Tecnológico de Monterrey

Reporte Solución Reto: Pong

Manuel Agustin Diaz Vicanco A01379673

Carlos Antonio Pazos Reyes A01378262

12 de marzo del 2021

Diseño con lógica programable (Gpo 101)

Profesores:

Dr. Andrés David García García y Dr. Francisco Javier Ortiz Cerecedo

Tecnológico de Monterrey

Introducción

A lo largo de estas 5 semanas, hemos tenido la oportunidad de trabajar con Intel como Socio Formador, con hardware desarrollado por Altera, empresa líder en desarrollo y manufactura de FPGAs en el mundo, la cual fue adquirida hace unos años por Intel. Esta tecnología Field Programmable Gate Array es esencial dentro de la industria de la electrónica, y nuestro Socio Formador la utiliza para prototipaje de su circuitería en diversidad de circuitos a nivel profesional.

El material que se nos prestó, la DE10-lite es una plataforma de diseño de hardware funcionando a partir del FPGA MAX10. Cuenta con una diversidad de componentes como interruptores, botones, LED, varios display de 7 segmentos, pines de propósito general, extensión con Arduino, convertidores analógico digitales, acelerómetro, conexión USB-blaster, conexión VGA.

Justamente gracias a las grandes capacidades del FPGA, de la mano con Intel, hemos podido entender y adentrarnos en el proceso del desarrollo de un prototipo. La idea central es que a través del reto de desarrollar un sistema ciber-físico (combinación software y hardware en dispositivos configurables conectados a través de sensores y actuadores físicos con el usuario y otros) podamos aprender a diseñar desde la planeación hasta la implementación y validación, con retroalimentación y guía de Intel sobre cómo se lleva a cabo este mismo proceso dentro de la industria profesional.

Reuniendo conocimientos básicos de previos semestres, nos enfocamos a aprender e implementar sistemas como compuertas y multiplexores en FPGA; latches, flipflops para control de flujo y salidas,; máquinas de estado en FPGA, para uso en componentes de hardware como display de 7 segmentos, pantalla de cuarzo LCD y VGA; comportamiento y programación en ensamblador Gumnut, con conexión a hardware como el acelerómetro o interruptores; uso de ALUs, y el entendimiento de protocolos de comunicación entre sistemas, como para la sincronización a VGA.

Análisis y descripción del problema

Más que resolver un problema, se requiere de utilizar todo lo aprendido en el lenguaje de diseño de hardware, VHDL para lograr realizar un videojuego arcade de Pong en un monitor VGA. Dentro de las mayores complicaciones del reto existen crear las condiciones para que se pueda visualizar un movimiento suave de todos los componentes movibles del pong y poder realizar un sistema que pueda desplegar el puntaje de cada jugador y el ganador de dicha partida. Eso por la parte visual, mientras que por parte del diseño, lograr una implementación completa con la DE10 lite en su sincronización con la pantalla mediante VGA, que pueda proveernos de un sistema de coordenadas para poder colorear pixeles en nuestra pantalla. Recordando también, que se trata de un sistema controlado por hardware, y hay que aprovechar la variedad de funciones que la tarjeta provee para lograr que el usuario movilice los gráficos de juego de la mejor manera posible.

Desarrollo de la propuesta de solución

Para la implementación de nuestra solución, debemos comprender antes que nada, el proceso de sincronización que ofrece la tarjeta con su salida VGA a una pantalla. Básicamente se tienen dos pulsos de sincronía: uno en vertical y uno en horizontal, seguidos por una zona negra back porch, luego la zona visible de la pantalla de 640 x 480 pixeles, y finalmente otra zona negra front porch.

