Universidad Carlos III de Madrid

Manual de Prácticas

Diseño de Circuitos Integrados

Grado en Ingeniería Electrónica Industrial y Automática

Curso 4º, Cuatrimestre 1º
2017-2018

Autores: Luis Entrena Arrontes
Enrique San Millán
Mario García Valderas
Fernando Casado Ortiz
Luis Mengibar Pozo
Almudena Lindoso

ESCUELA POLITÉCNICA SUPERIOR
DEPARTAMENTO DE
TECNOLOGÍA ELECTRÓNICA

ÍNDICE

1. NORMAS GENERALES

NORMAS GENERALES DEL LABORATORIO DE ELECTRÓNICA

Debido a la cantidad de alumnos que pasan por el Laboratorio de Electrónica a lo largo de un curso es conveniente seguir algunas normas generales que pasamos a enumerar.

Está totalmente prohibido:

- **Fumar en todo el recinto.**
- **Introducir cualquier tipo de comida o bebida en el Laboratorio**. El caso de la bebida es una cuestión, además de higiene, de seguridad. Un líquido derramado encima de cualquiera de los equipos puede dañarlos irreversiblemente, incluso, se pueden declarar incendios.
- **Realizar otras actividades** ajenas al trabajo específico que ha de desarrollar en el laboratorio.
- **Pasar a las zonas reservadas** al personal del laboratorio.
- **Dañar deliberadamente alguno de los instrumentos** de los que consta el equipamiento del laboratorio.
- **Sacar,** aunque solo sea de manera provisional, **cualquier tipo de instrumentación o material propio del laboratorio fuera del mismo**. Esta falta se considerará muy grave, poniéndose rápidamente en conocimiento de la Dirección de la Escuela para que tome las medidas oportunas.

Asimismo se recomienda observar las siguientes directrices siempre que se esté en el aula:

- Procure utilizar un tono de voz adecuado. Solo le tiene que escuchar su compañero y, en su caso, el profesor. El laboratorio es un aula de gran capacidad e, incluso, puede haber gente de varias titulaciones a la vez. Procure no molestar al resto de sus compañeros.
- Los profesores de prácticas y becarios están para ayudarle, no dude en llamarlos para que le resuelvan cualquier tipo de duda. No obstante, procure resolver los problemas usted mismo y no los llame a la mínima dificultad, ese es parte del aprendizaje.
- Las horas de laboratorio están concebidas para realizar, comprobar los montajes y llevar a cabo las medidas oportunas. Todos los cálculos teóricos y estudios han de llevarse a cabo con anterioridad para entrar al laboratorio sabiendo lo que hay que hacer.

Para la realización de las prácticas puede ser necesario un instrumental, además del propio de cada puesto, que le será facilitado por el maestro de laboratorio previa entrega de un carné identificativo del alumno (carné de la escuela, DNI., carné de conducir,...). Este material se presta a cada grupo solamente durante la duración de la sesión de prácticas, teniendo que ser devuelto a su finalización. Tenga en cuenta que el material que se le da es apuntado junto con el puesto que ocupa y el turno. Si faltara algún material sería rápidamente localizado el infractor tomándose las medidas oportunas. No preste su equipo a otro grupo de prácticas. El responsable de ese material es usted.

El objetivo que se persigue con las prácticas de laboratorio en el Departamento de Tecnología Electrónica es que usted adquiera una componente práctica en su formación como Ingeniero. La correcta observación de estas normas y recomendaciones hará que su estancia en el laboratorio sea más provechosa y se cumpla mejor dicho objetivo.

NORMAS DE SEGURIDAD

En las mediciones de valores de tensiones, intensidades y magnitudes de pequeño valor, no es necesario prestar atención a ningún tipo particular de precaución personal. Pero en los casos de medida de altos valores de tensiones, tanto en la industria, como en electrónica, reparación de televisión, etc., el manejo incorrecto de los equipos, puede ser causa de daños personales. Por tanto, en la medida de tensiones elevadas es una buena precaución hacer uso sólo de una mano, apoyando la otra en la espalda.

En las prácticas que se efectúen en el laboratorio, debe informarse siempre al responsable de cualquier daño personal, de los equipos, de circuitos etc., pues una avería sin reparar puede ser causa de mayores daños.

Siguiendo las recomendaciones de seguridad vigentes, tenga en cuenta las siguientes reglas de seguridad para evitar daños al usuario o a terceros:

1. No use el polímetro si las puntas de prueba están rotas o defectuosas.
2. El conmutador, conmutadores, selectores o elementos de selección (tales como clavijas, jacks, etc.) deben estar en la posición correcta para efectuar la medición. El instrumento ha de estar preparado para realizar el tipo de medida que deseemos, tales como tensión, corriente, resistencia etc.
3. El conmutador o selector de escalas, debe de estar siempre en la posición más alta siempre que se mida una tensión o corriente desconocida.
4. Cuando haga mediciones eléctricas, **NUNCA** se ponga usted a tierra. Trabaje siempre aislado, sobre una alfombra de goma, con zapatos de suela de goma, etc. **NUNCA** toque tuberías u otras partes metálicas con el cuerpo mientras esté efectuando medidas.
5. No sobrepasar las tensiones máximas especificadas por el equipo en ninguna de las medidas, podría dañarlo y además podría exponerse a una descarga.
6. No sobrepasar la tensión máxima especificada entre el borne COM y la toma de Tierra (sí tuviera).
7. Extreme las precauciones cuando mida tensiones que superen los 60V DC ó los 30V AC rms.
8. Si tiene que manipular un circuito (soldar, cortar, etc.) asegúrese, antes de hacerlo, de que la alimentación general del circuito esté desconectada.

2. ORGANIZACIÓN Y EVALUACIÓN DE LAS PRÁCTICAS

- Las prácticas de la asignatura Diseño de Circuitos Integrados son obligatorias. **La asistencia a las prácticas es obligatoria**. El trabajo se realizará por **grupos de dos personas**, a los que se asignará un puesto en el laboratorio que deberán mantener hasta el final de las prácticas. Los puestos constan del material necesario, tanto informático como de instrumentación. Al comienzo de cada sesión se suministrará a los alumnos material adicional que deberán devolver al final de cada sesión.
- Las prácticas forman parte de un Trabajo de Diseño que tiene un peso sobre la nota total de la asignatura del 35%. No obstante, tenga siempre presente que **las Prácticas de Laboratorio son sólo una parte de dicho Trabajo de Diseño**, concretamente la que permite comprobar el funcionamiento real del diseño con los medios del laboratorio. Adicionalmente, las sesiones en Aula Informática previas a la realización de las prácticas se dedicarán también a la realización del Trabajo de Diseño, como se indica en el cronograma.
- El Trabajo de Diseño es responsabilidad suya y deberá completarlo con su trabajo personal fuera del Aula o del Laboratorio. **Una semana después de la última sesión de prácticas deberá entregar una memoria donde se detalle el diseño realizado**. La memoria deberá incluir:
  - Descripciones VHDL de los diseños realizados y de los bancos de pruebas utilizados.
  - Exposición breve de cómo se han realizado los diseños y de los aspectos que se consideren importantes.
  - Resultados finales de síntesis (área y frecuencia máxima de reloj)
  - No es necesario entregar simulaciones de los diseños.
- En la evaluación del Trabajo de Diseño se tendrán en cuenta:
  - La consecución de los objetivos parciales al finalizar cada sesión de prácticas
  - El diseño final realizado
  - Mejoras con distinto nivel de complejidad realizadas
- Se valorará positivamente a la hora de calificar el Trabajo de Diseño:
  - Conseguir a tiempo los objetivos parciales de cada práctica.
  - Obtener un diseño óptimo en área y frecuencia de funcionamiento.
  - Realizar ampliaciones al diseño básico.
- Serán causa de suspenso del Trabajo de Diseño:
  - **Falta de asistencia a cualquiera de las sesiones.** Cualquier ausencia deberá ser justificada **de modo documental** ante los profesores de prácticas.
  - No entregar la memoria.
  - No conseguir el funcionamiento del diseño completo en su versión básica.
- En el presente manual de prácticas se describen los diseños que los alumnos deben realizar durante las sesiones. Adicionalmente, se sugieren algunas mejoras al diseño básico que los alumnos pueden implementar de manera opcional y que serán tenidas en cuenta en la nota final. Se anima a los alumnos a sugerir e implementar sus propias mejoras al sistema.
- Como forma de trabajo, se recomienda que realice y simule con anterioridad las partes del diseño que se vayan requiriendo, aproveche la sesión previa en Aula Informática para solventar las dudas o los problemas que hayan podido surgir, y demuestre en la sesión de Laboratorio el funcionamiento de su diseño sobre la FPGA. Organícese con anticipación para asegurarse de cumplir los objetivos marcados en este manual al final de cada sesión de Prácticas. Tenga presente que lo que no haga antes lo tendrá que hacer después, con riesgo de incumplir los objetivos marcados.

