Proyecto de Curso

Sistemas Electrónicos (Profundización en el manejo de placas FPGA)

Javier Gil León

Pablo López Arcila

Alfredo Zarazaga Montalbán

Índice:

1.- Introducción.

Como ingenieros especializados en el campo de la electrónica, uno de nuestros rasgos más característicos debe ser el correcto y frecuente uso de la lógica, pues nuestros objetivos sólo podrán llevarse a cabo si conseguimos, con nuestra destreza, poder definir cómo y cuándo ocurran las acciones.

Para ello, haremos uso de todo lo que esté a nuestro alcance (lógica boolena, lenguajes de programación, hardwares y softwares adaptados a la situación, etc), y así poder ofrecer la mejor solución al problema que se nos presenta.

Para que comencemos a desarrollar nuestras capacidades, además de los ejercicios propuestos en la asignatura, se nos presenta a los estudiantes de esta asignatura un proyecto de curso, en el que tendremos que aplicar todo lo aprendido durante las clases teórico-prácticas.

El proyecto en cuestión se nos presenta como el desarrollo de un videojuego (una versión del aclamado Pac-Man), siendo esto una simple excusa para demostrar el correcto uso del lenguaje VHDL, denotando nuestra capacidad del manejo de variables, señales y relación entre las conocidas máquinas de estado. Todo esto, aplicado en una placa FPGA, en la que se procesarán las únicas entradas al sistema (para nuestro caso particular, los botones que moverán al individuo), y se sacarán por pantalla mediante la transmisión en cable VGA el efecto de nuestras decisiones en las entradas (veremos cómo se recorre el mapa, se aumenta la puntuación e incluso se podrá llegar a un estado de final o muerte).

Durante las siguientes líneas de la presente memoria, se explicará cómo se ha llevado a cabo este proyecto, detallando el funcionamiento de cada bloque que lo conforma.

2.- Desarrollo General del proyecto.

Como se ha informado antes, el proyecto se ha desarrollado en su totalidad en lenguaje VHDL, e implementado en una placa de desarrollo de *Digilent BASYS3*, basada en la FPGA de Xilinx *Artix-7 35T*.

La estructura se basa en una concatenación de máquinas de estado y bloques funcionales, relacionados y comunicándose entre sí mediante señales. Las únicas entradas al sistema son los botones de la misma placa, asociados a los pines U18, T18, W19, T17 Y U17, (Central, Arriba, Izquierda, Derecha y Abajo, respectivamente), mientras que las salidas del mismo son los distintos valores de una secuencia RGB y su posición, para pintar correctamente lo que se desea mostrar por pantalla, mediante la conexión VGA.

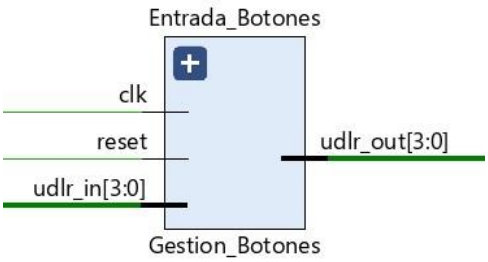
Las entradas mencionadas anteriormente se comportarán como los botones que decidirán el movimiento del comecocos, mientras que por pantalla se mostrará el mapa por el cual se moverá el susodicho, acompañado de los fantasmas.

Es necesario añadir que, durante el transcurso del proyecto, se ha decidido incluir una serie de mejoras, las cuáles no afectan a la funcionalidad inicial del mismo, pero nos permiten explotar más las opciones que nos ofrece el lenguaje VHDL.

Así pues, pasamos a estudiar los bloques uno por uno, con su respectiva funcionalidad singular.

3.- Descripción individual de bloques.

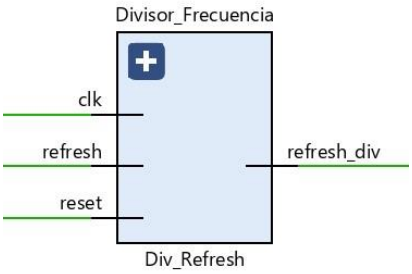
3.1.-Entrada Botones.