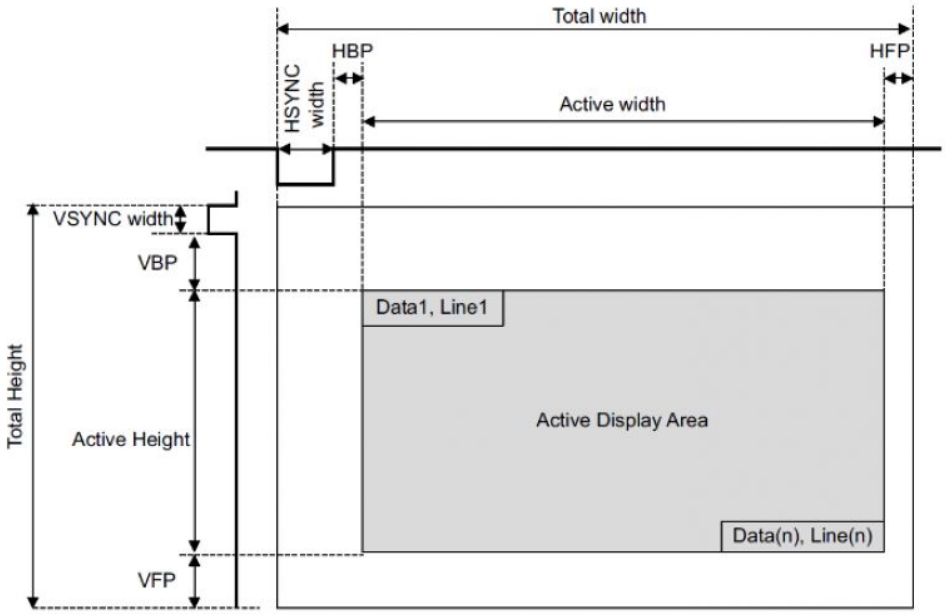


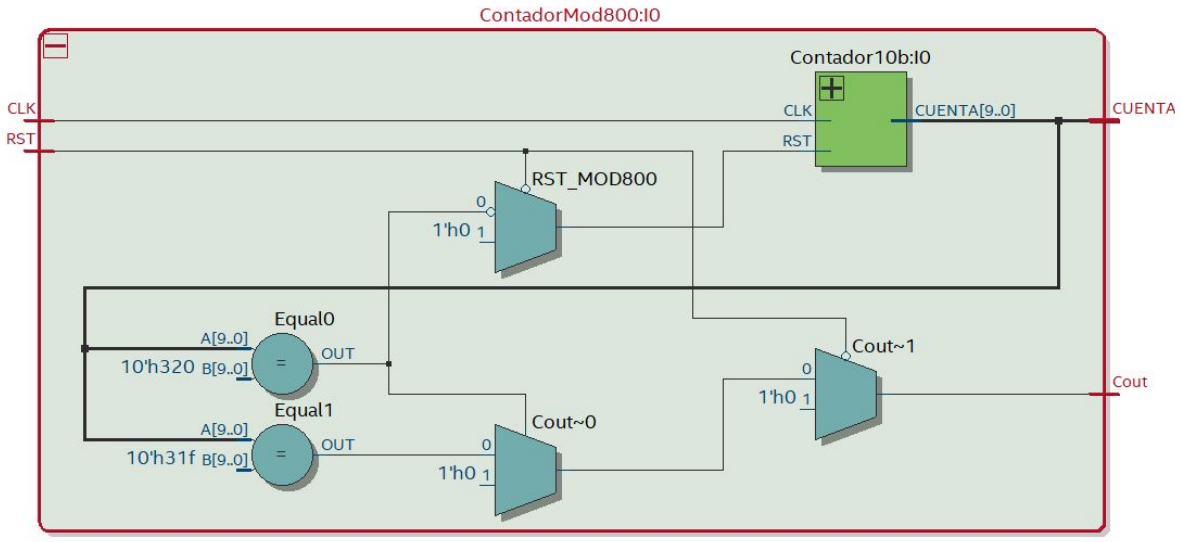
Imagen 1: Proceso de sincronización VGA (stepfpga.com)

De las especificaciones dentro del manual de nuestra tarjeta, el pulso de sincronía horizontal es de 96, el back porch de 48, seguidos los 640 visibles y 16 más del front porch: 800 en total. Para la sincronía vertical, pulso de sincronización de 2, back porch de 33, visible de 480 y 10 de front porch: 525 en total. Todos los pulsos en este proceso requieren de una señal de reloj con frecuencia 25 MHz.

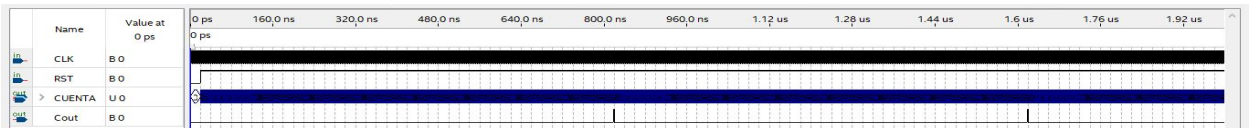
Una vez entendida la sincronización, lo primero que realizamos fueron dos contadores módulo 800 y 525 para poder controlar los sincronizadores vertical y horizontal. Estos contadores servirán para recorrer la pantalla y usarse como sistema de coordenadas x,y.

Ambos contadores se construyeron con cuentas de 10 bits, con base en 10 half adders, concatenando los carry out de cada uno al siguiente. De esta manera, se forma una entidad que solamente aumenta en 1 la cuenta de entrada. Para obtener una cuenta con módulo, se utiliza una señal de reset interno además del reset asíncrono, para reiniciar la cuenta cuando se de la combinación deseada.