Todos los alumnos implicados en cualquier intento de aprobar el Trabajo de Diseño mediante copia u otra forma similar serán automáticamente suspendidos. Esta norma se aplicará tanto a aquellos alumnos que intenten aprobar el Trabajo de Diseño utilizando medios ajenos al propio

esfuerzo personal como a aquellos alumnos que faciliten dichos medios o ayuden a los anteriores.

3. INTRODUCCIÓN

El objetivo de las prácticas es que los alumnos realicen, de un modo guiado, el diseño de un circuito digital utilizando el lenguaje de descripción de hardware VHDL. Este diseño se implementará y probará en un circuito programable (FPGA) del fabricante Xilinx, concretamente un dispositivo de tipo **Artix-7, xc7A35T-1 CPG236**.

En primer lugar se ofrece una descripción del sistema completo, con su funcionalidad y especificaciones, y el *hardware* en el que debe ser implementado. El sistema propuesto es una versión simplificada del conocido videojuego "Space Invaders" (Invasores del espacio).

Posteriormente se propone una división en bloques del sistema, para facilitar la labor de los alumnos.

3.1. DESCRIPCIÓN DEL SISTEMA: INVASORES DEL ESPACIO

El sistema que se quiere implementar es una versión simplificada del conocido juego recreativo Invasores del Espacio (Space Invaders). En dicho juego, una nave situada en la parte baja de la pantalla debe disparar y destruir a los invasores del espacio, que están situados en la parte alta de la pantalla.

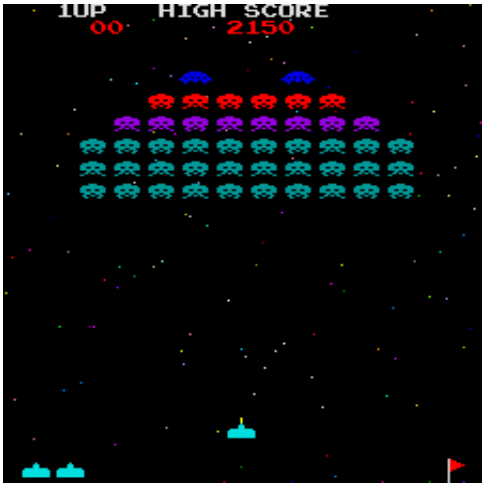

Figura 3.1. Una de las múltiples versiones del Space Invaders

La nave, controlada por el jugador, puede moverse libremente en sentido horizontal, pero sin salir de los límites de la pantalla. El movimiento en sentido vertical de la nave no está permitido. La nave puede realizar disparos con el fin de destruir a los invasores. Mientras un disparo esté a mitad de su recorrido, la nave no podrá realizar un nuevo disparo; es decir, nunca podrá haber dos disparos de la nave en la pantalla.

Los invasores se sitúan inicialmente en la parte superior de la pantalla. A medida que pasa el tiempo, los invasores se van moviendo lateralmente de izquierda a derecha, y cada vez que el grupo alcanza un extremo de la pantalla, se produce un ligero descenso del grupo.

Mientras el grupo de invasores sigue regularmente su pauta de movimiento, el jugador debe controlar la nave para destruirlos. Cuando un disparo de la nave impacta en un invasor, éste es destruido y desaparecen de la pantalla tanto la nave como el disparo.

La nave puede ser destruida si es alcanzada por los disparos de los invasores, o si choca directamente con algún invasor que haya descendido lo suficiente hasta llegar a la altura de la nave.

Cuando todos los invasores han sido destruidos, un nuevo grupo de ellos vuelve a aparecer en la parte superior de la pantalla y el juego continua.

3.2. ESPECIFICACIONES

HARDWARE A UTILIZAR

El juego se implementará en su totalidad en la placa de prototipos BASYS 3 que se suministra durante las sesiones de prácticas.

Dicha placa dispone, entre otros, de los siguientes elementos:

- FPGA: Xilinx XC7A35T-1 CPG236. Se trata de un circuito programable basado en SRAM, es decir, es volátil y, al desconectar la alimentación del circuito, perderá la programación que se hubiera grabado en él.
- Pulsadores (5). Son activos por nivel alto, es decir, tienen valor lógico '0' cuando están sin pulsar y '1' cuando están pulsados.
- Interruptores. También son activos por nivel alto. Cuando están abajo (posición más cerca del borde de la placa) toman el valor lógico '0' y cuando se colocan arriba toman el valor '1'.
- LEDs. Son activos por nivel alto, es decir, se encienden cuando se les asigna un valor lógico '1'.

Además la placa tiene las siguientes interfaces:

- Interfaz VGA, con conector DB15, que será el que se usará para la conexión de un monitor.
- Display de 4 dígitos de 7-segmentos.
- Puerto micro-USB que tiene varias funciones: alimentación de la placa (5V), programación del dispositivo y uso como puerto serie de tipo RS-232 (UART).

La FPGA se utilizará para implementar el hardware que se va a diseñar. Los pulsadores se usarán como controles del videojuego. Los LEDs podrán utilizarse como elementos de visualización adicionales, pero no serán necesarios en el diseño básico.

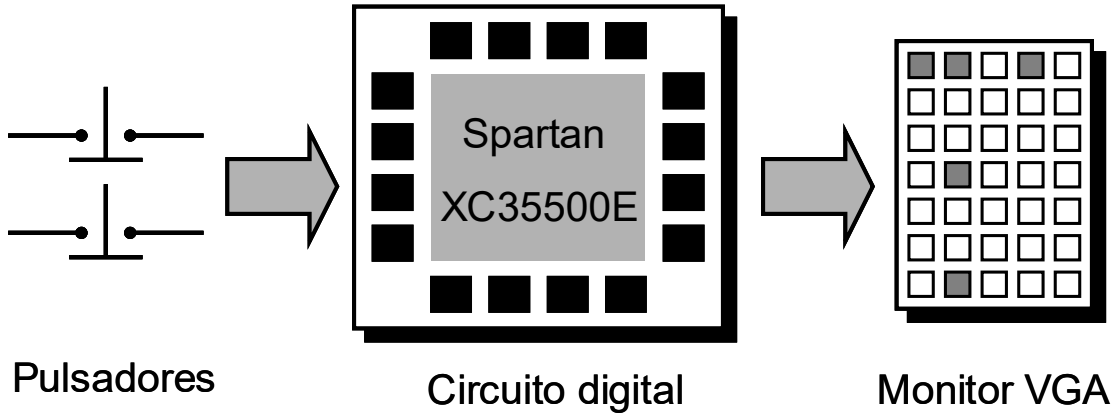

Figura 3.2. Estructura hardware del sistema básico.

En los apéndices se da información detallada de todos los elementos que se van a utilizar.

VERSION SIMPLIFICADA DEL VIDEOJUEGO

Se va a simplificar en gran medida las especificaciones del videojuego, dado que el realizar un juego que refleje todos los detalles requeriría un gran esfuerzo de desarrollo.

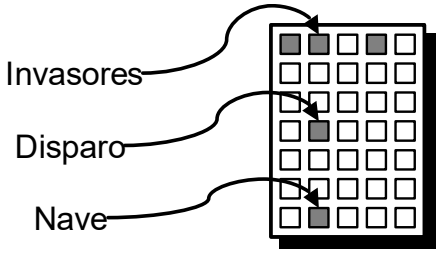

Figura 3.3. Pantalla del videojuego

Las características del videojuego que se va a diseñar realmente son las siguientes:

- Los invasores, la nave y los disparos serán representados en el monitor utilizando un cuadrado para cada uno.
- Inicialmente habrá 10 invasores, colocados en la línea superior de la zona de juego del monitor y en uno de los lados.
- Los invasores se desplazarán lateralmente hasta que uno de ellos llegue al límite del monitor. Cuando el invasor lateral llega al límite, todos los invasores se moverán en bloque una fila hacia abajo.
- Cuando el jugador pulse el botón de disparo, una bala saldrá desde la posición de la nave y se desplazará verticalmente hacia arriba. Si la bala coincide en su ascenso con algún invasor, éste desaparecerá de la pantalla. La nave no podrá disparar una nueva bala hasta que la anterior no haya terminado su recorrido.
- Las coordenadas de la zona de juego empiezan en la esquina superior izquierda y crecen hacia abajo y hacia la derecha, como indica la Figura 2.4. La zona de juego se compone de 20 columnas y 15 filas. Cada punto de la zona de juego está compuesto de 32x32 puntos de pantalla.

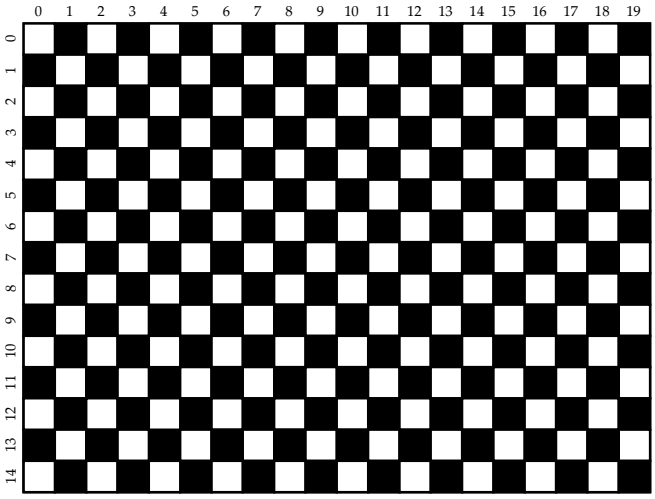

Figura 3.4. Coordenadas de la zona de juego.