La función de este bloque es capturar el movimiento y evitar así problemas de rebote, además de asegurar un solo movimiento. Con esto conseguimos que el comecocos no reciba direcciones no deseadas, implementando un proceso one-hot.

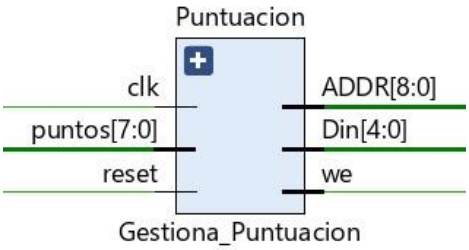
3.2.-Divisor de frecuencia.

Aquí se hace un divisor de frecuencia, gracias al cual conseguimos la señal refresh_div, que maneja el movimiento del comecocos. Se obtiene dividiendo entre 10 la señal refresh, proveniente del bloque VGA_driver (se explica más adelante), que indica que se ha pintado la pantalla completa.



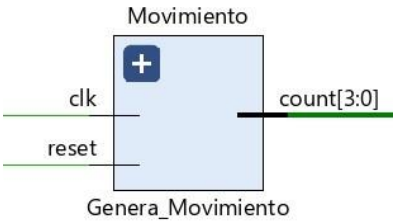
3.3.-Gestiona puntuación.

Bloque ocupado de recibir la puntuación directamente del comecocos. Dentro, esa misma puntuación se divide en sus dígitos, para enviarlos individualmente y poder mostrarlos por pantalla. Esto se considera una modificación, pues no afecta a la funcionalidad del proyecto.



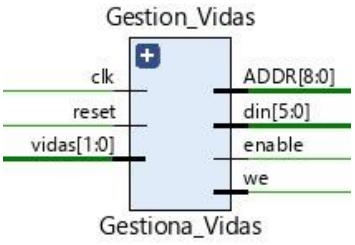
3.4.-Genera movimiento.

La funcionalidad de este código es proporcionar un algoritmo pseudoaleatorio, conseguido mediante lógica booleana y concatenación de vectores, para gobernar el movimiento de los fantasmas. Dos componentes de la salida irán al primer fantasma, y las otras dos al segundo.

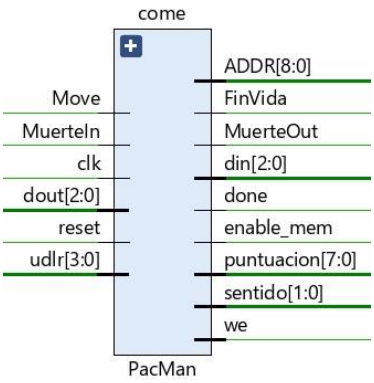


3.5.-Gestión vidas.

En este caso, el bloque recibe el número de vidas proveniente del comecocos y lo envía para escribirlo por pantalla. Vuelve a ser una mejora para la visualización del juego.



3.6.-PacMan.