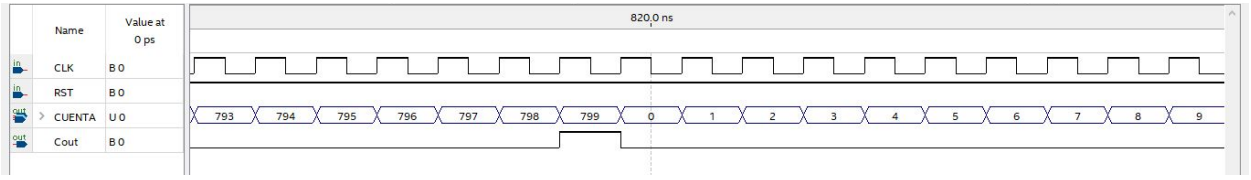
El contador módulo 800, se ve así en vista RTL:



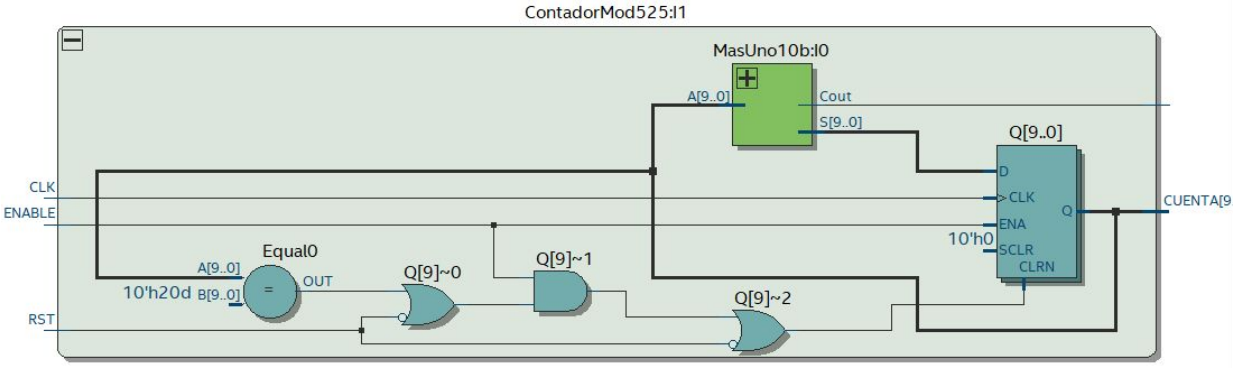
Utiliza un contador que avanza con la señal de CLK y cuenta con su reset interno para reiniciar la cuenta al llegar al 800 en 10 bits. El carry out de este contador se diseña para mostrarse en alto cuando se cumpla con la cuenta de 800 como se muestra en la siguiente simulación:



Misma simulación, con zoom en el módulo 800 y carry out alto:

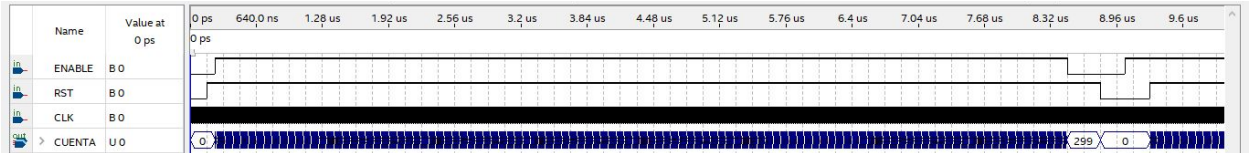


El contador módulo 525 se ve así en vista RTL:

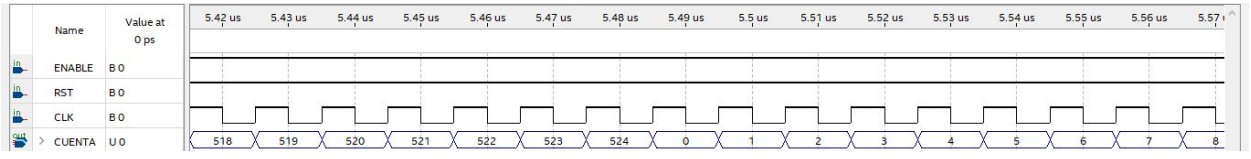


A diferencia del contador módulo 800, este contador no cuenta con una entidad contador de 10 bits, sino que utiliza un sumador 1 a la entrada, con un flip flop con señal de enable. Esta entidad recibe la misma señal de reloj para estar en sincronía con el otro contador, pero recibe también una señal de enable para aumentar la cuenta solamente cuando dicho enable se muestre en alto. El carry out de este contador funcionaría como overflow para algún otro componente, pero para este reto no lo necesitamos.