Se ha descrito la funcionalidad básica del circuito que se va a diseñar. Es casi seguro que habrá comportamientos concretos que habrán quedado por definir. El alumno deberá completar estos aspectos del diseño como crea conveniente, describiendo apropiadamente sus decisiones en la memoria.

ENTRADAS Y SALIDAS

A continuación se describen las entradas y salidas del circuito digital. Los alumnos deben tener en cuenta que estas entradas y salidas serán siempre las mismas para cualquier versión del diseño que se vaya a grabar en la FPGA. **Es necesario que los alumnos respeten el nombre**

de estas señales, ya que facilita la localización de errores por parte de los profesores en el caso de que surjan problemas. Si es necesario, se podrán añadir señales para realizar ampliaciones al diseño básico.

ENTRADAS

Las entradas del sistema son el reloj, el reset, los controles de la nave y una señal adicional para empezar una partida.

- **Clk**: *std_logic*. Es la señal de reloj que gobernará el circuito. Se utilizará el oscilador disponible en la placa, que funciona a una frecuencia de 100 MHz.
- **Reset**: *std_logic*. Es la señal de reset del sistema. Activa a nivel alto.
- **Inicio**: *std_logic*. Cuando se activa esta señal, dará comienzo una nueva partida.
- **Izquierda**: *std_logic*. Control para mover la nave hacia la izquierda.
- **Derecha**: *std_logic*. Control para mover la nave hacia la derecha.
- **Disparo**: *std_logic*. Controla que la nave efectúe un disparo.
- **Test**: *std_logic*. Se utiliza para comprobar si la pantalla VGA funciona correctamente.

SALIDAS

Las salidas del sistema básico son exclusivamente las señales que controlan el monitor VGA que usaremos para visualizar el juego. Estas señales son:

- **R, G, B**: *unsigned(3 downto 0)* cada una. Valores de color. Cuatro bits por cada color permiten controlar su intensidad y generar hasta 4096 colores diferentes mediante combinaciones de R, G y B.
- **Hsync, Vsync:** *std_logic*. Indicadores de sincronismo horizontal y vertical, respectivamente.

La representación del juego en el monitor va a ser de la siguiente manera. El controlador VGA está preparado para que el monitor funcione a una resolución de 640x480 píxeles, con una frecuencia de refresco de 60 Hz, que significa que se dibujan 60 pantallas completas por segundo. Los puntos que representan tanto la nave, como el disparo, como los invasores van a ser cuadrados de 32x32 píxeles de pantalla. De esta forma, podemos considerar que la zona de juego tiene un tamaño de 20 columnas x 15 filas (como se muestra en la Figura 3.4), donde cada coordenada (fila, columna) se refiere a un cuadrado de 32x32 píxeles.

3.3. FUNCIONAMIENTO BÁSICO DEL CONTROLADOR VGA

Se va a explicar ahora el funcionamiento básico del controlador VGA. En el Anexo I se puede encontrar una explicación más detallada del Funcionamiento de un monitor. **Se aconseja leer dicho Anexo I antes de continuar.**

La entidad del circuito controlador VGA que se proporciona a los alumnos es la que se muestra en el Listado 3.1.

En el siguiente apartado se va a explicar la funcionalidad de las entradas y salidas: coordenadas de cada pixel (X,Y), señales de sincronismo vertical y horizontal, color de entrada y de salida, etc.

```
entity vga is
  port (Clk          : in std_logic;
        Reset        : in std_logic;
        Color        : in std_logic_vector(2 downto 0);
        Hsync        : out std_logic;
        Vsync        : out std_logic;
        R,G,B        : out unsigned(3 downto 0);
        X            : out unsigned(9 downto 0);
        Y            : out unsigned(9 downto 0));
end vga;
```

Listado 3.1. Entidad del controlador VGA.

FUNCIONALIDAD

Un controlador VGA es un dispositivo que indica al monitor qué color debe dibujar en cada uno de sus puntos. La entrada color se corresponde con el color que se desea mostrar para el pixel con coordenadas (X, Y). El controlador VGA, genera los valores R, G y B correspondientes y las señales de sincronismo con el monitor Hsync y Vsync, llamadas señal de sincronismo horizontal, y señal de sincronismo vertical respectivamente. Para que todo funcione de manera sincronizada, el controlador VGA debe funcionar a una frecuencia determinada, véase el Anexo I.

Las señales de sincronismo horizontal y vertical sirven para indicar al monitor tres cosas:

- cuándo se han terminado de enviar los colores de una fila (Hsync)
- cuándo se han terminado de enviar los colores de toda una pantalla (Vsync)
- y en función del periodo de estas señales, cuál es la resolución a la que queremos dibujar.

El funcionamiento básico del controlador es:

- Se recorren todos los píxeles de una fila del monitor, asignando los valores que correspondan a las salidas R, G y B en función del valor de la entrada *color* para cada pixel. En este caso color solo permitirá seleccionar 8 colores diferentes, puesto que por simplicidad es una entrada de 3 bits, 1 para cada componente de color R, G y B.
- Cuando se ha recorrido una fila completa de la pantalla, el controlador genera un pulso de sincronismo horizontal (Hsync), según se define en el Anexo I.
- Se recorren todas las filas de la pantalla, tantas veces como líneas tenga la resolución del monitor en la que se quiera dibujar. Por cada señal de sincronismo horizontal, el monitor debe haber dibujado una línea de la pantalla.
- Cuando se ha completado este ciclo para todas las filas del monitor, se envía una señal de sincronismo vertical (Vsync), que indica al monitor que se ha terminado de dibujar una pantalla completa, y se vuelve a reiniciar el proceso.

La frecuencia con que se envían los datos depende de la resolución a la que se quiera trabajar. Para la resolución de 640x480 píxeles, los datos se ponen cada 0,0393 us, es decir, que cambian con una frecuencia de 25 MHz. El monitor se configura a la resolución adecuada en función de la frecuencia de las señales de sincronismo horizontal y vertical.

Para poder saber cuál es el píxel concreto que se está representando en un momento determinado, el controlador tiene 2 contadores. El primero va de 0 a 639, y sirve para conocer la coordenada *X* (qué píxel de la fila se está dibujando). Este contador cambia también con una frecuencia de 25 MHz. El segundo contador, se corresponde con la coordenada *Y*, y cuenta el número de fila en el que se está dibujando. Su valor cambia entre 0 y 479, y se incrementa cada vez que el primer contador completa un ciclo, es decir, llega a 639 y pasa a 0. Estos contadores representan las coordenadas XY del punto que se está dibujando. Se ponen como salidas del controlador para que la lógica a la que está conectado, sepa la posición del punto del que se pide el color. La frecuencia de funcionamiento de estos contadores es la misma con la que el monitor lee los datos del color a dibujar.

El diagrama de bloques de este controlador incluye dos instancias de una entidad contador. En la Figura 2.5. tenemos representada la jerarquía del controlador.

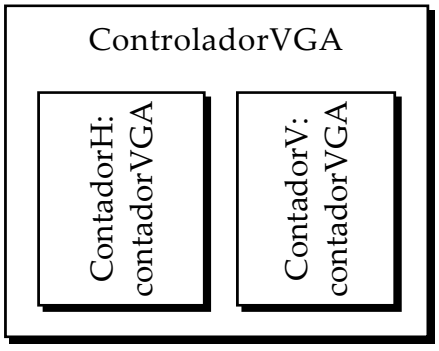

Figura 3.5. Jerarquía de bloques del controlador VGA.

En resumen, el controlador VGA genera las coordenadas del pixel que se va a mostrar en la pantalla (X, Y) y en función del valor que haya en la entrada *color* se deberá asignar un valor a las salidas *R*, *G* y *B*. Parte de la lógica que se deberá diseñar a lo largo de las prácticas es la que decide el color que hay que dibujar en función de esas coordenadas XY.

3.4. DIAGRAMA DE BLOQUES DEL SISTEMA COMPLETO

Para facilitar la labor del alumno a la hora de realizar los diseños, se sugiere una división del sistema en bloques. El diseño se hará por partes pero de modo que siempre se puedan sintetizar las partes diseñadas para probarlas en la placa. Se comenzará con la parte del control del monitor, ya que su funcionamiento es crucial para poder probar después el resto de los componentes. Se avanzará en el diseño mediante la inclusión de nuevos bloques.