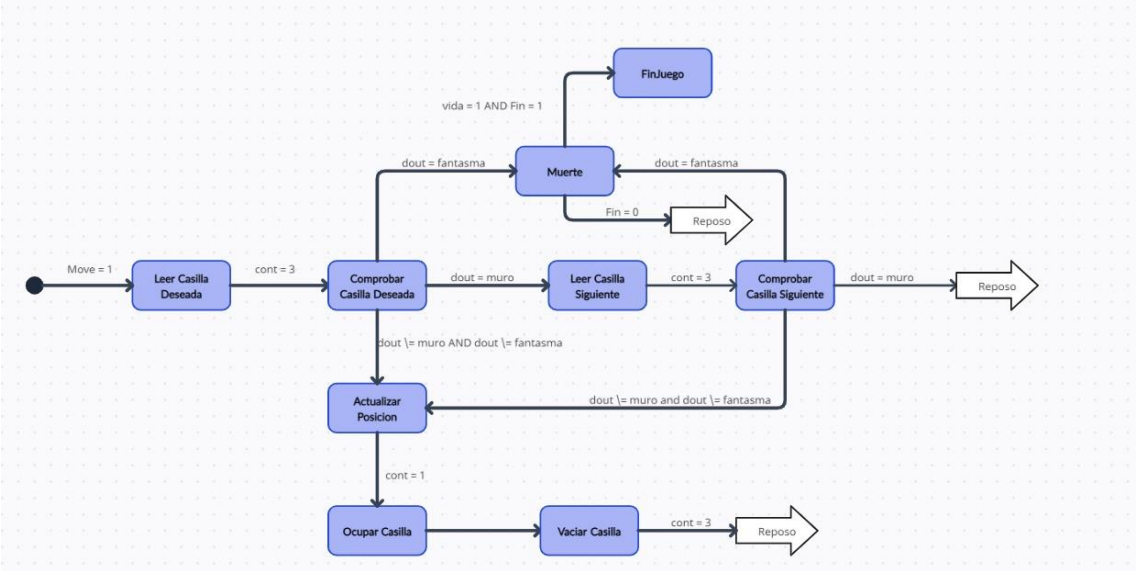


Llegamos al que se podría decir que es uno de los bloques principales del sistema. El comecocos se moverá cuando le llegue la señal proveniente del divisor de frecuencia, hacia la dirección insertada por el usuario y previamente filtrada por el bloque de gestión de botones. Esto lo hará solo y exclusivamente tras observar si la dirección de la casilla a la que apunta no está ocupada por un muro. En otros casos:

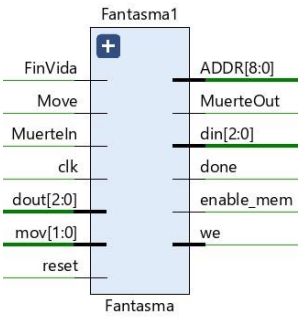
1. Bola: Se incrementa la puntuación. También es recalcable que hay distintos valores de incremento de puntuación (Monedas=1, Setas=3).
2. Vacío: simplemente se desplazará a esa casilla
3. Fantasma: se pierde una vida y se vuelve al punto inicial.

Cuando se encuentra con un fantasma, se envía la señal MuerteOut y se decrementa una vida. Cuando se llega al final de las vidas, el juego se congela.

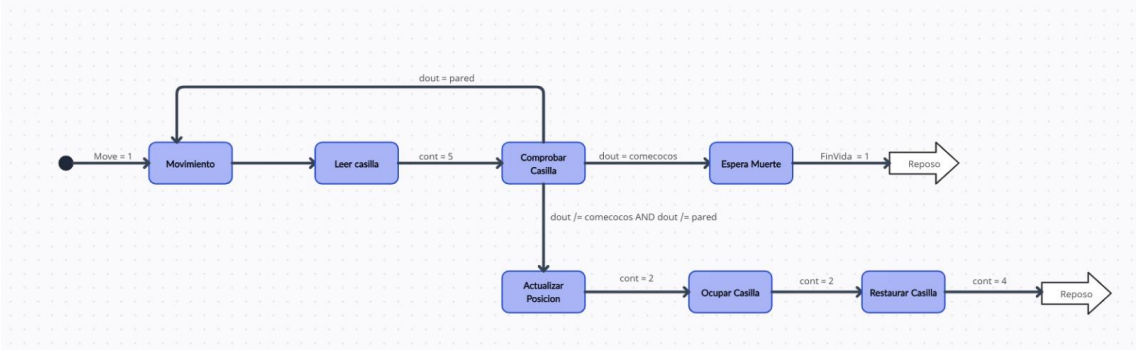
La salida Sentido forma parte del objetivo de la mejora estética del juego, para conseguir que el comecocos dirija su mirada hacia donde se mueve.



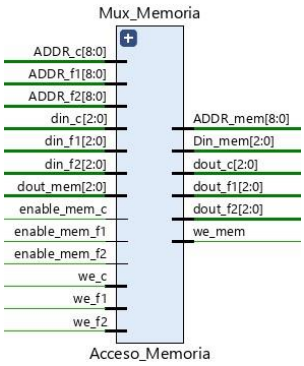
3.7.-Fantasma.



