquina de estados son el clk y el reset. Las salidas son los 3 estados diferentes (Figura 51). Los estados son los tres elementos, los cuales son las dos paletas y la bola.

  La única condición de transición de esta máquina de estados es el reloj debido a que queremos multiplexar entre estos tres estados para poder visualizar en la pantalla VGA, esto se hace con una frecuencia determinada, para que el ojo humano no lo alcance a detectar.

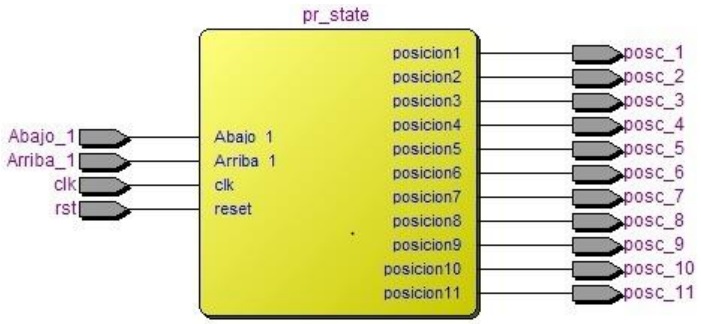

Fig. 47 Entradas/salidas de las posiciones de las raquetas.

Fig. 48 Entradas/salidas del control de las raquetas.

Fig. 49 Entradas/salidas del control de las direcciones de la bola.

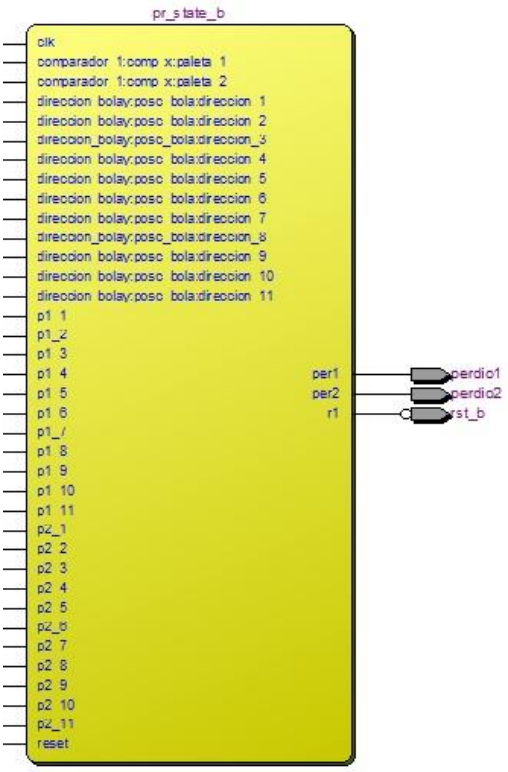

Fig. 50 Entradas/salidas de los límites de la bola.

Fig. 51 Entradas/salidas de la multiplexación.

IV. PRUEBAS DE FUNCIONAMIENTO

- En primera instancia hicimos pruebas para visualizar en la pantalla como se puede ver en la figura 52.

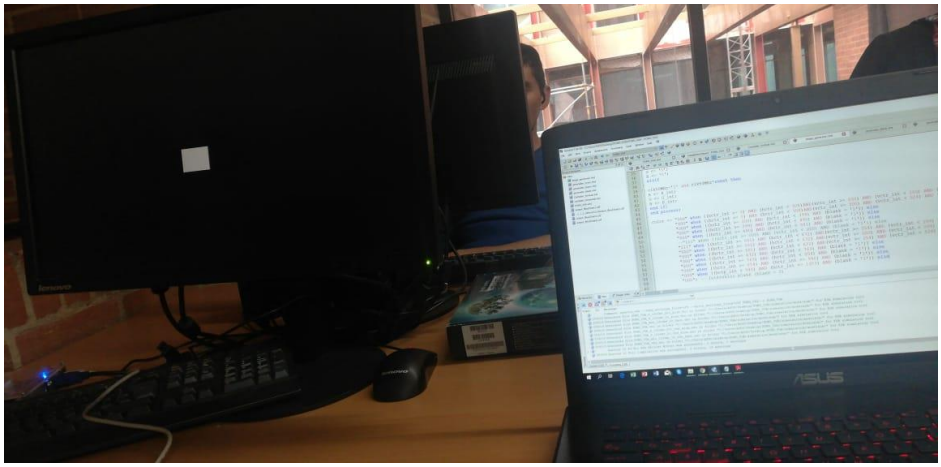

Fig. 32. Prueba de concepto de visualizacion de un punto.