Distinguiremos tres tipos de bloques:

- Bloques de gestión de entradas. Sincronizan las señales de entrada y evitan que se produzcan rebotes en los pulsadores.
- Bloques de control. Controlan el funcionamiento global del sistema
- Bloques de control de objetos. Controlarán los objetos que posteriormente se visualizarán, como por ejemplo, la nave, los disparos o los invasores.
- Bloques de salida. Controlan los dispositivos de salida. En principio, sólo hay que controlar el monitor, pero puede haber ampliaciones.

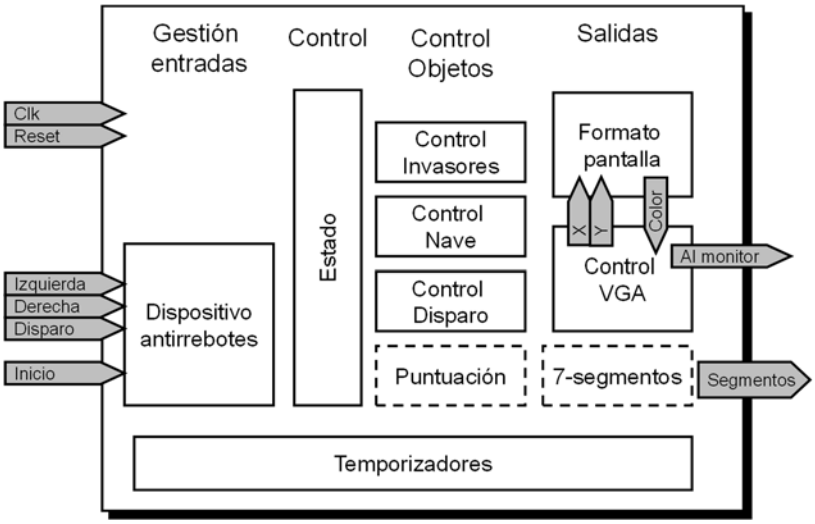

Figura 3.6. Estructura de bloques del diseño.

La filosofía del diseño es simple. Durante la partida, los diversos controles tendrán la posición de los invasores, de la nave, del disparo, y de cualquiera de los elementos adicionales que se quieran incluir. Cuando el control VGA solicite el color de una posición XY al bloque de formato de pantalla (formatoVGA), este traducirá las coordenadas de partida a coordenadas del juego y comprobará si alguno de los elementos del juego (invasores, nave, disparo, etc.), se encuentran en esa posición. Recordamos que la zona de juego ocupa 20 filas y 15 columnas.

Cada elemento del juego guarda sus coordenadas en unas señales propias. Así, por ejemplo, tendremos dos señales que nos dirán las coordenadas del disparo, una señal que nos dirá la coordenada de la nave (la nave siempre está en la última fila), y otras dos señales que nos dirán la posición de los invasores. Una posibilidad para los invasores, que simplifica el diseño, es que

la señal que nos da la información de posición horizontal tenga información de toda la fila. De esta forma tenemos en una sola pareja de señales, la información de los 10 invasores. Vamos a poner un ejemplo concreto para aclarar el funcionamiento. Supongamos que nuestras variables tienen el siguiente valor en un momento concreto de la partida:

Invasores: posicionH=0000011101111110000, posicionV=0

Nave: posicionH=1

Disparo: posicionH=1, posicionV=3

En este caso los invasores están aún en la fila cero y se van desplazando hacia la izquierda, durante el movimiento la nave ha conseguido eliminar a uno de ellos. Además la nave ha efectuado un disparo que aún no alcanzado la fila que ocupan los invasores y se encuentra en la fila 3.

A continuación se describen brevemente el resto de los bloques. Los bloques en línea discontinua representan bloques opcionales y no se describen aquí.

- Control VGA. Recibe la información del color que debe dibujar en cada momento, y se encarga de representarlo en el monitor. Realizará el barrido por la pantalla, informará del punto en el que quiere dibujar y pedirá el valor del color de dicho punto.
- Control Invasores. Controla los invasores que aparecerán en la parte superior de la pantalla. Los invasores tienen un funcionamiento independiente, y sólo reciben como entrada la posición del disparo si lo hubiera, para poder detectar si alguno de los invasores tiene que desaparecer. Se mueven lateralmente hasta que uno de ellos alcanza el límite de la pantalla, con la cadencia que indica el temporizador.
- Control Nave. Controla la posición de la nave. Recibe órdenes de los botones de control de la nave (Izquierda y Derecha) y comunica la posición de la nave al control de disparo.
- Control Disparo. Controla la posición del disparo de la nave (recuérdese que sólo puede haber un disparo de la nave en la pantalla). Recibe órdenes del botón de Disparo, y recibe la información de la posición horizontal de la nave. Se mueve verticalmente con la cadencia que indica el temporizador.
- Dispositivo antirrebotes. Los botones sufren un efecto no deseado llamado “rebote”. Consiste en un efecto mecánico por el cuál, al ser pulsado el botón, no se produce un único cambio de estado en la señal que controla, sino que produce un número indeterminado de ellos antes de adoptar la posición final. El resultado es un tren de pulsos. Es necesario procesar las señales de entrada procedentes de los botones para eliminar este efecto.
- Temporizadores. Como se ve, están a lo largo de todos los bloques. Los distintos objetos que forman parte del juego se mueven a distintas velocidades. Los invasores son los más lentos y el disparo debe ser el más rápido. Los temporizadores marcarán los tiempos de cada elemento.
- Estado. Consiste en una máquina de estados que controlará el videojuego. Algunos estados posibles serán INICIO, TEST, JUGANDO, etc. En función del estado, el resto de los bloques se comportarán de distinta manera.

Se ha descrito la funcionalidad básica del circuito que se va a diseñar. Es casi seguro que habrá comportamientos concretos que habrán quedado por definir. El alumno deberá completar estos aspectos del diseño como crea conveniente, describiendo apropiadamente sus decisiones en el código.

4. PRÁCTICA 1. CONTROL DEL MONITOR VGA

4.1. PARTE 1: TABLERO DE AJEDREZ

En esta primera parte de la práctica 1 se va a llevar a cabo un diseño sencillo, que nos ayude a comprobar cómo funciona el controlador VGA. Para ello, el objetivo de esta parte será conseguir dibujar un tablero de cuadrados en el monitor.

Los alumnos no deben pasar al diseño de otros bloques hasta que éste no funcione correctamente, dado que es el bloque que permitirá mostrar los resultados del resto de la lógica.

El proceso que se va a seguir es el siguiente:

- Descripción VHDL del circuito.
- Simulación del diseño mediante la herramienta ModelSim.
- Síntesis del diseño mediante la herramienta ISE.
- Programación de la FPGA y pruebas.

Recordamos cuales son las entradas y salidas de la entidad VGA.

```
entity vga is
  port (Clk          : in std_logic;
        Reset        : in std_logic;
        color        : in std_logic_vector(2 downto 0);
        Hsync        : out std_logic;
        Vsync        : out std_logic;
        R,G,B        : out unsigned(3 downto 0);
        X            : out unsigned(9 downto 0);
        Y            : out unsigned(9 downto 0));
end vga;
```

Listado 4.1. Entidad del controlador VGA.

El objetivo de esta primera parte del diseño es familiarizarse con el uso de esta entidad VGA y dibujar un sencillo patrón en el monitor. En concreto se pretende dibujar un tablero de cuadrados blancos y negros de 32 puntos de lado cada uno. Para ello tendremos que ir controlando cuando las salidas X e Y avanzan 32 puntos.

Para realizar este ejercicio, debemos crear una entidad que recoja el valor de las salidas X e Y de la entidad VGA y devuelva para los tres componentes de color el valor '1' o '0' (cada uno de los bits del puerto de salida *color* en la entidad *formatoVGA*), de forma que en el monitor aparezcan dibujados cuadrados blancos y negros formando un tablero (consultar el Anexo III con colores VGA).

Las entradas y salidas de la entidad podrían ser las que aparecen en el Listado 3.2.

```
entity formatoVGA is
  port (
    Color  : out std_logic_vector(3 downto 0);
    X      : in unsigned(9 downto 0);
    Y      : in unsigned(9 downto 0)
  );
end formatoVGA;
```

Listado 4.2. Entidad que da valor a la Entrada Data del controlador VGA.

Todo esto (vga y formatoVGA) hay que instanciarlo en otra entidad, que será el diseño completo de esta parte 1. Las entradas y salidas de esta entidad de mayor nivel jerárquico serían únicamente la señal de reloj y el reset (entradas) y las señales de control de monitor, R, G, B, HSync, y VSync (salidas). La entidad podría ser similar a la mostrada en el Listado 3.3.

```
entity SpaceInv is
  port (
    Clk        : in  std_logic;
    Reset      : in  std_logic;
    Hsync      : out std_logic;
    Vsync      : out std_logic;
    R,G,B      : out unsigned(3 downto 0)
  );
end SpaceInv;
```

Listado 4.3. Entidad de más alto nivel del diseño.

El esquema del resultado final, sería parecido al que aparece en la Figura 3.1.