Su simulación es muy similar a la del contador mod 800:



Con zoom en el módulo 525:

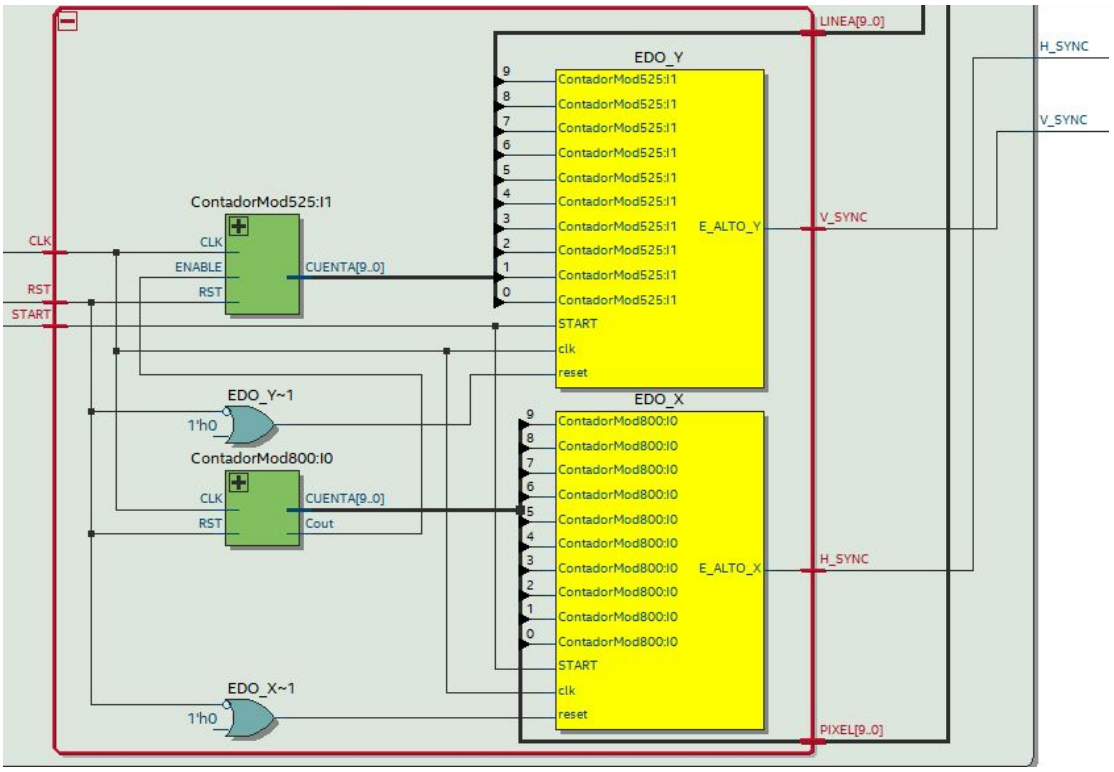


De esta manera, podemos colocar en cascada ambos contadores, de la forma que el carry out del contador módulo 800 funcione como señal de enable para el contador módulo 525, lo que en el

proceso de sincronía significa que por cada 800 píxeles recorridos en horizontal, o en eje x, apenas se recorre 1 en vertical, o eje y.

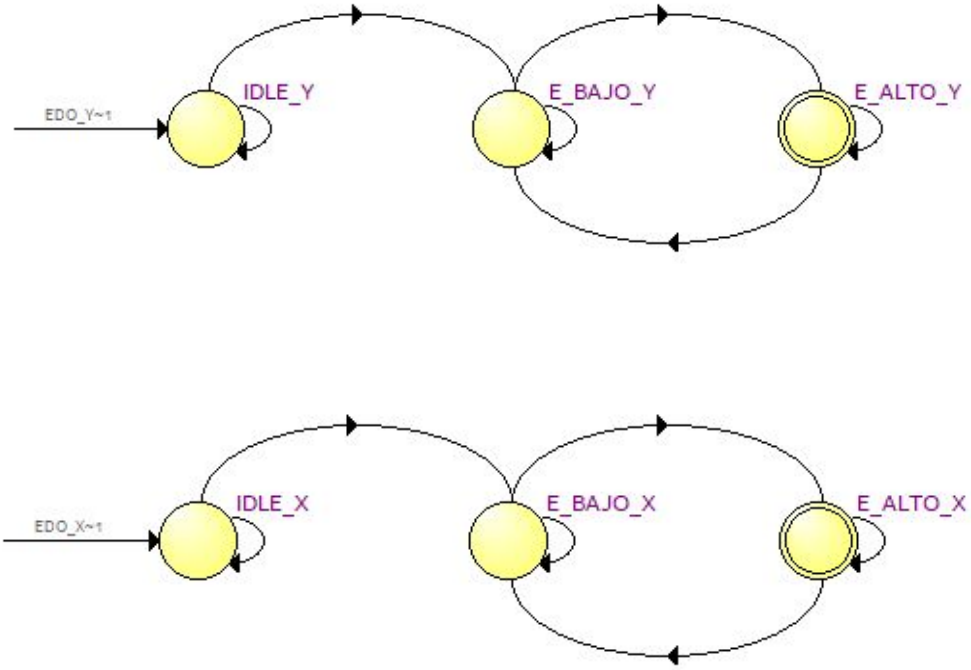
Ya con esta configuración, tenemos una manera de recorrer toda la pantalla, pero aún debemos de generar los pulsos de sincronía. Para ello, diseñamos dos máquinas de estados: una para el pulso de sincronía horizontal y otra para la sincronía vertical.