El fantasma se comporta de manera similar al comecocos. Se irá moviendo según lo que reciba desde el bloque Genera Movimiento, y a su paso irá dejando atrás todo lo que había. Es decir, si el fantasma ocupa una casilla donde antes hubiera una bola, al salir de la casilla la volvería a dibujar en su sitio. Cuando se encuentra con el comecocos, envía una señal de muerte, para indicar el encuentro.
También cabe recalcar que (del mismo modo que el comecocos se mueve con la señal del divisor de frecuencia) la máquina de estados del fantasma se ejecuta con la señal Done que sale del comecocos.

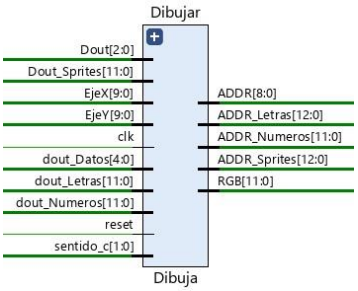


3.8.-Acceso Memoria.



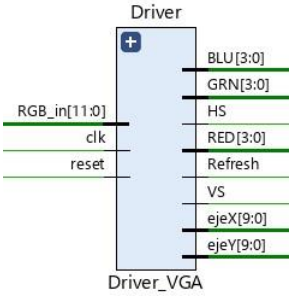
A nivel de conceptos, simplemente representa un multiplexor que hace que de todos los subsistemas que necesitan acceder a la memoria del tablero para leerla y modificarla, sólo esté accediendo una a la vez. Con esto, se consigue que, aunque tengamos más de un bus que conectar al mismo puerto de la memoria, se le proporcione prioridad a uno y se eviten cortocircuitos indeseados

3.9.-Dibuja.



En este programa se recibe desde el controlador de VGA (se explica más adelante) la posición de la pantalla, y se accede a las memorias para leer el dato que se encuentra en esa posición. Tras leerlo, se envía codificado en 12 bits el color en RGB, para mostrarlo por pantalla.

3.10.-VGA Driver.

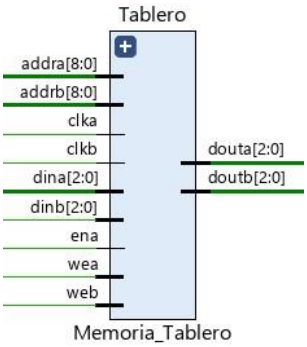


Como se ha mencionado antes, tras haber mandado la posición al bloque Dibuja, se recibe el color correspondiente, y se envía al monitor los tres colores del RGB de forma separada. Cuando se ha pintado toda la pantalla, se activa la señal de refresh, la cual es la entrada del bloque del divisor de frecuencia.

Estos dos últimos bloques se han realizado durante las sesiones prácticas de la asignatura.

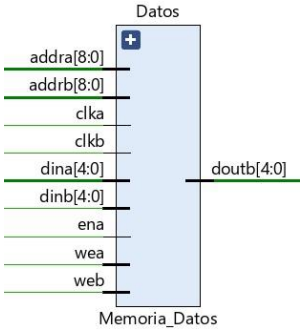
3.11.-Memoria Tablero.

En este bloque de memoria se carga en un inicio el mapa, cargado en un archivo .coe, y a través del Acceso Memoria, las máquinas del comecocos y el fantasma podrán leer y escribir en la memoria para moverse por el mapa. A pesar de tener dos puertos, nosotros sólo hacemos uso del puerto A, dejando el B completamente inutilizado.



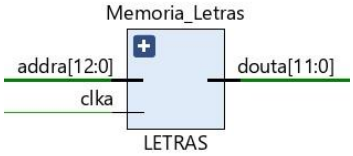
3.12.-Memoria Datos.

Memoria que recoge todo lo referente al espacio de pantalla directamente inferior al tablero. Forma parte de las mejoras, y se usa tanto como para mostrar la puntuación, como para llevar la cuenta del número de vidas y mostrar los nombres de los creadores del proyecto. Esta y las siguientes memorias conforman las últimas mejoras del proyecto.



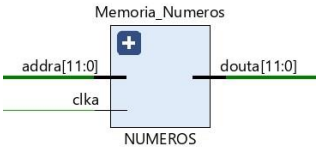
3.13.-Memoria Letras.