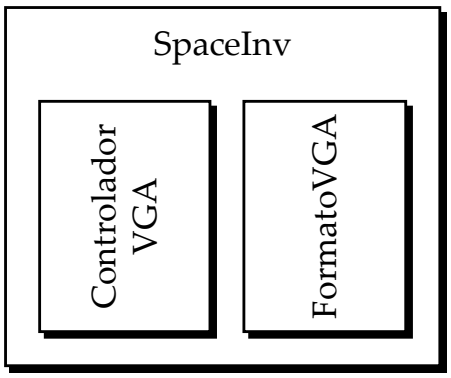

Figura 4.1. Diagrama de bloques de la parte 1.

4.2. SIMULACIÓN

Cuando se tenga descrito el sistema en VHDL, se procederá a la simulación del mismo mediante la herramienta ModelSim. Se recomienda usar la versión ModelSim 10 puesto que es la versión que está instalada en el laboratorio. Con esta herramienta se podrá simular el diseño, pudiéndose visualizar las entradas generadas por el banco de pruebas, las salidas generadas por el circuito y las señales internas que se deseen.

Para realizar la simulación es preciso realizar un Banco de Pruebas. Un banco de pruebas es una descripción VHDL que genera los estímulos de entrada necesarios para simular un circuito. Se trata de una entidad sin entradas ni salidas, y una arquitectura en la que se instancia el circuito que se quiere simular. Además, consta de varios procesos que generan las distintas señales de entrada del circuito: reloj, reset, etc.

La simulación debe contemplar los siguientes aspectos:

- Activación del reset
- Señal de reloj.
- Valor de las distintas entradas del sistema.

En nuestro caso, las únicas entradas van a ser las señales de reloj y reset. En los siguientes diseños habrá que ir añadiendo el valor de otras entradas.

Hay que tener en cuenta que simular el circuito significa hacer un barrido completo de la pantalla, por lo que puede ser necesario simular muchos ciclos de reloj.

Los alumnos deberán mostrar a los profesores el resultado de la simulación antes de proceder a la síntesis y la programación.

4.3. SÍNTESIS

Tras comprobar en la simulación que el comportamiento del circuito es el deseado, se procederá a realizar el proceso de síntesis mediante la herramienta Vivado de Xilinx. La versión instalada en el laboratorio es la 2015.4. En este proceso se debe hacer hincapié en los siguientes puntos:

- Asignación de entradas y salidas a pines de la FPGA.
- Revisión de los resultados:
  - Compruebe los posibles mensajes de advertencia que genera el sintetizador. Muchos de esos mensajes pueden estar advirtiendo de prácticas de diseño que generarán un error.

ASIGNACIÓN DE PINES

Es especialmente importante la asignación de entradas y salidas a pines de la FPGA, ya que de no hacerlo correctamente **PODRIA DAÑARSE LA FPGA COMO CONSECUENCIA DE CORTOCIRCUITOS.**

Para evitar problemas potenciales de daño a los equipos, el fichero de asignación de pines, denominado "spaceinv.xdc", se indica en el Anexo II. Para que la asignación de pines tenga efecto basta añadir al proyecto el fichero "spaceinv.xdc" como un fichero fuente más. Este fichero estará disponible en AGII. Esta asignación de pines es para el sistema completo, con lo cual será necesario comentar aquellas asignaciones correspondientes a pines que aún no se utilizan.

RESULTADOS DE LA SÍNTESIS

Tras la síntesis se deben comprobar los resultados. Consulte el tutorial de Vivado (disponible en Aula Global) para saber dónde encontrar dicha información. La información que los alumnos deben buscar es la siguiente:

- Número y porcentaje de celdas lógicas utilizadas (LUTs).
- Número de biestables (flip-flops) utilizados. Comprobar que no se han usado más de los necesarios, ya que esto puede ser causa de que el diseño no funcione correctamente.
- Comprobar que no se han generado latches no deseados.
- Comprobar que la asignación de pines se ha realizado correctamente.

4.4. PROGRAMACIÓN Y PRUEBAS

Tras la síntesis, se procederá a la programación de la FPGA y a las pruebas del circuito. Los alumnos deberán enseñar el circuito en funcionamiento a los profesores.

4.5. RECOMENDACIONES

- Es conveniente que los nombres de las entidades coincidan con los de los ficheros que las contienen y que en cada fichero sólo haya una entidad y su correspondiente arquitectura. Los ficheros deben tener extensión ".vhd".
- Se recuerda que las señales que generan los pulsadores e interruptores de la placa son activas por nivel alto.
- Es importante respetar el nombre de los puertos de entrada y salida de las entidades, ya que eso facilita la detección de errores en caso de que aparezcan problemas.
- Se recuerda a los alumnos que para poder utilizar los tipos de datos y sus operaciones hay que incluir las siguientes líneas al comienzo de los ficheros VHDL.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
```

El dibujo final que debemos obtener debe ser parecido al de la Figura 3.2.

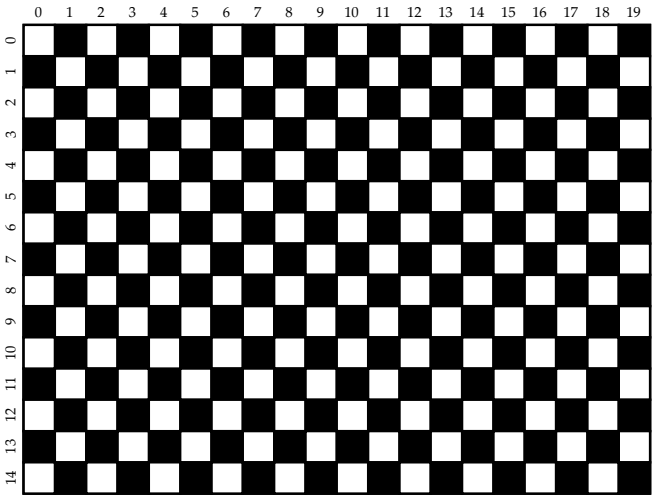

Figura 4.2. Tablero de cuadros en el monitor.

Habrá que incluir este diseño en todos los subsiguientes, ya que servirá para comprobar que la visualización del monitor es correcta.

4.6. PARTE 2: GENERACIÓN DE UNA PANTALLA FIJA

Una vez que se ha conseguido dibujar el tablero en el monitor, el siguiente paso que se va a dar es el dibujo de una pantalla que represente un instante cualquiera de una partida. Además, vamos a realizar una pequeña lógica de control, que en función del valor del botón Test, muestre bien la pantalla de cuadros diseñada en la parte anterior, bien la posición estática de la partida.

Para ello se van a definir unas señales, que serán las que guarden las posiciones de los elementos del juego, y se les va a dar un valor estático.

El nombre de las señales puede elegirse a voluntad. En cualquier caso, conviene que los nombres sean representativos del contenido de las señales.

El primer paso será elaborar una lógica que traduzca las coordenadas de los puntos del monitor a dibujar, a posiciones de partida. Para ello se recomienda que se traduzcan las coordenadas XY de pantalla, a coordenadas basadas en el hecho de que los elementos a dibujar tienen 32x32 píxeles, es decir, que se traduzcan las coordenadas de las filas y las columnas del monitor (píxeles), a filas y columnas de la pantalla de juego. Con una resolución de 640x480 píxeles el monitor tiene 20 columnas y 15 filas de 32x32 puntos cada una. Estás nuevas coordenadas X'Y', que se corresponden con las filas y columnas de la zona de juego, pueden numerarse iniciándose según su posición absoluta en el monitor (la primera fila sería la 14), o según su posición en la zona de juego (la primera fila o columna sería la 0).

Una vez que tenemos las coordenadas X'Y' (fila y columna) del punto que queremos dibujar dentro de partida, hay que comprobar si dichas coordenadas coinciden con las de algún elemento. Para ello, la información de dónde están situados los elementos tiene que llegar al bloque *formatoVGA*.

La entidad principal en este caso se reutilizará para ir ampliando el diseño. Las entradas y salidas que se van a necesitar para esta parte serían las que se muestran en el Listado 4.4.

