Sistema FPGA para control inalámbrico y visualización gráfica: base para videojuegos

De Pietro, Guillermo L.; Preves, Santiago; Puebla, Rodrigo A.
gdepietro; spreves; rodrigopuebla (@alumnos.fi.mdp.edu.ar)

Departamento de Ingeniería Electrónica y Computación
Facultad de Ingeniería – UNMDP
Mar del Plata, 7600, Argentina
Docentes: Medina, M.; Rabioglio, L.

1. Introducción

Este proyecto se desarrolla en el marco de la materia *Diseño Digital con Técnicas de Alto Nivel*, con el objetivo de implementar un videojuego estilo Pong utilizando una FPGA *Cyclone 2* y módulos ESP32. El sistema se diseña de forma modular: la FPGA ejecuta completamente el videojuego, incluyendo el procesamiento gráfico, la lógica de juego y la generación de señales VGA; mientras que las placas ESP32 actúan como controladores remotos, captando las entradas del usuario vía Bluetooth y transmitiendolas a la FPGA por cable.

El desarrollo integra conocimientos de diseño digital, lenguaje VHDL, interfaces analógicas y digitales, sincronización VGA y comunicación inalámbrica, con el fin de lograr una experiencia de juego accesible, autónoma y sin hardware adicional.

2. Objetivos

El objetivo principal consiste en desarrollar un juego interactivo con control inalámbrico, implementado sobre FPGA mediante VHDL y con salida VGA. El sistema incluye la generación interna de las señales de sincronismo y color necesarias para la visualización.

Como objetivos secundarios se plantea agregar efectos visuales y sonoros, un sistema de puntaje con representación mediante LEDs, y lógica de juego basada en vidas, con indicadores gráficos de victoria y derrota para mejorar la experiencia del usuario.

3. Desarrollo

Para facilitar la comprensión, se presenta un diagrama en bloques que organiza el análisis sin requerir el estudio detallado del código.

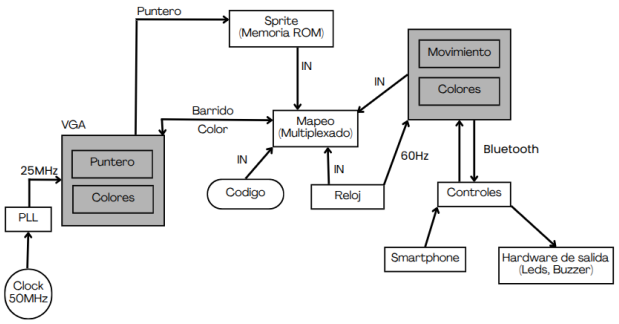

Figura 1: Diagrama el bloque del sistema

3.1. Bloque VGA

La placa permite la conexión de un monitor VGA, lo que requiere la generación de señales de sincronismo y color según el protocolo *(figura 2)*.

El sistema utiliza un reloj de 25MHz *(figura 1)* y una máquina de estados desarrollada específicamente para este proyecto, que se basa en dos contadores (uno por eje) y dos máquinas de estado asociadas. Dicho arreglo determina si cada posición se encuentra

dentro de una zona visible y habilita la salida de video en consecuencia. Este bloque se ejecuta de forma continua, ya que controla la visualización de todo el contenido generado por la FPGA.

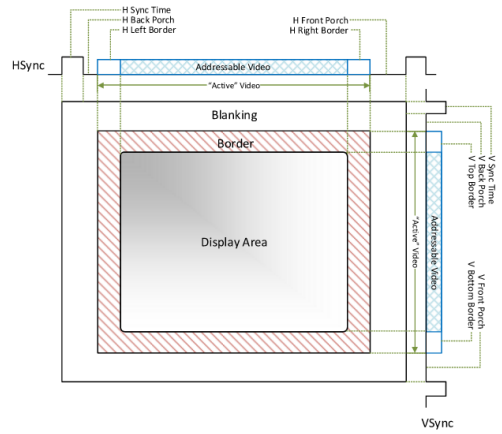

Figure 4. VGA timing specification.

Figura 2: Zonas y señales de protocolo VGA

3.2. Bloque mapeo

Este bloque es clave para el diseño, ya que delimita regiones dentro del área activa de la pantalla. Aprovecha el barrido de la señal VGA, que recorre la pantalla de izquierda a derecha y de arriba hacia abajo, para detectar la posición actual mediante los contadores horizontal y vertical.Con esa información se pueden establecer zonas con forma de rectángulos, definidos por un punto de inicio (x0, y0), un alto y un ancho. Estas áreas no muestran solo un color de fondo, sino que se procesan para formar imágenes, fijas o dinámicas, que tienen diferentes funciones en el juego.

El bloque actúa como un pseudo-multiplexor, que lee constantemente los contadores y, al estar dentro de una zona, busca el color correspondiente al píxel desde distintas fuentes (ROM o código) y lo envía al monitor a través del bloque VGA.

3.3. Bloque sprites

Un sprite es una imagen 2D utilizada en videojuegos. En este proyecto se diseñan píxel a píxel y se convierten a archivos `.mif` con datos de 30 bits (10 para cada color RGB), que son cargados luego en memoria ROM.

La visualización se sincroniza con el barrido VGA mediante una fórmula que calcula, en función de los contadores de barrido y las dimensiones del sprite, el índice del píxel a mostrar. Debido a un retardo de tres ciclos en el procesamiento del bloque, se anticipa la lectura y se calcula el valor para `x+3`.