- En paralelo a la visualización tuvimos un pequeño reto, el cual fue hacer que nuestros joysticks análogos me dieran una señal de entrada digital lo que nos llevo a realizar varias pruebas hasta escoger un adc como solución.

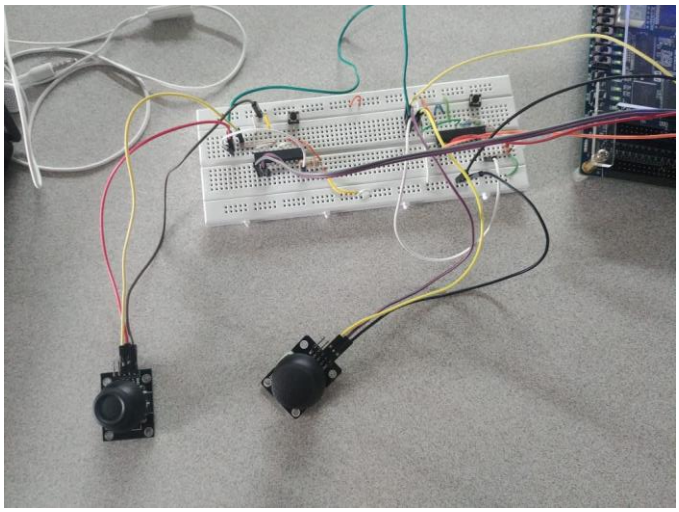

Fig. 43. Prueba de concepto de joysticks con adc.

- Después de ajustar los joysticks comenzamos a hacer pruebas de funcionamiento de las raquetas controlados con los joysticks.

Fig. 54. Prueba de concepto del movimiento de la paleta con los joysticks .

- Después de tener las raquetas comenzamos a hacer pruebas para el movimiento de la bola.

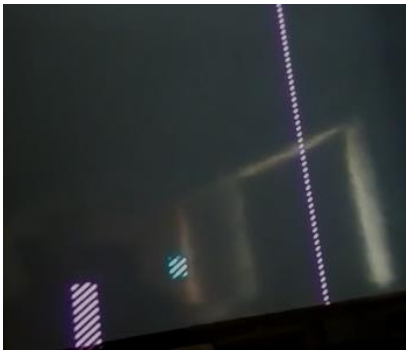

Fig. 65. Prueba de concepto del movimiento de la bola.

V. Conclusiones

Pese a que las pantallas VGA fueron demasiado útiles a lo largo del tiempo la tecnología ha avanzado y gracias a esto, género que tecnologías como HDMI sustituyeran los cables VGA

El resultado del juego pongo fue optimo se logro un juego fluido y una buena interacción al jugar.

Al desarrollar el proyecto notamos que gracias a como estructuramos nuestro diseño tuvimos un juego mas diverso con varios colores y con movimientos singulares en la pantalla. Lo que hace que sea un juega bastante llamativo al espectador.

Pese a las dificultades de diseño que tuvimos logramos superar y apropiar saberes que creemos útiles para proyectos futuros.

Referencias

[1] Sistemas.com. (2019). VGA. [online] Available at: https://sistemas.com/vga.php [Accessed 2 Apr. 2019].

[2] Digilogic.es. (2019). Máquinas de estado finito en VHDL – Digilogic. [online] Available at: https://www.digilogic.es/maquinas-de-estado-finito-fsm-vhdl/ [Accessed 2 Apr. 2019].

[3] Planeta Chatbot : todo sobre los Chatbots y la Inteligencia Artificial. (2019). Qué es una FPGA y por qué jugarán un papel clave en el futuro. [online] Available at: https://planetachatbot.com/qu%C3%A9-es-una-fpga-y-por-qu%C3%A9-jugar%C3%A1n-un-papel-clave-en-el-futuro-e76667dbce3e [Accessed 2 Apr. 2019].

[4] ingeniatic. (2019). Conversor analógico-digital. [online] Available at: https://www.etsist.upm.es/estaticos/ingeniatic/index.php/tecnologias/item/425-conversor-anal%C3%B3gico-digital.html [Accessed 4 Apr. 2019].