```
entity SpaceInv is
  port (
    Clk        : in  std_logic;
    Reset      : in  std_logic;
    Test       : in  std_logic;
    Vsync      : out std_logic;
    Hsync      : out std_logic;
    R,G,B      : out unsigned(3 downto 0)
  );
end SpaceInv;
```

Listado 4.4. Entradas y salidas de la entidad SpaceInv.

4.7. SIMULACIÓN

Al igual que antes, cuando se tenga descrito el sistema en VHDL, se procederá a la simulación del mismo mediante la herramienta ModelSim.

Como Banco de Pruebas puede usarse él del diseño anterior, añadiendo sentencias para dar valor a la entrada *Test*.

Hay que tener en cuenta que la nueva simulación debe contemplar las dos posibilidades de funcionamiento del circuito, es decir, el modo de dibujo de un instante fijo de la partida, o el dibujo del tablero de cuadros.

Los alumnos deberán mostrar a los profesores el resultado de la simulación antes de proceder a la síntesis y la programación.

4.8. SÍNTESIS

Tras comprobar en la simulación que el comportamiento del circuito es el deseado, se procederá a realizar el proceso de síntesis.

La señal de Test está conectada al interruptor SW0 de la placa (esquina inferior derecha de la placa, cuando tenemos la placa delante y podemos leer correctamente la serigrafía de la placa).

4.9. RESULTADOS DE LA SÍNTESIS

De nuevo, deben hacerse las mismas comprobaciones tras la síntesis, ya que es necesario comprobar que se han obtenido los resultados que esperábamos. Resultados inesperados podrían indicar errores de diseño no detectados, errores al elegir los ficheros .vhd para síntesis, etc.

4.10. PROGRAMACIÓN Y PRUEBAS

Tras la síntesis, se procederá a la programación de la FPGA y a las pruebas del circuito. Los alumnos deberán enseñar el circuito en funcionamiento a los profesores. Es necesario seguir las mismas recomendaciones que en la primera parte de la práctica.

5. PRÁCTICA 2. INVASORES Y NAVE

En esta práctica se va a modificar el diseño que se hizo en la práctica anterior, para añadirle la funcionalidad que permitirá incorporar al juego los invasores y la nave. No obstante, es muy recomendable guardar la versión correcta de la práctica 1, para evitar posibles pérdidas del trabajo realizado. Al final de la práctica, los alumnos deben haber conseguido un diseño, implementado en la FPGA, en el que se vea a los invasores moverse con su correspondiente pauta de movimiento, y a la nave, que se moverá respondiendo a los impulsos de los botones de *izquierda* y *derecha.* El movimiento de los invasores comenzará al pulsar el botón *Inicio.*

Además se deben incluir temporizadores que permitan marcar las frecuencias con que estos objetos se mueven. Recordamos que la frecuencia del oscilador de la placa es de 100 Mhz, frecuencia que tiene que ser reducida para que el movimiento de los objetos pueda ser visto por el ojo humano.

SPACEINV

Se modificará la entidad de más alto nivel *Spaceinv* para incorporar las entradas de los botones de *inicio*, *izquierda* y *derecha*.

```
entity SPACEINV is
  port(
    Clk        : in  std_logic;
    Reset      : in  std_logic;
    Test       : in  std_logic;
    Inicio     : in  std_logic;
    Izquierda  : in  std_logic;
    Derecha    : in  std_logic;
    Vsync      : out std_logic;
    Hsync      : out std_logic;
    R,G,B      : out unsigned(3 downto 0)
  );
end SPACEINV;
```

Listado 5.1. Entidad SPACEINV modificada

5.1. INVASORES

En una primera aproximación se implementarán sólo los invasores y se probará el circuito. Para ello, hay que añadir al diseño dos bloques más, un temporizador y los invasores propiamente dichos.

TEMPORIZADOR

El sistema tiene, en principio, tres elementos que se mueven a velocidades diferentes. Los invasores, que son los más lentos, el disparo, que es el más rápido y la nave, que se controla mediante botones. Hay que controlar la velocidad de movimiento de los elementos que se mueven automáticamente, los invasores y el disparo. Para ello se utilizará un temporizador que emita un pulso cada cierto tiempo.

Como va a ser necesario mover dos elementos, invasores y disparo, se van a necesitar dos temporizadores. Como los dos temporizadores van a tener la misma funcionalidad, pero distinta constante de tiempo, se realizará un diseño genérico que se utilizará para los dos casos.

Se deja libertad al alumno para diseñar este bloque, teniendo en cuenta que debe cumplir:

- Debe tener entradas de reloj y de reset.
- Debe tener señales de habilitación y de inicialización síncrona.
- Debe tener una salida, que muestre el valor de la cuenta. Fuera del temporizador se podrá comparar esta señal con el límite de cuenta, de forma que se emita un pulso periódico. Este pulso durará un ciclo de reloj y se utilizará como señal de habilitación en el bloque de control correspondiente, y como señal de inicialización síncrona del contador. En ningún caso debe generar una nueva señal de reloj.

- Debe ser genérico, siendo configurable la constante de tiempo.

Para diseñar el temporizador de los invasores, lo primero que necesitamos es saber cuántos ciclos de reloj tiene que contar. Se sugiere que los invasores avancen una posición cada 0,5 segundos más o menos. Habrá que determinar qué número de bits se necesitan para contar 0,5 segundos, y pasarle ese valor como genérico al contador. En el Listado 4.2 tenemos una posible entidad del temporizador, con sus entradas y salida correspondiente.

```
entity temporizador is
  generic(
    Ancho     : integer := 8
  );
  Port (
    Clk       : in std_logic;
    Reset     : in std_logic;
    Clear     : in std_logic;
    Enable    : in std_logic;
    Cuenta    : out unsigned(Ancho-1 downto 0)
  );
end temporizador;
```

Listado 5.2. Posible entidad para el temporizador

INVASORES

Debe diseñarse una entidad llamada INVASORES que implemente el comportamiento de los invasores (que se describió en apartados anteriores) y lo transmita al bloque formatoVGA para su visualización.

Aunque hay múltiples modos de realizar el diseño, se recomienda seguir las siguientes indicaciones:

- Se suponen como entradas el reloj, el reset y una señal de inicialización síncrona. La entrada de inicialización servirá para que los invasores vuelvan a su posición inicial. Se conectará la entrada *inicio* del circuito a esta entrada de inicialización.
- Una señal almacenará la posición vertical (fila) de los invasores. Esta señal será un entero entre 0 y 14.
- Otra señal almacenará los invasores que queden vivos. Los bits que estén a '1' serán invasores. El movimiento horizontal de los invasores se conseguirá desplazando los bits como si se tratara de un registro de desplazamiento. Al iniciar la partida, los invasores serán 10 y se encontrarán situados en la fila 0 en uno de los extremos de la pantalla.
- Los invasores se moverán de extremo a extremo de cada fila y cuando alcancen el extremo bajarán una fila.
- La información que debe salir al exterior será el número de fila donde se encuentran los invasores y el registro de invasores.
- Debe incorporarse al diseño una entrada de habilitación para que el circuito siga el ritmo marcado por el temporizador diseñado anteriormente.

5.2. NAVE

Se diseñará una entidad llamada NAVE que implementará el comportamiento de la nave. En este bloque es suficiente con almacenar la posición horizontal (columna) de la nave, y modificarla según los puertos de entrada *derecha* e *izquierda*.

La respuesta que se desea de los botones es que por cada pulsación del botón, la nave se mueva una posición. No se producirá un nuevo desplazamiento hasta que el jugador suelte el botón y lo vuelva a pulsar. Esto equivale a decir que la lógica de control de la nave debe detectar flancos en los botones de movimiento.

La manera de detectar flancos en un circuito síncrono es mediante la utilización de una estructura lógica denominada *conformador de pulsos*, cuyo esquema se muestra en la Figura 5.1.

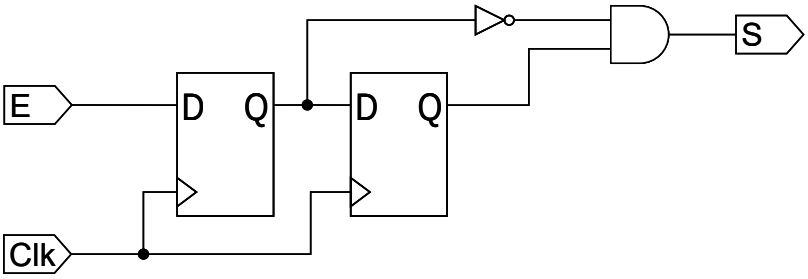

Figura 5.1. Conformador de pulsos

Este conformador de pulsos, junto con el posible temporizador que filtre los rebotes de los botones, y del que se hablará más adelante, deberían encontrarse en el bloque *gestionEntradas*.

Con este esquema, cuando la señal de entrada E pasa de '0' a '1', es decir, se pulsa el botón, la salida S no varía. Pero al soltar el botón, es decir, cuando E pase de '1' a '0', la salida S se pondrá a '1' durante un único ciclo de reloj. Este circuito detecta, por tanto, flancos de bajada en la señal E, y generará un único pulso por cada pulsación del botón.

Es recomendable incluir en el bloque nave una entrada para su inicialización, que sitúe la nave en la posición central.

5.3. DISPOSITIVO ANTIRREBOTES (OPCIONAL)

De manera opcional, los alumnos podrán diseñar un bloque que elimine el efecto de los rebotes de los botones de izquierda y derecha. Se recomienda intentar la implementación de este bloque sólo en el caso de que ya se haya completado el resto.

Los rebotes de los botones ocasionan que, aunque el jugador pulse un botón una sola vez, la nave se mueva varias veces. Esto se debe a que, debido a efectos mecánicos, se origina un tren de pulsos en la señal gobernada por el botón antes de que estabilice.

Se puede evitar este efecto de varias maneras. Una de ellas se basa en la idea de que el jugador no puede pulsar los botones muy deprisa, de modo que las pulsaciones muy seguidas son rebotes. Se trataría de diseñar un circuito que sólo acepte pulsaciones cada cierto tiempo, por ejemplo, cada 100ms.

5.4. RECOMENDACIONES

- Se recomienda diseñar primero los invasores, probarlos, y posteriormente incluir el diseño de la nave.
- Los invasores deben tener un temporizador que les marque la velocidad de movimiento. En la siguiente práctica, el disparo llevará otro temporizador. Cada uno de ellos será una instancia del temporizador diseñado en esta práctica, pero con distinta constante de tiempo.
- La comunicación entre el temporizador y los invasores debe ser síncrona, es decir, los dos bloques deben usar el mismo reloj, y se deben comunicar mediante una señal de habilitación.

El dispositivo antirrebotes se debe implementar sólo cuando se haya terminado con el resto de la funcionalidad.

6. PRÁCTICA 3. DISPAROS Y DISEÑO COMPLETO

En esta práctica se añadirán al diseño las partes que le faltan para completar el juego. Hay que añadir el bloque de disparo y una máquina de estados para controlar el estado del juego. Al final de la práctica, se debería tener implementado el sistema básico completo, funcionando en la FPGA.

En el caso de que algún grupo no tuviera suficiente tiempo para completar el diseño, podrá terminarlo en la práctica 4.

SPACEINV

Se modificará la entidad de más alto nivel *Spaceinv* para incorporar las entradas del botón de *disparo*.