Se utilizan cuatro imágenes distintas para representar el final del juego (trofeo, “WINNER”, calavera y “LOSER”) *(figura 4)*, ajustando su posición inicial para evitar zonas vacías y optimizar memoria. Para sprites de tamaño fijo como los corazones, esto no es necesario.

3.4. Bloque reloj

Este bloque genera y muestra el tiempo de juego sin utilizar sprites, para evitar un excesivo uso de memoria al almacenar los dígitos. Un divisor reduce la frecuencia de 25 MHz a 1 Hz y utiliza contadores para obtener unidades y decenas de segundo y minutos.

La visualización se realiza mediante una función que, usando las coordenadas del barrido y el valor del contador, determina qué segmentos encender para formar los números. Los puntos separadores permanecen activos y son gestionados desde el bloque mapeo. Este módulo también genera pulsos de 40 ns a 60 Hz para sincronizar el bloque de movimiento.

3.5. Bloque movimiento y colisión

Los objetos móviles (pelota y barras) se definen por una zona rectangular de color variable, con coordenadas dinámicas. Todo el sistema se actualiza a 60 Hz, asegurando fluidez perceptible por el humano.

Las barras se mueven en el eje vertical, controladas por señales que indican dirección y limitadas por bordes superior e inferior. La pelota varía en ambos ejes con velocidad

aleatoria, determinada por contadores desfasados. La posición se actualiza con la fórmula $P_x = P_x + V_x \cdot D_x$ (y análoga para y), donde D_x, $D_y \in \{-1, 1\}$.

Las colisiones se detectan por zonas: con bordes superior/inferior se invierte D_y, con barras se invierte D_x. Si toca los extremos laterales, se reinicia al centro y queda a la espera de la señal de inicio.

3.6. Bloque control

El control se realiza mediante dos módulos ESP32 conectados por Bluetooth a celulares, a través de la app *Arduino Bluetooth Controller*. Cada módulo recibe comandos y genera tres señales: dos para mover la barra y una para iniciar el juego, compartida entre ambos jugadores. Las señales llegan a la FPGA a través del puerto GPIO 0 mediante un cable de 40 pines.

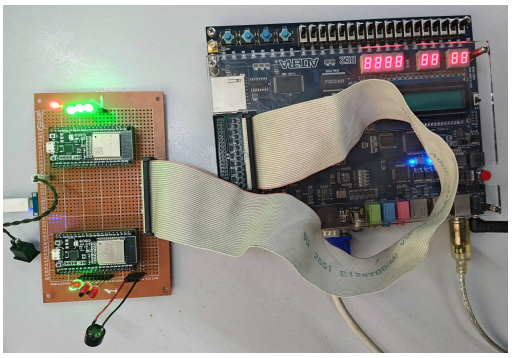

Figura 3: Hardware y conexión física

Además, la FPGA genera variables internas que se envían de vuelta a los ESP32, como los pulsos de colisión para activar un buzzer y estados de vida para controlar efectos LED. Esto permite una interacción bidireccional entre dos sistemas embebidos, uno basado en FPGA y otro basado en un microcontrolador, con el uso de Bluetooth como medio de transmisión.

4. Resultados

Se validó el sistema mediante pruebas en laboratorio y en campo, incluyendo una instancia con ocho docentes del Departamento de Electrónica. Se comprobó el funcionamiento correcto de los elementos gráficos, la lógica de juego, las colisiones y la comunicación Bluetooth. El sistema mostró buena respuesta, estabilidad visual y dinámica fluida. El sistema ocupa el 58 % de la memoria ROM disponible en la FPGA (280 kbits), ademas emplea 136 registros y 1744 componentes lógicos. Estos valores reflejan la complejidad del proyecto, cuyo diagrama .BDF final puede consultarse en la *sección de Recursos adicionales*.

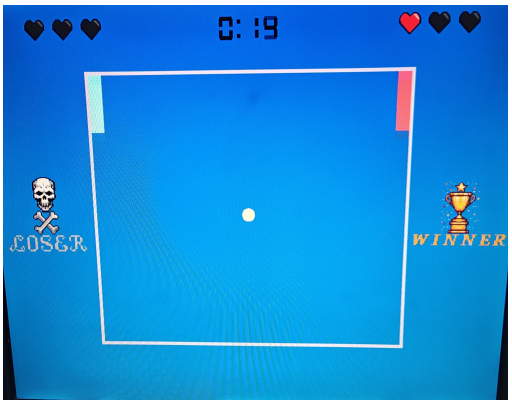

Figura 4: Interfaz final

5. Discusión y Conclusión

El proyecto integra conocimientos de diseño digital en FPGA, VHDL, comunicación Bluetooth y señales VGA. Se desarrolló un sistema funcional e interactivo, controlado de forma inalámbrica desde smartphones, sin requerir periféricos adicionales.

Se validaron todos los objetivos planteados, con un sistema robusto, visualmente atractivo y técnicamente sólido. Como posibles mejoras se propone aumentar la calidad gráfica, incluir un menú de selección, modos de juego y permitir partidas remotas mediante red Bluetooth propia o conexión Wi-Fi.

Recursos adicionales

- Repositorio en GitHub: `https://github.com/SantiagoPreves/Proyecto_estudiantil_SASE`
- BDF final: `Ver archivo en Drive`