En esta memoria se carga desde un .coe todas las letras necesarias para la escritura en el proyecto, es decir, la palabra SCORE y los nombres.



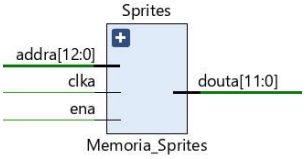
3.14.-Memoria Números.

El mismo propósito de antes, pero guarda los números que representan la puntuación y el número de vidas

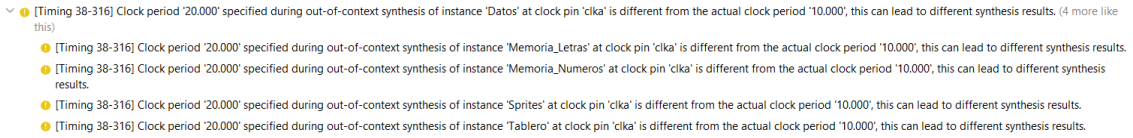


3.15.-Memoria Sprites.

Esta memoria guarda los sprites necesarios para obtener la temática de Super Mario Bros. Cambia el comecocos por Mario, los fantasmas por Goombas, las bolas por monedas y los muros los convierte en los del videojuego de la NES.



4.-Warnings en Síntesis.



Estos warnings se producen debido a que, en el fichero de restricciones, se le ha especificado un periodo de reloj de 10, pero durante la síntesis, se le asigna un periodo de reloj de 20

5.-Recursos utilizados de la placa.

Se presenta unos datos que nos informan de los recursos que hemos necesitado usar de la placa para la realización del proyecto.

-Completos

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs*	358	0	0	20800	1.72
LUT as Logic	358	0	0	20800	1.72
LUT as Memory	0	0	0	9600	0.00
Slice Registers	214	0	0	41600	0.51
Register as Flip Flop	214	0	0	41600	0.51
Register as Latch	0	0	0	41600	0.00
F7 Muxes	0	0	0	16300	0.00
F8 Muxes	0	0	0	8150	0.00

-Memoria Tablero

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs*	0	0	0	20800	0.00
LUT as Logic	0	0	0	20800	0.00
LUT as Memory	0	0	0	9600	0.00
Slice Registers	0	0	0	41600	0.00
Register as Flip Flop	0	0	0	41600	0.00
Register as Latch	0	0	0	41600	0.00
F7 Muxes	0	0	0	16300	0.00
F8 Muxes	0	0	0	8150	0.00

-Memoria Datos

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs*	0	0	0	20800	0.00
LUT as Logic	0	0	0	20800	0.00
LUT as Memory	0	0	0	9600	0.00
Slice Registers	0	0	0	41600	0.00
Register as Flip Flop	0	0	0	41600	0.00
Register as Latch	0	0	0	41600	0.00
F7 Muxes	0	0	0	16300	0.00
F8 Muxes	0	0	0	8150	0.00

-Memoria Números

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs*	0	0	0	20800	0.00
LUT as Logic	0	0	0	20800	0.00
LUT as Memory	0	0	0	9600	0.00
Slice Registers	0	0	0	41600	0.00
Register as Flip Flop	0	0	0	41600	0.00
Register as Latch	0	0	0	41600	0.00
F7 Muxes	0	0	0	16300	0.00
F8 Muxes	0	0	0	8150	0.00

-Memoria Letras

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs*	5	0	0	20800	0.02
LUT as Logic	5	0	0	20800	0.02
LUT as Memory	0	0	0	9600	0.00
Slice Registers	2	0	0	41600	<0.01
Register as Flip Flop	2	0	0	41600	<0.01
Register as Latch	0	0	0	41600	0.00
F7 Muxes	0	0	0	16300	0.00
F8 Muxes	0	0	0	8150	0.00

-Memoria Sprites

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs*	7	0	0	20800	0.03
LUT as Logic	7	0	0	20800	0.03
LUT as Memory	0	0	0	9600	0.00
Slice Registers	2	0	0	41600	<0.01
Register as Flip Flop	2	0	0	41600	<0.01
Register as Latch	0	0	0	41600	0.00
F7 Muxes	0	0	0	16300	0.00
F8 Muxes	0	0	0	8150	0.00