```
entity SPACEINV is
  port(
    Clk        : in  std_logic;
    Reset      : in  std_logic;
    Inicio     : in  std_logic;
    Izquierda  : in  std_logic;
    Derecha    : in  std_logic;
    Disparo    : in  std_logic;
    Vsync      : out std_logic;
    Hsync      : out std_logic;
    R, G, B    : out unsigned(3 downto 0)
  );
end SPACEINV;
```

Listado 6.1. Entidad SPACEINV modificada

DISPARO

Cada vez que el jugador pulsa el botón de *disparo*, una bala saldrá de la nave y se desplazará verticalmente hasta llegar a la parte superior de la pantalla. Si la bala coincide con algún invasor, éste y la bala deberán desaparecer de la pantalla.

Debe implementarse un bloque, cuya entidad se llamará DISPARO, que gestionará el movimiento de la bala y debe cumplir con los siguientes requisitos:

- Debe salir una bala cuando se pulse el botón de disparo. No debe salir otra bala hasta que la primera haya desaparecido de la pantalla.
- La bala debe moverse a un ritmo distinto al de la nave o los invasores. Para ello, será necesario instanciar de nuevo el bloque temporizador que se diseñó en la práctica anterior.
- Obsérvese que no siempre hay un disparo en la pantalla. Debe haber un mecanismo que contemple esta situación.
- Debe tener una señal de inicialización, que hará que la bala desaparezca de la pantalla.

Lo más sencillo puede ser almacenar las coordenadas X e Y de la bala, y sacarlas al exterior del componente. La coordenada X se copiará de la posición de la nave en el momento del disparo y la coordenada Y empezará valiendo 0 y se irá incrementando hasta llegar a 14.

Obsérvese que, al no poderse realizar un nuevo disparo hasta que el anterior no desaparece, no es necesario utilizar el conformador de pulsos ni un dispositivo antirrebotes para el botón de disparo.

Al igual que con el bloque de los invasores diseñado en la práctica anterior, es necesario incluir un temporizador que marque la velocidad del disparo.

La detección de que la bala ha impactado un invasor puede realizarse en el bloque de los invasores. Para ello es necesario que lleguen a ese bloque las coordenadas de la bala, y que a su vez devuelva una señal que indique si se ha producido el impacto, de forma que el control del disparo sepa que la bala tiene que desaparecer.

COMUNICACIÓN ENTRE BLOQUES

Debe haber cierto intercambio de información entre los bloques controlDISPARO, controlINVASORES, controlNAVE y formatoVGA:

- El bloque formatoVGA debe recibir información de los demás para realizar la representación visual.
- El bloque controlINVASORES necesita la información de la posición del disparo para comprobar si se ha impactado a un invasor y eliminarlo en caso afirmativo.
- El bloque controlDISPARO necesita tener acceso a la posición de la nave, para establecer su coordenada X en el momento del disparo, y debe saber si se ha eliminado a un invasor para terminar su recorrido.

Para implementar estas transmisiones de información, los bloques dispondrán de entradas con la información que necesiten y de salidas con la información relevante para otros bloques.

Se deberán modificar los bloques ya diseñados para implementar la comunicación necesaria entre ellos.

MÁQUINA DE ESTADOS

Es necesario que el sistema completo tenga un mecanismo para controlar su estado. Esto se llevará a cabo por medio de una máquina de estados. Se sugieren los estados siguientes, aunque podrían implementarse algunos más:

- TEST. Estado en el que se obliga a la pantalla a mostrar un patrón fijo (por ejemplo pantalla con el tablero de cuadros). De este estado no se sale, salvo que se pulse el botón *inicio*, en cuyo caso se pasará al estado INICIO.
- INICIO. Estado por el que se pasa para iniciar una partida. La máquina irá a este estado cuando se pulse el botón *inicio*.
- JUGANDO. Es el estado en el que el jugador juega. Todos los bloques funcionan normalmente. Se sale de este estado cuando todos los invasores han sido destruidos o cuando algún invasor llega a la parte baja de la pantalla (fila 14). En el primer caso se pasará al estado GANADO y en el segundo, al estado PERDIDO.
- GANADO. Estado al que se llega cuando los invasores han sido destruidos. Puede optarse por comenzar una nueva partida, parar el circuito o ir al estado de TEST.
- PERDIDO. Estado al que se llega cuando los invasores alcanzan a la nave. Puede optarse por parar el juego o ir al estado de TEST.

En realidad, hay muchas opciones para el comportamiento de esta máquina de estados. Los alumnos deberán decidir el comportamiento que van a implementar, y en consecuencia, el diagrama de estados y las señales que se van a controlar. **En la memoria de las prácticas deberá incluirse el diagrama de estados y las señales que se van a controlar.**

Se recomienda implementar este bloque como procesos dentro de la entidad de más alto nivel (SPACEINV), ya que hacerlo así permite mayor flexibilidad a la hora de realizar modificaciones. Es mejor este método que realizar la implementación mediante un componente.

6.1. SÍNTESIS

La asignación de pines es igual que la de la práctica anterior, salvo que se ha añadido la entrada *disparo*, y se ha asignado al botón principal de la placa.

Al igual que en la práctica anterior, se deberán consultar los datos producidos por la síntesis sobre la utilización de recursos de la FPGA y los requisitos temporales. **En este caso y dado que el diseño ya está completo, se reportarán estos datos en la memoria final que se entregará a los profesores.**

7. PRÁCTICA 4. AMPLIACIONES

El juego admite muchas ampliaciones. Para llevarlas a cabo, pueden utilizarse el resto de los elementos de la placa que se ha venido utilizando.

A continuación, se enumeran algunas de las ampliaciones que se pueden realizar, aunque se anima a los alumnos a que hagan sus propias ampliaciones.

- Dibujar cada elemento de un color distinto.
- Optimizaciones generales. Modificaciones del diseño para mejorarlo en área y/o tiempo. Por ejemplo, el sistema utiliza varios temporizadores, que podrían fundirse en uno solo.
- Los invasores disparan. Puede ser complicado, porque hay que disparar al azar. Podría hacerse de modo que si un marciano pasa por la posición de encima de la nave y no hay otro disparo en el aire, el marciano dispara.
- Incrementar la dificultad del juego a medida que se matan oleadas de marcianos. Para ello. se puede variar la constante de tiempo del temporizador de los invasores para que vayan más deprisa.
- Dibujar iconos para los marcianos y la nave. Estos iconos serán matrices de valores constantes (dibujo del icono), de tamaño 32x32 que enmascaran los píxeles correspondientes a un punto de la pantalla de juego.

La evaluación tendrá en consideración la dificultad de las ampliaciones implementadas.

8. MEMORIA DE PRÁCTICAS

Se abrirá una tarea en Aula Global para realizar la entrega los ficheros .vhd diseñados el día de la última sesión de prácticas.

Una semana después de la última sesión de prácticas deberá entregar una memoria donde se detalle el diseño realizado. La memoria deberá incluir:

- Descripciones VHDL de los diseños realizados y de los bancos de pruebas utilizados.
- Exposición breve de cómo se han realizado los diseños y de los aspectos que se consideren importantes.
- Descripción de las ampliaciones realizadas.
- Resultados finales de síntesis (área y frecuencia máxima de reloj)
- No es necesario entregar simulaciones de los diseños.

Anexo I. FUNCIONAMIENTO DE UN MONITOR

Un monitor VGA se controla por medio de tres señales analógicas, que especifican la intensidad de los colores (RGB, Red Green Blue), y dos señales de sincronismo, horizontal y vertical.

Para dibujar una imagen en la pantalla, hay que realizar un barrido de líneas, empezando por la esquina superior izquierda y barriendo hacia la derecha y hacia abajo. Es decir, el monitor va dibujando los datos en líneas horizontales. Después de cada línea horizontal es necesario dar un pulso a la señal de sincronismo horizontal HSYNC, y después de cada pantalla, cuando están dibujadas todas las líneas horizontales, hay que dar un pulso a la señal de sincronismo vertical VSYNC. En la Figura 8.1 se muestran las formas de onda de un barrido de línea horizontal.

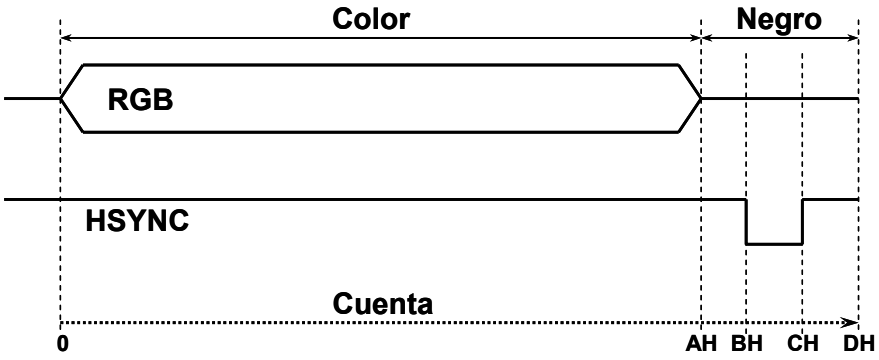

Figura 8.1. Formas de onda de un barrido de línea horizontal

Para implementar el barrido de cada línea, se utilizará un contador, que se reiniciará al comienzo de cada línea, y que permitirá temporizar la señal HSYNC apropiadamente.

El proceso de barrido de línea se repite tantas veces como líneas haya en la pantalla, y después se procede a realizar el sincronismo vertical. La señal VSYNC tiene una forma equivalente a HSYNC, pero el pulso se produce tras haberse barrido todas las líneas horizontales, como puede observarse en la Figura 8.2.

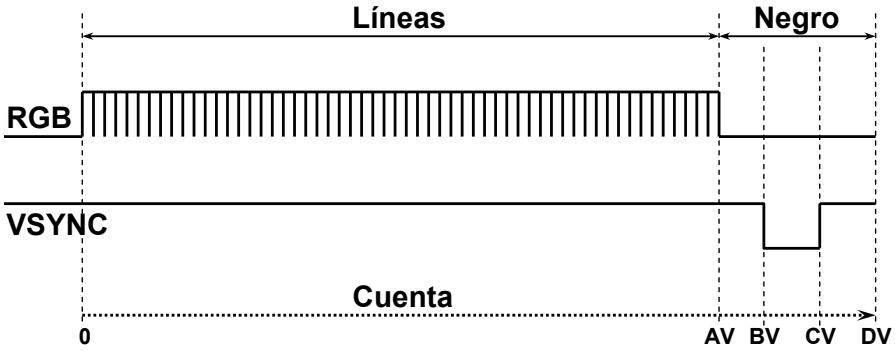

Figura 8.2. Formas de onda de un barrido de pantalla

Para implementar el barrido de líneas, se utilizará otro contador, que se reiniciará al comienzo de cada pantalla, y que permitirá temporizar la señal VSYNC.

La resolución que se va a utilizar será de 640x480. Para esta resolución y suponiendo una frecuencia de reloj de 25 MHz, los tiempos correspondientes a cada evento y los valores correspondientes de los contadores se muestran en la Tabla siguiente:

Horizontal			Vertical		
	Tiempo	**Valor de cuenta (25MHz)**		**Tiempo**	**Valor de cuenta (25MHz)**
AH	25,60 us	640	**AV**	15,36 ms	480
BH	26,16 us	654	**BV**	15,68 ms	490
CH	30,08 us	752	**CV**	15,744 ms	492
DH	32,00 us	800	**DV**	16,672 ms	521

Tabla 8.1. Tiempos de sincronismo

Obsérvese que los valores de cuenta en los puntos AH y AV son 640 y 480 respectivamente, y se corresponden con el número de píxeles de anchura y altura de la pantalla. Por tanto los valores de los contadores representarán las coordenadas X e Y del píxel que se está representando en un momento dado.

Asegúrese de cumplir los siguientes puntos:

1.- Comprueba que los tiempos son los mismos que en la Tabla 8.1.

2.- Entre los periodos AH-DH y AV-DV el valor de cada componente de color (R, G y B) debe ser 0000, como se muestra en las figuras 8.1 y 8.2.

3.- El pulso de sincronismo HSync debe realizarse siempre, incluso en el periodo de sincronización de VSync.

Anexo II. FICHERO XDC

El fichero XDC (Xilinx Design Constraints) es un fichero ASCII que especifica las restricciones del diseño. A continuación, se indican los contenidos del fichero XDC utilizado en este diseño. Copie estos contenidos en un fichero con la extensión .xdc y añádalos a su proyecto.