En vista RTL obtenemos lo siguiente:

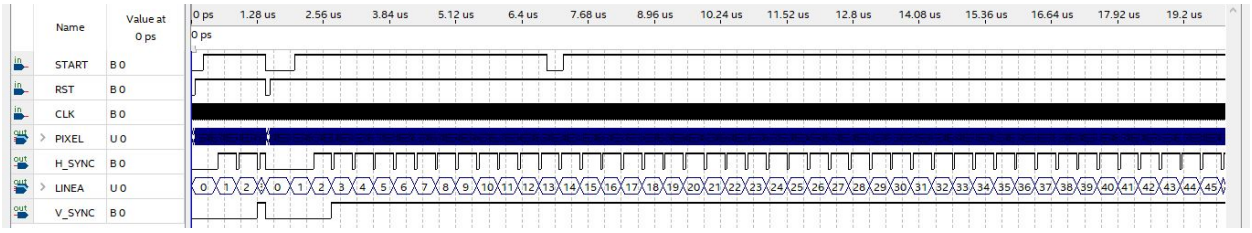


Las máquinas de estado son básicamente las mismas puesto que tanto en horizontal como en vertical, se cuenta con primero un pulso de sincronización, luego back porch, luego zona visible y finalmente un front porch. Lo que cambia son los pulsos, que en una máquina de estados se entienden como las transiciones.

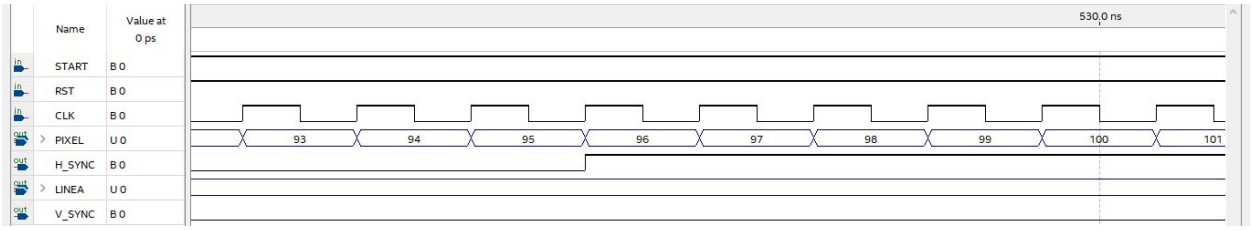
Las máquinas de estado, en vista State Machine:



La simulación de los pulsos de sincronización horizontal y vertical se ve así:

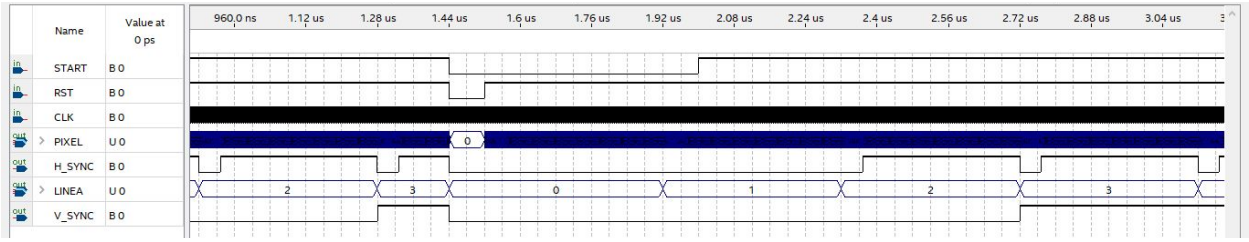


Primero zoom en la sincronización horizontal:



El pulso de sincronía se activa en alto a los 96 cambios de la señal de reloj.