```
## Clock signal
set_property PACKAGE_PIN W5 [get_ports Clk]
set_property IOSTANDARD LVCMOS33 [get_ports Clk]
create_clock -period 10.0 -name Clk100 -waveform {0.0 5.0} -add [get_ports Clk]

## Buttons
#set_property PACKAGE_PIN T18 [get_ports Fire]
#set_property IOSTANDARD LVCMOS33 [get_ports Fire]
#set_property PACKAGE_PIN W19 [get_ports Left]
#set_property IOSTANDARD LVCMOS33 [get_ports Left]
#set_property PACKAGE_PIN T17 [get_ports Right]
#set_property IOSTANDARD LVCMOS33 [get_ports Right]
#set_property PACKAGE_PIN U17 [get_ports Start]
#set_property IOSTANDARD LVCMOS33 [get_ports Start]

## Switches
set_property PACKAGE_PIN V17 [get_ports Reset]
set_property IOSTANDARD LVCMOS33 [get_ports Reset]
#set_property PACKAGE_PIN V16 [get_ports Test]
#set_property IOSTANDARD LVCMOS33 [get_ports Test]

## VGA Connector
set_property PACKAGE_PIN G19 [get_ports {R[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {R[0]}]
set_property PACKAGE_PIN H19 [get_ports {R[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {R[1]}]
set_property PACKAGE_PIN J19 [get_ports {R[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {R[2]}]
set_property PACKAGE_PIN N19 [get_ports {R[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {R[3]}]
set_property PACKAGE_PIN N18 [get_ports {B[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {B[0]}]
set_property PACKAGE_PIN L18 [get_ports {B[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {B[1]}]
set_property PACKAGE_PIN K18 [get_ports {B[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {B[2]}]
set_property PACKAGE_PIN J18 [get_ports {B[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {B[3]}]
set_property PACKAGE_PIN J17 [get_ports {G[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {G[0]}]
set_property PACKAGE_PIN H17 [get_ports {G[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {G[1]}]
set_property PACKAGE_PIN G17 [get_ports {G[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {G[2]}]
set_property PACKAGE_PIN D17 [get_ports {G[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {G[3]}]
set_property PACKAGE_PIN P19 [get_ports Hsync]
set_property IOSTANDARD LVCMOS33 [get_ports Hsync]
set_property PACKAGE_PIN R19 [get_ports Vsync]
set_property IOSTANDARD LVCMOS33 [get_ports Vsync]
```

El presente fichero XDC especifica fundamentalmente dos tipos de restricciones:

- El periodo de la señal de reloj Clk, que debe coincidir con el que proporciona el oscilador presente en la placa (100 MHz)
- Los pines de la FPGA asociados a las entradas y salidas de nuestro diseño, y que están conectados en la placa a los elementos correspondientes (botones, pulsadores, conector VGA, etc.). **Si en las fases iniciales del diseño no utiliza todos los pines especificados en el fichero XDC, deberá comentar las líneas correspondiente utilizando el símbolo #**
- **El nombre de los** puertos del fichero XDC debe coincidir **exactamente, incluyendo mayúsculas y minúsculas**, con el de los ficheros VHDL.

Anexo III. COLORES VGA

El código del color que se quiere dar al píxel consiste en tres vectores de 4 bits cada uno, R, G y B. Cuando todos los bits de los tres componentes son '1' se consigue el color blanco, mientras que cuando todos son '0' se obtiene el color negro. En la siguiente tabla se muestran los colores que se obtienen cuando se combinan los tres componentes sin considerar distintas intensidades. Cuando, para un componente de color determinado, aparece '1' en la tabla significa que los 4 bits están activados, mientras que si en la tabla aparece '0', significa que los 4 bits están a nivel bajo. La tabla de colores es la siguiente:

VGA_RED	VGA_GREEN	VGA_BLUE	Resulting Color
0	0	0	Black
0	0	1	Blue
0	1	0	Green
0	1	1	Cyan
1	0	0	Red
1	0	1	Magenta
1	1	0	Yellow
1	1	1	White