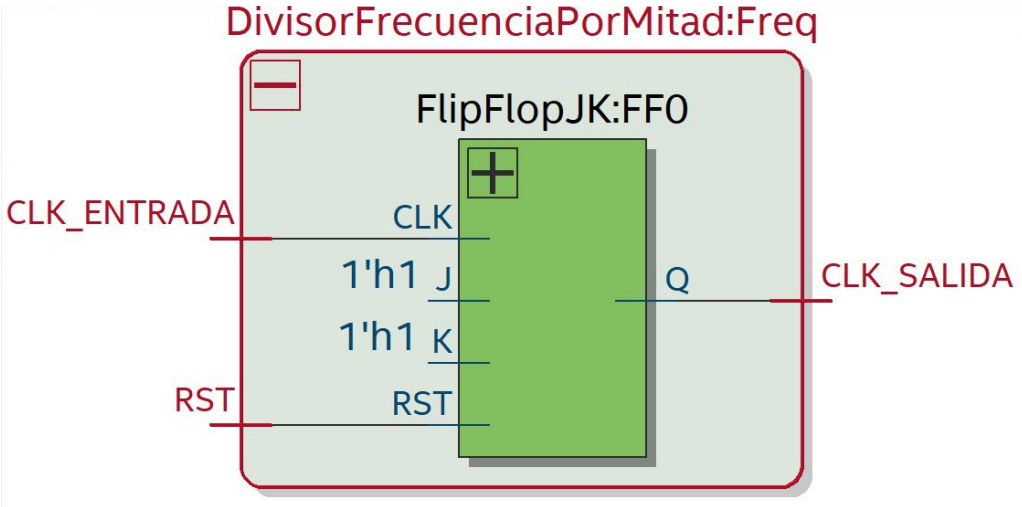
Con zoom en la sincronización vertical:



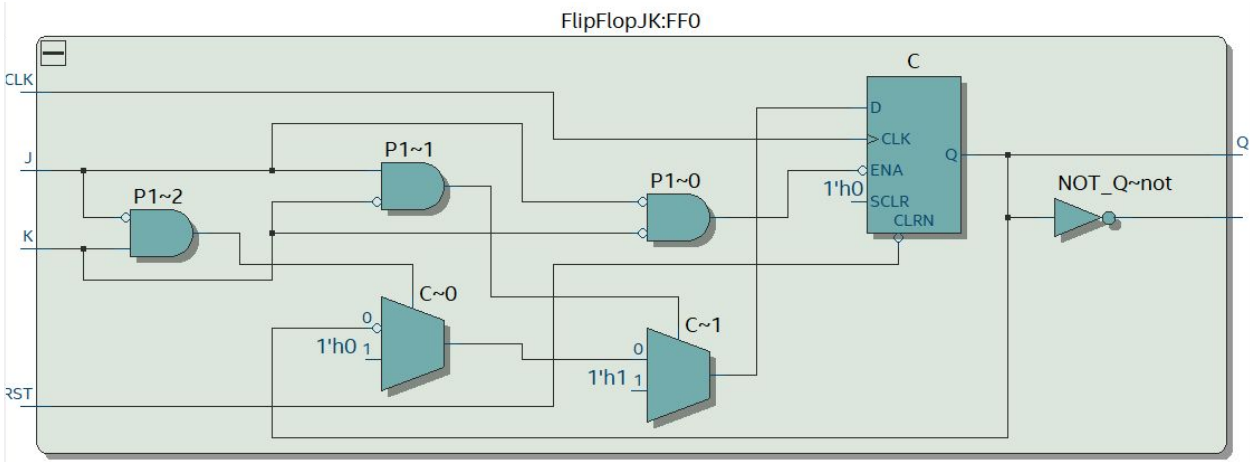
Cabe mencionar, que para esta entidad de sincronización, se introduce la señal START, que es la encargada de empezar la sincronía con la pantalla y recorrerla, es por ello que es la primera transición, para ir de estados IDLE a los primeros estados.

Ahora, recordando que la sincronización se debe realizar con una frecuencia de 25 MHz, para lograr dibujar en la pantalla, se creó un divisor de frecuencia por mitad con un Flip Flop JK para lograr dividir la señal de reloj de 50 Mhz que ofrece la tarjeta, en un clock de 25 MHz que nos permite adaptarnos a los 60 fps que utiliza el VGA.

Este componente es muy sencillo, pues solamente recibe el reloj de la tarjeta, pasa por el Flip Flop con valores de 1 en ambos J y K, y sale un reloj con frecuencia reducida a su mitad:

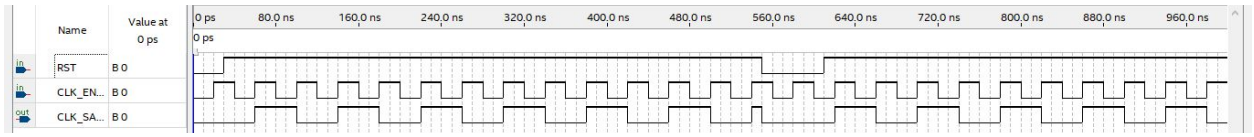


El Flip Flop tipo JK es un componente pequeño, que se diseña con base en su tabla de verdad, y se construye a partir de compuertas, puesto que Quartus solamente sintetiza Flip Flops tipo D:



Para este reto, no necesitamos la señal de salida Q negada.

Podemos comprobar la división de frecuencia con la siguiente simulación:



Donde podemos observar que ahora en la salida, el ciclo de reloj dura el doble.

Ya con todos estos componentes, procedemos a determinar el color de los pixeles. Mediante la tarjeta DE10-Lite se escogen los pines del sincronizador vertical y el horizontal, se indican los valores RGB en números binarios de 4 bits o sus representaciones en hexadecimal y con eso se escoge la combinación de colores para el pixel o el rango de píxeles que el usuario quiera.

Para realizar el juego Pong se propuso programarlo de forma comportamental (behavioral) para poder animar el movimiento de los paddles y la pelota del juego. Por otra parte, podemos agregar variables y constantes a nuestra discreción como lo son las velocidades de los paddles y la pelota. Éstas mismas se conectan a las diferentes entradas de la tarjeta para integrar al sistema ciber-físico, para que el usuario puede escoger, por ejemplo la dificultad del juego, aumentando la velocidad de la pelota, y desde luego controlar los paddles.

A grandes rasgos, la programación RGB del juego consta primero de todas las constantes, como los son los colores del fondo, de los paddles, de la pelota, también los tamaños de los visuales y los límites del juego. Luego dentro de un proceso, programamos primero la lógica de

movimiento, basado en condicionales sobre las entradas de los botones y las posiciones en coordenadas x,y de cada componente visual. Dentro de esta lógica se encuentra la lectura de los controles de los paddles, las colisiones, el movimiento de la pelota y los paddles, botón de reset, y la lógica de anotaciones con puntajes. Todos los condicionales, dentro de sincronía de reloj, y se actualizan con cada frame, es decir cada que los contadores recorren la pantalla entera y vuelven al origen (que es la esquina superior izquierda).

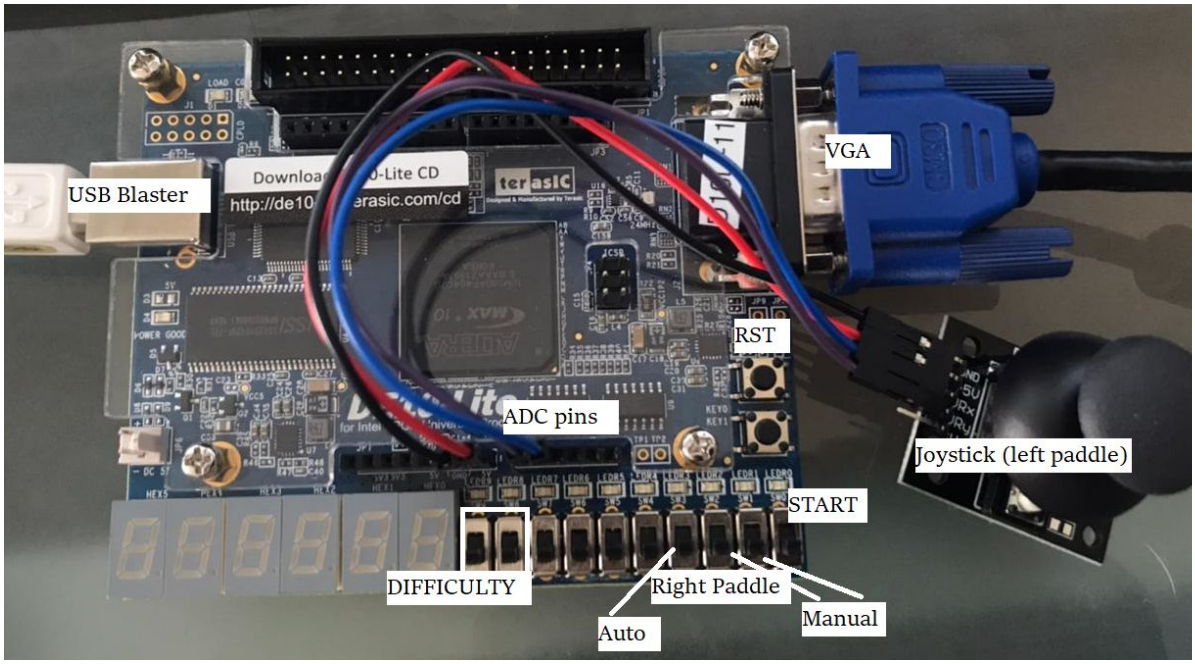
Finalmente se realiza el "coloreado" de los pixeles. Para los componentes, se considera la esquina superior izquierda como el punto central de cada componente, y se aprovechan las variables y constantes para dibujar grosor y altura de cada componente, con base en condicionales y ANDs, que "encierran" el área a dibujar desde la esquina hacia los extremos. Pensamos todo como vectores horizontales y verticales, dibujando primero en la base horizontal y luego la vertical.

Incluímos tres líneas blancas: los dos bordes inferior y superior, al igual que una línea divisoria al centro para que divida la cancha.

Se mantienen los puntajes por medio de variables y dependiendo del lado del tablero en que la pelota salga, se van aumentando los puntajes. A partir de estas variables, se dibujan el mensaje de ganador y perdedor. Con ello, se dibuja el puntaje y el número de jugador de cada uno en sus respectivas esquinas superiores, por medio de arreglos de píxeles rectangulares como si fueran un pixel art.

Por último, se dibuja una "w" del lado del jugador ganador, y una señal de "nulo" o "p de perdedor" horizontal para el perdedor.

El resultado de nuestra implementación del hardware consiste en la siguiente:



- Conexión USB blaster para el programa.
- Conexión de salida VGA.
- Conexión de entrada joystick para controlar la paleta izquierda, usando los pines convertidores de señal analógica a digital.
- Botón de reset para reiniciar juego o empezar uno nuevo.
- Interruptor START para empezar la sincronización y conexión con la pantalla VGA.
- Interruptores de control de la paleta derecha: un interruptor en alto para control automático de la paleta (para que juegue sólo un usuario contra el programa) y cuando en bajo, control manual mediante dos interruptores para subir y bajar la paleta.
- Par de interruptores para cambiar la dificultad:
    - 00: dificultad fácil y lenta. En esta dificultad, con la paleta derecha en control automático, no perderá nunca, apto para probar o “pelotear”.
    - 01: juego tendido en horizontal. La pelota tiene más velocidad en x que en y.
    - 10: juego tendido en vertical. La pelota tiene más velocidad en y que en x.
    - 11: juego difícil y rápido. La pelota se mueve igual de veloz en ambos ejes.

La paleta derecha en control automático básicamente no perderá a menos que la dificultad se encuentre en 10 o 11.

Finalmente para una demostración, tenemos la siguiente liga de vídeo demostrativo:https://drive.google.com/file/d/18WPeiUcm74iS8CjDnd_SinSyeYXQt_LH/view?usp=sharing

Conclusiones

Llegando al final de la materia nos parece sorprendente cuánto avanzamos en VHDL, de no saber nada, pasamos a crear un videojuego de Pong y solamente en 5 semanas. Tristemente esta materia ya terminó, sin embargo, sabemos que estos conocimientos nos serán de mucha utilidad en el futuro, ya sea para proyectos personales, trabajos u otras materias que se avecinan.

A lo largo de las entregas que realizamos en la materia, nos dimos cuenta que muchas de las cosas ya las habíamos empezado a ver los semestres pasados, y gracias a las herramientas con las que contamos esta vez, pudimos ver en sistemas físicos, y con vistas más visuales lo que es por ejemplo un ensamblador, o un protocolo de comunicaciones, complementando enormemente la comprensión de los temas. Sentimos que todo lo que hemos visto será el pilar formal para empezar nuestros estudios especializados de nuestra carrera. De primera instancia nos servirá para extender nuestros conocimientos con los siguientes bloques, y gracias a nuestro Socio Formador, podemos empezar a planificar nuestro camino a lo largo de esta carrera, para empezar a formarnos como profesionales de gran valor y voltear a ver a lo largo de la industria en donde se mueve un ingeniero en sistemas digitales.

Finalmente agradecimientos a Intel por proveernos de esta tarjeta de alto calibre y por compartirnos su conocimiento y experiencia. Agradecimientos a los Profesores de la materia por su gran apoyo, enseñanza y la mejor de las disposiciones para nuestro aprendizaje. Esperemos poder volvernos a encontrar y seguir programando cosas de carácter profesional con el hardware que nos prestaron.

Tecnológico de Monterrey

Referencias

Ashenden, P. J. (2007). Digital Design (Vhdl): An Embedded Systems Approach Using VHDL (Illustrated ed.). Morgan Kaufmann Publishers.

Ashenden, P. J. (2008). The Designer´s Guide to VHDL (Tercera Edición). Morgan Kaufman.

Components101. (2018, 2 abril). *Joystick Module Pinout, Features, Arduino Circuit & Datasheet*. https://components101.com/modules/joystick-module

JoyStick. (s. f.). EnergiaZero. Recuperado 10 de marzo de 2021, de http://www.energiazero.org/arduino_sensori/joystick_module.pdf

Pedroni, V. A. (2004). Circuit Design with VHDL. Books24x7.com.

Smith, W. D. & Range Voting.org. (2006). *Color codes*. RangeVoting. https://rangevoting.org/ColorCode.html

terasIC. (2003–2016). DE10-lite User Manual. Terasic Inc.

vga显示. (s. f.). StepFpga. Recuperado 10 de marzo de 2021, de https://www.stepfpga.com/doc/vga%E6%98%BE%E7%A4%BA%E6%A8%A1%E5%9D%97